Community Detection

Aitor Zubillaga
UPV/EHU
zubillaga012@ikasle.ehu.eus

Julen Etxaniz
UPV/EHU
jetxaniz007@ikasle.ehu.eus

ABSTRACT

%85 motibazioa, kontextua, proposamena, algoritmoa %15ondorioak esaldi bat

1 SARRERA ETA MOTIBAZIOA

Azken urteetan sare sozialek indar handia hartu dute. Arrazoi desberdin asko daude sare sozialen hazkundeen atzean, baina ukaezina da arrazoi nagusienetako bat, gure interesak partekatzen dituzten pertsonak aurkitzeko ematen dituzten erraztasunak direla [1]. Interesak edo erlazioak partekatzen dituzten pertsonen multzoei komunitate deitzen zaie. Pertsona multzo batean komunitateak aurkitzeko gai izan ezkero, posible da erabiltzaileari bere komunitate berdineko beste erabiltzaileei gogoko duten orrialdeen edo pertsonen gomendioak egitea, besteak beste. Hau horrela izanda, sare sozialen atzean dauden enpresek, interes handia erakutsi diote beraien erabiltzaileen artean komunitateak detektatzeko[2] modu egoki bat aurkitzeari. Komunitateak islatzeko orokorrean grafo egiturak erabiltzen dira, non pertsona edo erabiltzaile bakoitza nodo bat den. Komunitateak elkarri oso loturik dauden nodoen bidez adieraziko genituzke. Lotura hauek nodo arteko ertzak dira. Komunitate desberdineko kideen artean ere egongo dira arkuak baina askoz kantitate txikiagoetan.

1. irudian ikus daiteke adibide bat. Komunitateak nodoen koloreekin eta formekin adierazita daude eta ikus daitekeen bezala komunitate bereko nodoen artean ertz asko daude. Bestalde, komunitate desberdinekoen artean oso gutxi.

Gure probleman, NIPS kongresuan publikatzen duten autoreen komunitateak aztertu nahiko ditugu. Zehazki, 2014 eta 2015 urteen bitartean, kongresu honetan eman diren elkarlan komunitate desberdinak aurkitu nahi ditugu. Komunitateak hainbat artikulu batera idatzi dituzten autoreek osatuko dituzte. Eta komunitate ezberdineko autoreen artean elkarlan gutxi egongo dira.

Bilaketa espazioa $\{1, \ldots, k\}^n$ da, non k komunitateak eta n autoreak diren. Autore bakoitza zein komunitatekoa den zehaztuko dugu kodeketa horrekin. Beraz, autoreak komunitatetan banatzen dituen partizio onena aurkitzea izango da helburua. Onena zein den jakiteko, helburu-funtzio egoki bat definitu beharko dugu.

Lan honen helburua grafoetan komunitateak aurkitzeko bi algoritmo diseinatu eta aztertzea da. Bata soluzio bakarrean oinarritutakoa izango da, eta bestea poblazionala. Lortutako emaitzak aztertzeko modularitatea erabiliko da. Modularitateak sare baten komunitate banaketaren egokitasuna neurtzen du. Banaketa egokia izango da komunitateen barruan ertz asko badaude, eta komunitate artean gutxi [2].

Izan bedi auzokidetasun matrizeko A_{vw} elementua, ertzen arteko pixuekin:

$A_{vw} = v$ nodotik w nodora doan ertzaren pisua.

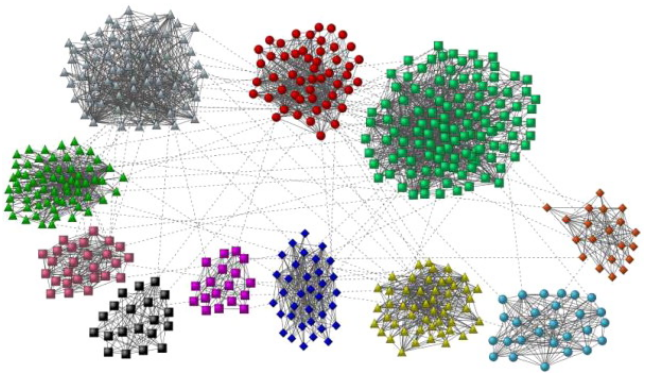

Figure 1: Komunitateak grafo batean

eta suposa dezagun erpinak komunitatetan banaturik daudela non v ren komunitatea c_v den. Beraz, komunitate berdinean erortzen diren pisuen frakzioa honakoa da

$$\frac{\sum_{vw} A_{vw}\delta(c_v, c_w)}{\sum_{vw} A_{vw}} = \frac{1}{2m}\sum_{vw} A_{vw}\delta(c_v, c_w),$$

non δ-funtzioa $\delta(i, j)$ en emaitza 1 den $i = j$ bada eta 0 bestela, eta $m = \frac{1}{2}\sum_{vw} A_{vw}$ grafoko ertzen pixuen batura den. Kantitate hau handia izango da sarearen banaketa onentzat, hau da, komunitateen barnean ertz asko badaude, baina hau ez da nahikoa banaketa ona den jakiteko. Adibidez, balio posible haundiena kasu tribial batean hartzen duelako, ertz guztiak komunitate berdinaren direnean, hau da, komunitateko bakarra dagoenean. Hau horrela, kantitate berdineko ausazko sare batengandik espero den balioa kentzen badiogu, neurri erabilgarri bat lortzen dugu.

Pixudun sare bateko v erpin baten k_v *graduak* erpin honi lotutako ertzen pixuen batura da.

$$k_v = \sum_w A_{vw}.$$

Konexioak ausaz eginez baina erpinen graduak errespetatuz, v eta w erpinen artean ertz bat existitzeko probabilitatea $k_v k_w/2m$ da. Q modularitatea honela definitzen dugu

$$Q = \frac{1}{2m}\sum_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\delta(c_v, c_w).$$

Komunitate barruko ertzen frakzioa ausazko sare batetik espero dezakegunaren desberdina ez bada, kantitate hau 0 izango da. 0 ez diren balioek ausazkotasunetik debiazioa errepresentatzen dute, eta praktikan 0.3 balioa sarearen komunitate estruktura ona denaren adierazgarria da. Beraz, gure helburua modularitatea maximizatzea izango da.

Gure algoritmoa sinplifikatzeko hurrengo bi kantitate definitko ditugu:

$$e_{ij} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, j),$$

i komunitatetik hasita j komunitatea lotzen duten ertzen pisuen frakzioa, eta

$$a_i = \frac{1}{2m} \sum_{v} k_v \delta(c_v, i),$$

amaiera i komunitateko nodoetan duten pisuen frakzioa. Beraz, $\delta(c_v, c_w) = \sum_i \delta(c_v, i)\delta(c_w, i)$, idatzi ezkero, hurrengoa dugu, (4) Eq.-tik

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \sum_i \delta(c_v, i)\delta(c_w, i)$$

$$= \sum_i \left[\frac{1}{2m} \sum_{vw} A_{vw}\delta(c_v, i)\delta(c_w, i) - \frac{1}{2m} \sum_v k_v \delta(c_v, i) \frac{1}{2m} \sum_w k_w \delta(c_w, i) \right]$$

$$= \sum_i (e_{ii} - a_i^2).$$

2 ALGORITMOEN DISEINUA

Soluzio bakarreko algoritmo moduan ILS bat inplementatu dugu. Algoritmo poblazional moduan GA-EDA algorimoa aukeratu dugu. Bi algoritmoek hasierako soluzioak lortzeko metodo eraikitzaile estokastiko bat erabiltzen dute, Louvain algoritmoan oinarritzen dena. Beraz, hasteko Louvain azalduko dugu eta ondoren besteak.

2.1 Louvain

Louvain algoritmoaren [1] pausuak 2. irudian ikus daitezke. Iterazio bakoitzak bi fase ditu: batean komunitateen aldaketa lokala bakarrik eginda modularitatea optimizatzen saiatzen da eta bestean, aurkitutako komunitateak agregatzen dira komunitateen sare berri bat sortzeko. Iterazio hauek modularitatea hobetzea posible ez den arte errepikatzen dira.

Community moduluko `generate_dendogram` funtzioak komunitateak elkartzen ditu bakarrik hobetzen bada. Beraz, komunitate kopuru optimora iritsitakoan gelditu egiten da. Guri interesatzen zaigu emandako komunitate kopurua izatea, ez optimoa.

Beraz, helburuko komunitate kopurura iritsitakoan algoritmoa geratzen dugu. Gainera, aldaketa batzuk egin ditugu komunitate kopurua gehiago jaitsi ahal izateko modularitatea asko txartu gabe. Horrela, 292 komunitate ingurutik 279 komunitate ingurura jaistea lortzen dugu.

Hala ere, honek ez du balio emandako komunitate kopurua 279 baino txikiagoa bada. Izan ere, komunitateen artean ertzik ez dagoenean algoritmoa geratu egiten da. Hurrengo atalean komentatuko dugu nola lortu dugun txikitzea.

Community moduluko `best_partition` edo gurea erabiltzen badugu antzeko arazoa daukagu, komunitate kopurua ezin da nahi adina jaitsi. Hori konpontzeko, `best_fixed_partition` funtzioa inplementatu dugu. Honek, lehenengo aurreko funtzioari deituko dio. Gero, soberan geratzen diren komunitate txikienak hartu eta guztiak komunitate berean elkartuko ditu. Izan ere, komunitateen artean konexiorik ez dagoenean, modularitate hobea lortzeko modurik onena komunitate txikienak elkartzea da, komunitate handiak eragin handiago baitute modularitatean eta hauek txartzeak beraz, asko jaisten du fitnessa. Komunitate txikienak non elkartu aukeratzeko, aukera guztiak aztertu eta onena aukeratzen da.

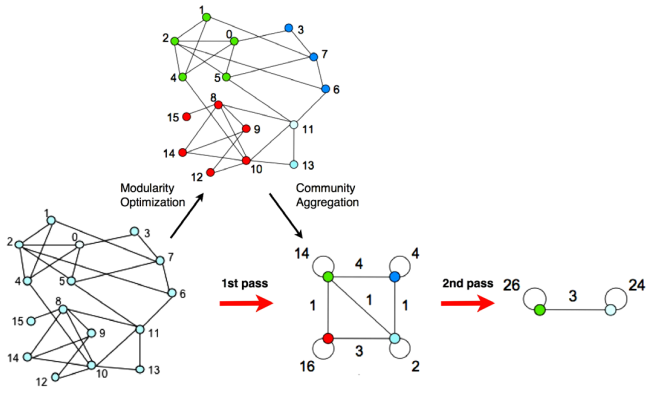

Figure 2: Louvain

2.2 ILS

Iterated Local Search [3] soluzio bakarreko algoritmoa inplementatu dugu . Hasierako soluzioa sortzeko Louvain eraikitzailea erabiltzen da. Ondoren, perturbazioak sartu, bilaketa lokala egiten da eta soluzioa eguneratzen da. Ikusi 1. algoritmoa.

Algorithm 1: ILS

Input : *local_search* bilaketa algoritmoa, *perturbation* perturbazioa, *simmulated_annealing* onarpen-baldintza eta *stop_criterion* gelditzeko irizpidea
Output : s^* soluzioa

```
while !stop_criterion() do
    s* = perturbation(s*)
    s = local_search(s*)
    s* = simmulated_annealing(s)
end
```

Perturbatzeko, lehenik soluzioan 2 komunitate ausaz hartu eta elkartu egiten dira, eta ondoren komunitate bat ausaz hartu eta bi zatitan banatzen da. Zati hauek ausaz aukeratzen dira. Funtzioak jasotako graduak, soluzioa zenbat aldiz perturbatuko den zehazten du.

Behin perturbazioa sortuta, soluzio honen optimo lokala aurkituko da best first estrategia eta Swap ingurune funtzioa erabiliz. Emaitza eguneratzeko Simulated Annealing estrategia erabiliko dugu, emaitza txarragoak onartu ahal izateko. Gero eta tenperatura handiagoa, orduan eta probabilitate gehiago egongo da emaitza txarrak onartzeko. Hasierako tenperatura eta tenperatura egokitzeko modua aukeratu beharko ditugu.

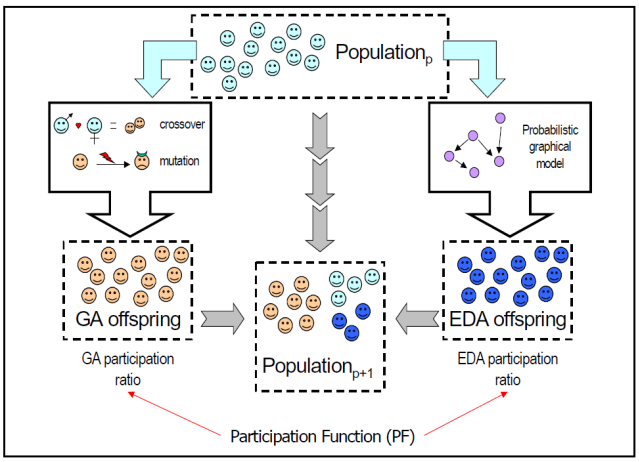

Figure 3: GA-EDA

2.3 GA-EDA

GA-EDA [4] populaziotan oinarritutako algoritmoa inplementatu dugu. Algoritmo honek Genetic Algorithms eta Estimation of Distribution Algorithms konbinatzen ditu. Bi algoritmo hauek antzekotasun asko dituzte, baina biak konbinatzean populazioetan dibertsitate gehiago eta emaitza hobeak lor daitezkeela ikusi dugu. Ikusi 2. algoritmoa eta 3. irudia.

Hasteko, hasierako populazioa eraikiko dugu, ausaz edo moldatutako louvain heuristikoa erabiliz. Gero, hainbat iterazio egingo ditugu ebaluazio kopuru maximora iritsi arte. Iterazio bakoitzean pauso berdinak errepikatuko dira. Amaitzeko, populazioko soluzio onena itzuliko dugu.

- Selection eragiketan hainbat soluzio aukeratuko ditugu aurreko populaziotik. `best_selection` aukeraketan beti soluzio onenak aukeratzen dira. `roulette_selection` aukeraketan soluzioen probabilitateak kalkulatzeko fitness balioa erabiltzen du. `rank_selection` aukeraketan soluzioen probabilitateak kalkulatzeko ranking-eko posizioa erabiltzen da. `tournament_selection` aukeraketan hainbat txapelketa egiten dira soluzioak ausaz aukeratuz.
- Crossover eragiketan aukeratutako soluzioak birkonbinatzen dira soluzio berriak sortzeko. `one_point_crossover` algoritmoan soluzioko puntu bat aukeratzen da. Puntu horretatik eskuinera dagoen soluzio zatia trukatzen da bi soluzioetan soluzio berriak sortuz.
- Mutation eragiketan, aurreko soluzioei mutazio bat egiten zaie probabilitate batekin. `bit_mutation` eragiketan soluzioen posizio bakoitzaren balioa ausaz aldatzen da.
- Learning eragiketan, aukeratutako soluzioak erabilita eredu probabilistikoaren parametroak ikasten dira.
- Sampling eragiketan, ikasitako eredutik abiatuta soluzio berriak lagindu eta ebaluatzen dira.
- Update eragiketan, soluzio berrien eta aurrekoen artetik soluzio batzuk gordeko ditugu. `elistism_update` eragiketak aurreko populazioko soluzio onena gordetzen du eta populazio berriko onenak. `elistism_update` eragiketak aurreko populazioko eta populazio berriko soluzioak elkartzen ditu eta onenak aukeratzen ditu.

Algorithm 2: GA-EDA

Input : $evaluate$, $select$, $learn$, $reproduce$, $sample$, $update$ eta $stop\_criterion$ operadoreak, P_0 hasierako populazioa
Output: s^* soluzioa

while !$stop\_criterion()$ **do**
  $f_t = evaluate(P_t)$
  $P_t^s = select(P_t, f_t)$
  $P_t^g = reproduce(P_t^s)$
  $P_t^e = sample(M_t)$
  $P_{t+1} = update(P_t, P_t^g, P_t^e)$
  $t = t + 1$
end
s^* = Populazioko soluziorik onena

3 ESPERIMENTAZIOA

Esperimentazioak bi fase izango ditu. Lehenengo fasean, ILS eta GA-EDA algoritmoen parametroak kalibratuko ditugu. Bigarren fasean, algoritmoak konparatuko ditugu 2 komunitatetik 100 komunitatera arte. Beti ere, jakiteko gure algoritmoak onak edo txarrak diren, RS baseline algoritmoa erabiliko dugu.

3.1 Parametroak kalibratu

Parametroak kalibratzeko Grid Search erabili dugu. Parametro garrantzitsuenetan hainbat balio posible definitu ditu eta beraien arteko konbinaketa denak probatu. Komunitate kopuruari dagokionez, muturreko kasuak bakarrik probatu ditugu, 2 eta 100 komunitate. 50 komunitaterekin ere probatu dugu baina antzekoak zirenez denbora aurrezteko kendu dugu. Gainerako parametroak gure ustez egokia den balio estatiko batean utzi ditugu. Izan ere, exekuzioak egiteko denbora asko behar da, eta luzea da parametro guztiak kalibratzea.

Parametro aukera bakoitzerako 10^4 ebaluazio eta 5 errepikapen egin ditugu. Errepikapen bakoitzeko fitness eta denbora balioaekin batazbestekoak kalkulatu ditugu. Fitness-en batazbesteko balioekin heatmap motako grafikak atera ditugu.

Ikusi 4. irudia.

ILS algoritmoan komunitate kopuru bakoitzerako 6 parametro konbinazio probatu ditugu. Zehazki, perturbazio gradurako $\{1, 3, 5\}$ balioak eta simmulated annealing-en hasierako tenperaturarako $\{0.0001, 0.001\}$. Tenperatura eguneratzeko 0.9 balioarekin bidertzen dugu beti. Bi komunitate kopuruetarako emaitza onena 3 eta 0.001 parametroekin lortu ditugu. Beraz, esperimentazioaren hurrengo faserako parametro horiek erabiliko ditugu.

GA-EDA algoritmoan, berriz, 4 parametro konbinazio probatu ditugu. Populazio tamainarako $\{30, 40\}$ eta aukeraketa tamainarako $\{15, 20\}$. Komunitate kopurua 2 denean, 30 eta 15 balioekin lortzen da fitness altuena. 100 komunitaterekin onena ez den arren, denak antzekoak dira. Gainera, denbora aldetik azkarrena da, soluzio kopurua txikiagoa delako. Ondorioz, parametro horiek erabiliko ditugu konparaketan.

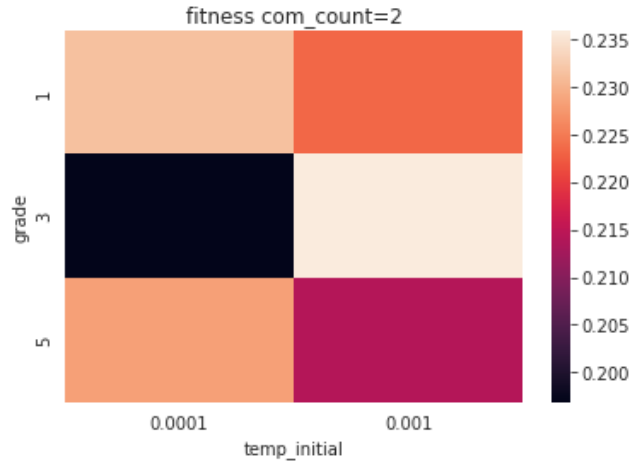

Figure 4: Heatmap

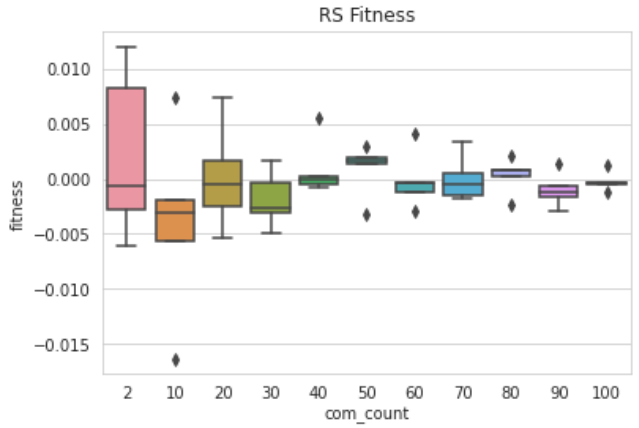

Figure 5: Boxplot Fitness

3.2 Algoritmoak konparatu

Hiru algoritmoen konparaketa egiteko, komunitate kopuru bakoitzerako 10^4 ebaluazio eta 5 errepikapen egin ditugu. Guztira 11 komunitate kopuru desberdin probatu ditugu, 2, 10, 20, ..., 100. Emaitzekin boxplot motako grafikak atera ditugu, x ardatzean komunitate kopurua jarriz.

Ikusi **??**. eta **??** irudiak.

4 ONDORIOAK

%15 motibazioa %85 azaldu proposamena + experimentuak + future work + limitations

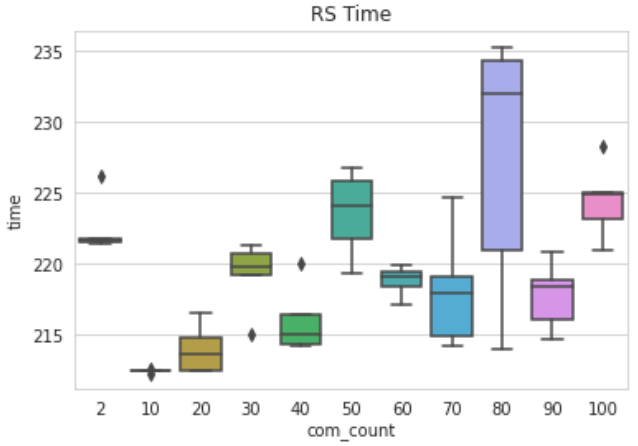

Figure 6: Boxplot Time

REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[2] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E* 70, 6 (2004), 066111.

[3] Chao Liu, Qinma Kang, Hanzhang Kong, Wenquan Li, and Yunfan Kang. 2020. An iterated local search algorithm for community detection in complex networks. *International Journal of Modern Physics B* 34, 04 (2020), 2050013.

[4] José M Peña, Victor Robles, Pedro Larranaga, Vanessa Herves, Francisco Rosales, and María S Pérez. 2004. GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 361–371.