Komunitate-detekzio problema NIPS kongresuko autoreekin

Alexander Diez
Euskal Herriko Unibertsitatea
adiez134@ehu.eus

Xabier Irastorza
Euskal Herriko Unibertsitatea
xirastorza003@ehu.eus

Iñigo Salazar
Euskal Herriko Unibertsitatea
isalazar030@ehu.eus

ABSTRACT

Paper honetan Community Detection Problem (CDP) ebazteko hainbat algoritmo inplementatu eta alderatu ditugu. Ahal bezain beste algoritmo ezberdin alderatu ahal izateko bilaketa lokaleko algoritmoak, eraikitzaileak eta poblazionalak aztertu ditugu. Ondoren, kategoria horietatik errendimendu hoberena duen algoritmoa aukeratuz, beste kategorietakoekin konparatzeko.

ACM Reference Format:
Alexander Diez, Xabier Irastorza, and Iñigo Salazar. 2022. Komunitate-detekzio problema NIPS kongresuko autoreekin. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SARRERA

Komunitate-detekzioaren problema (CDP) [2] sare konplexuen analisiaren problema oso ezaguna da, hainbat arlotan aplikatu dena, hala nola gizarte-zientzietan, neurozientzian eta biologian... Komunitate estrukturak interes handikoak dira, haien ezaugarri propioetaz baliatuz, sareak aztertzea errazten dutelako. Honek eragin handia izan dezake, adibidez, epidemien hedaketan bezalako interes handiko arloetan, honek orokorrean komunitate-egituren eragina izaten duelako.

Paper honetan hainbat algoritmo proposatzen eta alderatzen ditugu problema hau ebazteko. Bilaketa lokaleko algoritmoak, algoritmo poblazionalak eta algoritmo eraikitzaileak aztertu ditugu. Erabili ditugun algoritmoak zabal aipatzen eta aztertzen dira literaturan, eta, beraz, paper honetarako egokiak zirela ondorioztatu genuen.

Bilaketa lokalekoetatik suberaketa simulatua [3] eta deabruaren algoritmoa aukeratu genituen, beste problema batzuetarako emaitza oso onak lortu izan ditugulako algoritmo horiekin. Poblazionaletatik, algoritmo genetiko bat eta Estimation of Distribution Algorithm (EDA) [4] bat aukeratu genituen.

Paperaren gainerakoa honela antolatuta dago. Hurrengo atalean CDP-ri buruzko oinarrizko ezagutzak azaltzen dira. Paper honen ekarpena 3.atalean aurkezten da, CDP ebazteko erabilitako algoritmoak. Ondoren, esperimentazioa proposatutakoa egiaztatzen saiatzen dena. Gero, emaitzen analisi estatistikoa esplikatuta dago. Azkenik, ateratako ondorioak erakuzten ditugu.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 OPTIMIZAZIO PROBLEMA

2.1 Problemaren formalizazioa

Paper honetan ebatziko dugun problema, CDP problemaren instantzia espezifiko bat da. Zehazki, Neural Information Processing Systems (NIPS) kongresuan 2014 eta 2015 urteen artean publikatu duten autoreen arteko komunitateak aurkitu nahi ditugu. Autoreak sarean nodo bezala errepresentatuko dira eta haien arteko elkarlanak, nodoen arteko ertz bezala.

Literaturan bilatzean, aurkitu genuen ez dagoela unibertsalki onartuta dagoen komunitateen definiziorik, baina badaude gidalerro bezalako batzuk: komunitateko nodoen artean ertz gehiago egon behar dira, komunitateko nodoak komunitatean ez dauden beste nodo batzuekin lotzen dituzten ertzak baino. Hori kontuan hartuta, paper honetan komunitatearen hurrengo definizio hau erabiltzea erabaki genuen.

Komunitate bat gai antzekoari buruz elkarlanean aritu diren autoreek osatzen duten grafoko azpimultzo bat da. Komunitate bat gutxienez 2 autorek osatzen dute eta horren barruan dauden autoreek haien artean lan asko eginak izan behar dituzte eta beraz, haien artean lotura asko eta pisutsuak (hau da, bi autoreren artean egindako lan kopuru altua) egongo dira. Beste komunitateetako autoreekin konexioak egon daitezkeen arren, hauek gutxiago eta pisu gutxiagokoak izango dira.

Autoreak K komunitateetan banatuko dira. Gure probleman milaka autore daude eta horien artetik beste autoreekin loturarik ez duten autoreak egon daitezke edo lotura oso gutxi dituztenak (bikoteak, hirukoteak, etab.). Arazoa dago autore-multzo oso txiki hauek sailkatzean zein komunitatekoak diren erabakitzean. Guk autore multzo hauek aparteko komunitate batean sartuko ditugu, beste komunitateetan sartzen baditugu ez dugulako jakingo benetan komunitate horretan dauden beste autoreekin asko lan egin duten edo beste autore oso gutxirekin lan egin duten pertsonak diren.

Komunitateen barruan normalean azpikomunitateak daude, hau da, haien artean besteekin baino harreman gehiago dituzten azpimultzoak, eta izan daiteke azpikomunitate horien barruan azpikomunitate gehiago egotea, adibidez, giza gorputzean komunitate bat organoak izango lirateke, organoen barruan ehunak izango lirateke komunitateak, ehunen barruan zelulak etab. Beraz, sortuko ditugun komunitateak ez dira zertan tamaina berdin-berdinekoak izan, baina bai tamaina antzekoak, maila berdineko komunitateak sailkatu nahi baititugu.

Hori dela eta, komunitateak grafoko azpigrafo konektatu handienetan bilatuko ditugu. Dena den, azpigrafo hoietatik kanpo dauden beste azpigrafo konektatu txikiagoak baldin badaude azpigrafo konektatu handienetan sortuko ditugun komunitateen tamainakoak, azpigrafo konektatu txiki horiek ere komunitate bezala sailkatuko ditugu. Horrela, bat izan ezik, (konexio gutxi dituzten autoreen komunitatea) komunitate denak tamaina antzekokoak izango dira eta

ahalik eta komunitate gehien hartuko ditugu kontuan, ez bakarrik azpigrafo konektatu handienetan daudenak.

Soluzioak errepresentatzeko 0-tik k-rako balioz betetako bektorea egokiena iruditu zaigu, non indizeak autorea errepresentatzen duen eta zenbakiak autore hori dagoen komunitatea.

2.2 Helburu funtzioa

Erabili genuen helburu-funtzioa Clauset, Newman eta Moore-k proposatutako modularitatearen oso antzekoa da [1]. Problema honen grafoko ertzek pisuak dituztenez, apur bat moldatu behar izan genuen. Clauset et.al-ek proposatutako helburu-funtzioa hurrengoa da: $\sum_i (e_{ii} - a_i^2)$, non e_{ij} i komunitateko nodoak j komunitateko nodoekin konektatzen dituzten **ertzen** frakzioa den eta a_i i komunitateko nodoetara konektatuta dauden **ertz-muturren** frakzioa den.

Guk erabili genuena berdina da; $\sum_i (e_{ii} - a_i^2)$, baina e_{ij} i komunitateko nodoak j komunitateko nodoekin konektatzen dituzten ertzen **pisuen** frakzioa da eta a_i i komunitateko nodoetara konektatuta dauden **ertz-muturren pisuen** frakzioa. Helburu-funtzio hau maximizatuko dugu.

Helburu-funtzioan e_{ii}-k, i komunitateak barnean dituen ertzen pisuen frakzioa (barruko ertzen pisuen gehiketa / grafoko ertz guztien pisuen gehiketa) errepresentatzen du eta a_i-k, i komunitateak barneko eta kanpoko nodoekin dituen ertzen muturren pisuen frakzioa. Hau honela izanda, i-ko nodoen arteko ertzen muturrak birritan kontatuko dira (bat nodo bakoitzeko). a_i-ren balioa (frakzioa) beti 0 eta 1 artean egongo denez, berretzean ia beti balio hori txikituko da. Hau beharrezkoa da, e_{ii} beti delako a_i^2 baino txikiagoa edo berdina, definizioz. Horrela, komunitateek barnean ertz asko dituztenean eta kanpoan gutxi, balio altuenak emango ditu helburu-funtzio honek.

3 EKARPENA

3.1 Algoritmo eraikitzailea

Algoritmo eraikitzaileak grafoan eskatutako partizio adina nodo hautatu eta horiek epizentro gisa hartuta partizioak sortzen ditu. Jarraian sasikodea:

```
hasierako nodoak hautatu ausaz
nodo mugakideak partiziorik gabeko nodoen zerrendara gehitu
while not partiziorik gabeko nodoen zerrenda hutsa do
    partiziorik gabeko nodoa hautatu
    mugakide duen lehen partiziodun nodoaren partizioa esleitu
end while
itzuli soluzioa
```

3.2 Suberaketa simulatua

Suberaketa simulatua bilaketa lokal arruntaren hedapen bat da, oinarrian Kirkpatrick [3] eta Cerny-k proposatua. Bilaketa lokalen esplorazio eta esplotazio gaitasunak handitu eta koordinatzeko duen gaitasunagatik da ezauguna. Soluzio kaskarragoak onartu ditzake horretarako, baina azterketa-prozesuaren hastapenetan gehiagotan eta gerora gutxiagotan. Hona hemen sasikodea:

```
tenperatura eta soluzio bat ausaz aukeratu
soluzio horren ingurunea kalkulatu
while tenperatura > hozte_tenperatura do
    soluzio onenaren ingurunea berreskuratu
    while oreka_it_kop < oreka_it_max do
        ingurunetik soluzio bat ausaz aukeratu
        if soluzioa inoizko onena da then
            gorde bera eta bere ingurunea
            hasieratu oreka_it_kop
        else
            erabaki onartu edo ez
            eguneratu oreka_it_kop
        end if
    end while
    eguneratu tenperatura
end while
itzuli soluzio onena
```

3.3 UMDA

UMDA algoritmoak soluzioaren probabilitate distribuzioa estimatzen saiatzen da. Helburu-funtzio balio hoberenak lortzen dituzten soluzioak hobetsiz probabilitate distribuzio bat poblazioaren arabera aldatzen du, konbergitzen duen arte.

Ulergarritasunaren mesedean, UMDAren sasi-kodea ematen dugu.

```
banaketa modeloa uniformeki hasieratu
banaketa modeloa laginduz pop_size tamainako populazioa eraiki
while iterazioak < iterazio_maximoa do
    banaketa modeloa eguneratu populazioaren arabera
    banaketa modelotik pop_size/2 soluzio berri lagindu
    populazioa eta soluzio berriekin populazio berria sortu
end while
populazioko soluzio hoberena bueltatu
```

4 ESPERIMENTAZIOA

Inplementatu genituen bost algoritmoen artean esperimentazioan hiru baino ez ditugu konparatuko: Algoritmo eraikitzailea, suberaketa simulatua eta UMDA. Honela, inplementatu genituen hiru algoritmo motetatik bat kontuan hartuko dugu. Esperimentazioa burutzeko erabili ditugun baliabide guztiak eskuragarri dago GitHuben.

4.1 Instantzia generadoreak

Grafoa generadorera sartu baino lehen grafoa prozesatzen genuen nodo bakartiak eta multzo txikietan daudenak kentzeko, horiek ez genituelako nahi instantzia generatuetan, erredundantzia sortzen baitute. Instantzia generatzeko, instantzia generadoreak grafoko nodo bakoitzeko, nodo horrek beste nodo ezberdinekin dituen ertzak kontatzen zituen eta horren arabera lagintzen zituen; ertz kopuruarekiko ponderatua. Sortzen ziren instantziak berriz prozesatzen genituen, geratu ahalko liratekeen nodo bakartiak edo multzo txikietan daudenak kentzeko eta erabiliko genituen multzoak errealetatik gertuago egon zitezen.

4.2 Parametroen aukeraketa

Parametroak ajustatzeko grid search egin genuen. Algoritmo bakoitza grafoko 3 instantzia desberdinen gainean hiru (suberaketa simulatuan sei) exekuzio egin genituen (instantzien tamainak 159, 212

eta 265 nodokoak), algoritmo horren parametro konbinazio desberdin bakoitzeko. Suberaketa simulatuan parametroen 49 konbinazio ezberdin probatu genituen instantzia bakoitzean. UMDAk parametro bakarra duenez, 140 balio ezberdin probatu genituen instantzia bakoitzean.

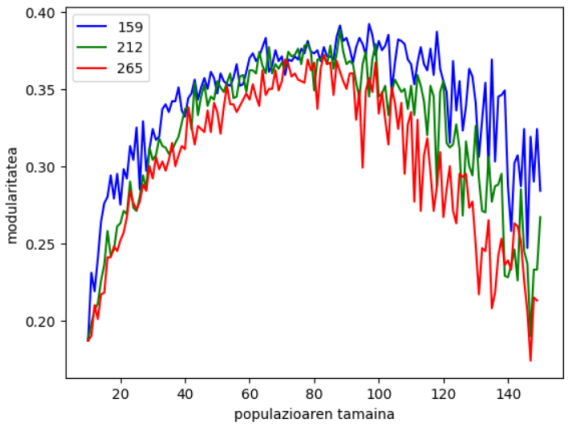

Figure 1: UMDA algoritmoaren modularitateak populazio tamainarekiko instantzia ezberdinekin

1. irudian ikus dezakegu hiru instantzietan populazio tamainaren aldaketek eragin oso antzekoa dutela. Hiru funtzioetan pop_tam = 80 eta pop_tam = 90 artean daude modularitate balio maximoak. Hori dela eta, instantzia guztietarako pop_tam = 85 balio hoberenetarikoa dela ondorioztatu genuen.

Suberaketa simulatuaren parametroak hautatzeko hainbat mapa termiko sortu genituen, alabaina, lehen begiratuan argi ikusten zen irudiak ez zeudela eskualde oso bereizgarrietan banatuta. Horregatik, hiru instantzietan onargarria den, agian ez onena, balio bat aukeratu genien parametro bakoitzari: atalasea=10 eta hasierakoBeroa=1. Hirugarren parametroari hoztenDa = 0 balioa esleitu genion, algoritmoak 5000 ebaluazio egingo zituela bermatzeko hurrengo azterketa guztietan. Mapa termiko horiek hemen eskuragarri daude.

4.3 Hipotesiak

(1) UMDA, suberaketa simulatua baino helburu funtzioaren balio hobeak ematen ditu CDP instantzia originalean, 5000 ebaluazio maximorekin.
(2) k (komunitate kopurua) handitu ahala, suberaketa simulatua txarragoa bilakatzen da UMDA baino eta UMDA eraikitzailea baino txarragoa bilakatzen da, 5000 ebaluazio maximorekin.
(3) Instantzien tamainia handitu ahala, helburu funtzio balio txarragoak lortuko dituzte algoritmo guztiek, 5000 ebaluazio maximorekin.

4.4 1. Hipotesia

Bi algoritmoak probatu genituen bost komunitate desberdinetan (K=5) autoreak banatzeko. Problema osoaren tamaina 1843 nodokoa da nahiz eta grafoa aurreprozesatu eta nodo batzuk baztertu, tamaina handia izaten jarraitzen zuen. Konplexutasun arazoengatik 20 exekuzio egitea lortu genuen. Lortutako emaitzak oso berdintsuak dira 2.irudia, beraz ezin dugu zihurtatu zein algoritmo lortzen dituen emaitza hobeak.

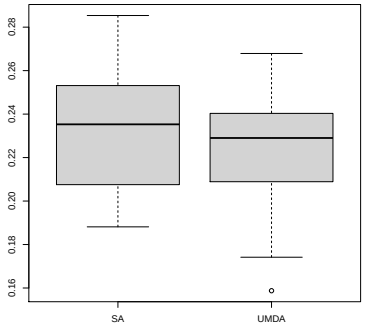

Figure 2: Simulated Annealing eta UMDA algaritmoen exekuzioak.

4.5 2. Hipotesia

Bigarren hipotesia lantzeko aipatutako hiru instantziak erabili genituen hasiera batean. Hiru instantzia hauetako bakoitzean suberaketa simulatua, UMDA eta algoritmo eraikitzailea exekutatu genituen k partizio-kopuru ezberdinetarako, aldi bakoitzean 25 aldiz. Eskatu genizkien partizio-kopuruak instantzia bakoitzeko k=3tik k=45era bitartekoak izan ziren, hiruko pausoarekin (k=3,k=6,k=9, etab.).

Suberaketa simulatua eta UMDA algoritmoen gure inplementazioak oinarrizko desberdintasun bat zuen: soluzio bideragarrien bermea. Suberaketa simulatuak gaitasuna zuen soluzio bidergarriak bermatzeko (2 kide edo gehiagoko komunitateak), aldiz, UMDAk ez. Hori horrela, 159 nodoko instantzian k=18tik aurrera eta 212 nodoko instantzian k=27tik aurrera UMDA soluzio ez bidergarriak itzultzen hasten da, k areagotu ahala gero eta gehiago. Algoritmo eraikitzaileak ere bideragarritasun bermea du. Hipotesiaren frogapenerako ohar hau garrantzitsua da.

UMDAk k handiagoekin soluzio bidergarriak eman zitzan bi instantzia berri sortu ditugu: 251 eta 263 nodokoak. 251 nodoko instantziarekin egin ditugu exekuzioak 159koarekin egin beharrean k=18tik aurrera eta 263koarekin k=27tik aurrera, 212 nodokoaren ordez. 265 nodoko instantzia beti erabili dugu. Aipatu behar k berean algoritmo guztiekin instantzia berdinak erabili ditugula beti.

4.6 3. Hipotesia

Hirugarren hipotesia egiaztatzeko, algoritmo bakoitza instantzia bakoitzean 50 aldiz exekutatu genuen. 6. eta 7. Irudietan lortutako emaitzak daude.

Suberaketa simulatua ezin du aldaketarik (7. Irudia). Eraikitzailea (irudi hau) instantzia handitu ahala ez du aldaketa askorik helburu funtzioan. UMDA-n (6. Irudia) horrela izatea izan liteke, baina teilakatze handiegia dago instantzia tamaina ezberdinen modularitateen artean hori ondorioztatzeko.

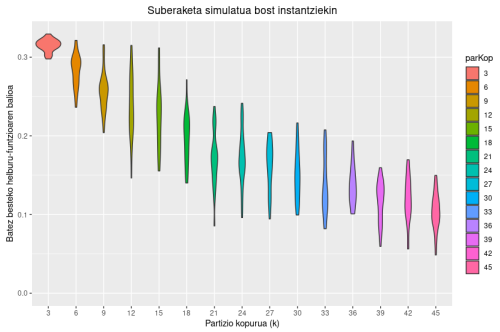

Figure 3: Suberaketa simulatua algoritmoaren portaera k ezberdinekin.

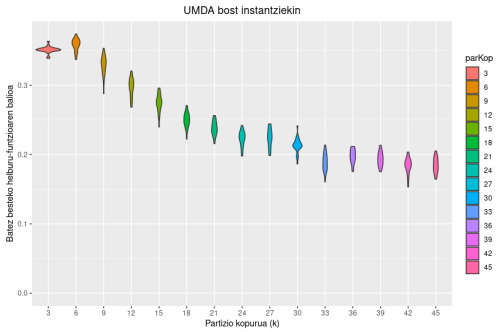

Figure 4: UMDA algoritmoaren portaera k ezberdinekin.

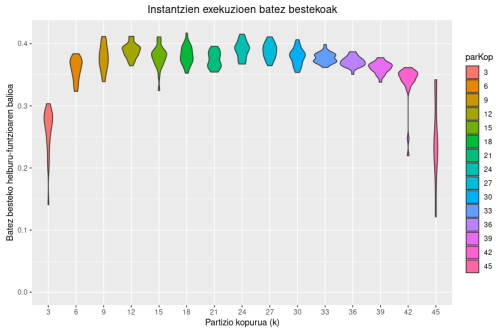

Figure 5: Algoritmo eraikitzailearen portaera k ezberdinekin.

4.7 Analisi estatistikoa

3. eta 4. irudietan ikusi daiteke UMDAk k partizio-tamaina guztietan emaitzen mediana hobea eta bariantza nabarmen txikiagokoa duela suberaketa simulatuarekin alderatuta. 5. irudian aldiz ikusi daiteke algoritmo eraikitzaileak UMDAren bariantza antzeko soluzioak itzultzen dituela, baina k handitu ahala haien egokitasuna ez dela gutxitzen, UMDArenekin ez bezala. Aipatu beharra dago k=42tik aurrera erabilitako instantzietan soluzioak sortzeko zailtasuna duela, hasierako puntu asko hautatu behar dituelako (ikus 3.1 atala). Instantzia handiagoekin arazo hau gainditu litzateke, baina kontuan hartzekoa da.

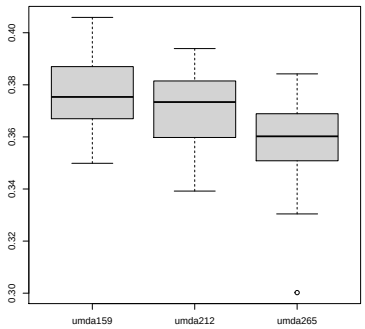

Figure 6: UMDA modularitateak instantzia tamainarekiko instantzia ezberdinekin

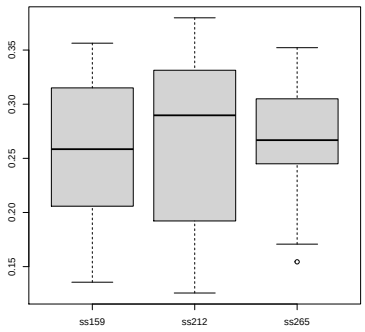

Figure 7: Suberaketa simulatua modularitateak instantzia tamainarekiko instantzia ezberdinekin

5 ONDORIOAK

Artikulu honetan CDP problemaren ezaugarriak aztertu ditugu egokitu zaigun probleman oinarrituta. Problemari aurre egiteko algoritmo ugari landu ditugu eta azaldutako hiru horiekin lortu ditugu emaitza onenak. Suberaketa simulatuak partizio kopurua handituz gero UMDAk baino bariantza handiagoko eta helburu-funtzioaren balio baxuagoko emaitzak itzultzen dituela ondorioztatzea lortu dugu eta honek algoritmo eraikitzaileak baino okerragoak zentzu horretan, instantzien tamainek soluzio bideragarriak sortzeko besteko handitasuna badute. Gainera, suberaketa simulatua instantzien tamainarekiko eskalablea dela ondorioztatu dugu.

REFERENCES

[1] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6 (dec 2004). https://doi.org/10.1103/physreve.70.066111

[2] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[3] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

[4] Pedro Larrañaga and Jose A Lozano. 2001. *Estimation of distribution algorithms: A new tool for evolutionary computation.* Vol. 2. Springer Science & Business Media.