Algoritmo Genetikoa eta Simulated Annealing Community Detection Problem-a ebazteko

Xuban Barberena
Euskal Herriko Unibertsitatea UPV/EHU
xbarberena001@ikasle.ehu.eus

ABSTRACT

Azken urteetan, enpresa eta ikerlariek komunitateak grafoetan detektatzea oso erabilgarria izan daitekela ohartu dira. Baina, NP-Hard problema bat denez, oraindik ez da soluzio egokirik lortu. Urteetan zehar hainbat metodo desberdin planteatu dira, Artikulu honetan, optimizazio metaheuristikoko bi algoritmo proposatu dira: Algoritmo Genetikoa eta Simulated Annealing. Honela, inplementazioaren eta experimentazioaren diseinu azaldu da. Azkenik, algoritmoekin konparaketa egin da eta Random Search baseline algoritmoak baino emaitza hobeak lortzen dituztela frogatu da.

1 SARRERA

Gaur egungo aro digital honetan informazio sisetmak sare konplexuen bidez adierazten dira. Sare hauen ezaugarrietako bat komunitateak defini daitezkela da. Grafoei dagokienez, komunitateak elkarri trinkoki loturik dauden nodoen azpimultzo bezala definitu daitezke, grafo berdineko beste komunitateetako nodoekin lotura gutxi izanik.

Sare konplexu hauetan komunitate banaketa aurkitzea aplikazio asko ditu mundu errealean, hala nola, gomendatze-sistema, sare sozialen analisia, politikan etb. Azken urteetan, enpresa eta ikerlari gehio komunitate detekzioaren erabileratarataz ohartu dira, gai honi buruzko ikerketak areagotuz.

Baina, komunitate detekzioa NP-Hard problema bat da. Beraz, ez da denbora efizientean ebazteko gai den algoritmorik lortu. Hainbat metodo proposatu dira [1], hala nola, clustering algoritmo desberdinetan oinarrituak, greedy teknikak etab.

Problema konplexuak ebazteko aukera nagusienetako bat hurbilketa algoritmoen erabilera da. Hauen abantaila nagusia arrazoizko denbora batean soluzio optimo edo ia optimoa eman dezaketela da. Beraz, algoritmo metaheuristikoen erabilera baliagarria izan daiteke gure probleman. Artikulu honetan, optimizazio metaheuristikoko bi algoritmo proposatu dira: Algoritmo Genetikoa eta Simulated Annealing.

2 COMMUNITY DETECTION PROBLEM

Lehenik eta behin, ebatzi nahi den problema azalduko da. Artikulu honen helburua Neural Information Processing Systems (NIPS) kongresuan publikatzen duten autoreen komunitateak aztertzea da. Komunitateak hainbat elkarlana egin duten autoreek osatzen dute.

Problema hau grafo baten bidez adierazi daiteke, zeinetan autoreak nodo bezela adierazten diren eta hauen arteko elkarlana ertz bezela (grafo ponderatua, bi nodoen arteko elkarlan kopurua adierazizten da). Honela, soluzioaren kodeaketa grafoko nodo bakoitza errepresentatzen duen n tamaineko lista bat da, zeinetan nodo bakoitza ze komunitatekoa den adierazten den. Kasu honetan, soluzio espazioa eta bilaketa espazioaren berdinak dira: {1,...,k}^{n}, non n nodo kopurua eta k komunitate kopurua diren.

Metaheuristikoekin lorturiko emaitza posible bakoitza ebaluatu beharra da ea ze ona den jakiteko. Ondorioz, helburu-funtzio bat definitu beharra da. Horretarako grafoen propietate bat erabili da: modularitatea [2]. Modularitateak komunitate banaketa bat zein ona den adierazten du komunitate horren barruko konexioen dentsitatea neurtuz. Modularitate puntuazio altua duten grafoek lotura asko izango dute komunitate berdineko erpin-en artean eta gutxi gainontzekoekin. Beraz, soluzio hoberena lortzeko maximizatzea da helburua.

$$Q = \sum_i (e_{ii} - a_i)^2 \tag{1}$$

non, e_{ij}: i komunitateko eta j komunitateko erpinak lotzen dituzten ertzen frakzioa den:

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

eta, a_i: i komunitatearekin loturiko ertzen frakzioa den.

$$a_i = \frac{1}{2m} \sum_v k_v \delta(c_v, i) \tag{3}$$

Algoritmoa sakonago aztertzeko irakurri [2] artikulua.

3 ALGORITMOEN DISEINUA

Aurretik aipatu den bezela, artikulu honetan optimizazio metaheuristikoko bi algoritmo landu dira: Algoritmo Genetikoa eta Simulated Annealing. Bi algoritmoak estokastikoak dira, hasierako soluzioak ausazkoak izanik.

3.1 Algoritmo Genetikoa

Algoritmo genetikoak bilaketa- eta optimizazio-problemak ebazteko erabiltzen dira, populazioak problemaren balio hoberenetarantz eboluzionatzean oinarritzen baitira. Beraz, gure problemarentzat egokia da. Eraikitako algoritmoaren egitura edo fluxua hurrengoa da [3]:

1- Hasieraketa Operadorea: hasierako populazioa sortu. Hainbat aukera daude, hala nola, ausaz, banaketa probabilistiko bat jarraituz, algoritmo heuristiko eraikitzaile baten bitartez etab. Kasu honetan, ausaz sortu da dibertsifikazioa mantentzeko egokia delako eta inplementatzeko erraza.

2- Soluzioak ebaluatu: populazio bakoitzeko soluzio bakoitza ebaluatu nahi da ea zein onak diren jakiteko. Horretarako, aurretik azaldu den modularitate helburu-funtzioa erabiltzen da. Hau izango da gelditze irizpidea, ebaluazio maximo batera iristean amaitu.

3- Aukeraketa operadorea: uneko populaziotik indibiduo batzuk aukeratu ondoren birkonbinaketa aplikatzeko. Irizpide desberdinak daude: Truncation Selection (ingurune bateko k hoberenak hartu),

ACM Conference, , July 2017, Washington, DC, USA. 2023.

Rank Selection etab. Kasu honetan, Truncation Selection erabili da problema handi honetan inplementazio merkea nahiago delako (baina dibertsitatea azkar murrizten du).

4- Birkonbinaketa operadorea: aukeratu diren soluzioekin indibiduo berriak sortu gurasoen propietateak (gene honak). Modu desberdinetan egin daiteke, baina One-Point Crossover inplementatu da. Konputazionalki merkea da eta problema honetan soluzio bideragarriak sortzen ditu, beraz, erabiltzeko egokia.

5- Mutazio operadorea: populazioko indibiduo batzuetan ausazko aldaketa txikiak aplikatzea dibertsifikazioa lortzeko. Hainbat metodo, baina kasu honetan Bit-Mutation erabili da: gene bati ausazko beste balio bat eman bideragarritasuna mantenduz. Inplementazio merkea eta debertsifikazioa handitzen duenez aukera egokia da.

6- Eguneraketa operadora: populazio berritik eta zaharretik belaunali berria sortu. Aukera ohikoena inplementatu da: bi populazioak batera jarri, ordenatu eta m soluzio onenak aukeratu.

7- Gelditze irizpidea bete ez bada, itzuli 2. puntura.

Algorithm 1 Genetic Algorithm

Input: evaluate, select, update, reproduce 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^n = reproduce(P_t^s)
    P_{t+1} = update(P_t, P_n)
    $t = t + 1$
end while
s^* = Best solution of the population

3.2 Simulated Annealing

Simulated Annealing bilaketa globala optimizatzeko algoritmo estokastikoa da, ausazkotasuna erabiltzen du bilaketa prozesuaren zati gisa [4]. Hori dela eta, algoritmoa egokia da helburu funtzio ezlinealetarako, non beste bilaketa lokaleko algoritmo batzuek ez dute ongi funtzionatzen. Hill Climb bilaketa lokaleko algoritmoa bezela, soluzio bateko ingurunea aztertzen du optimo lokalera iritsi arte. Baina, Hill Climbining ez bezela, Simulated Annealing-ek momentuko soluzioa baino txarragoak onartu ditzazke optimo lokaletatik ihes egiteko helburuarekin.

Hainbat irizpide hartu beharra dira kontuan:

- Hasierako Tenperatura: oso garrantzitsua hasierako tenperatura ongi erabakitzea. Hurrengo formula erabili da:

$$T_0 = \frac{-\Delta E}{ln(p)} = \frac{-(f_u - f_l)}{ln(p)} \tag{4}$$

non f_u helburu-funtzioak hartu dezaken balio maximoa da eta f_l balio minimoa. Erabilitako modularitate helburu funtzioan $f_u = 1$ eta $f_l = -0.5$ dira. Baina, problema honetan soluzioen harteko diferentzia oso txikia denez, balio maximo eta minimo hauek erabiltzeak hasierako tenperatura handiegia ematen du.

Muturreko balio hauek hartu beharrean, beste modu batean f_u eta f_l lortzea pentsatu da: helburu funtzioa askotan ebaluatu da ausazko soluzioekin, balio maximoa eta minimoa aukeratuz.

- Oreka: tenperatura eguneratzen den bakoitzean hainbat iterazio egin nahi dira tenperatura berdinarekin. Balio estatiko bat erabili da iterazio guztietarako: $\rho \left|N(s)\right|$, *non* $\rho \in [0, 1]$. Aurrerago erakutsiko da ze balio hartu den.

- Temperatura Eguneraketa: iterazioak pasa ahala soluzio txarrak gutxiagotan aukeratu nahi dira. Hiru aukera nagusi tenperatura gutxizteko: linealki, geometrikoki edo logaritmikoki. Kasu onetan, lineala erabili da inplementatzeko erraza delako eta emaitza honak ematen dituelako. β lortzeko ondorengo formula erabili da:

$$\beta = t_0 \frac{oreka}{evals} \tag{5}$$

eta tenperatura linealki honela eguneratzen da: $T_i = T_{i-1} - \beta$

- Ingurune funtzioa: Swap edo hamming erabiltzeko egokiak izan daitezke. Kasu honetan, hamming erabiltzea erabaki da nodo bati beste komunitate batean jartzea emaitza hobeak ematen dituela pentxatzen delako.

Algorithm 2 Simulated Annealing

Input: random_neighbor(), update_temperature(), equilibrium(), stop_condition()
Output: s^* solution
$s^* = s_0$
$T = T_0$
while !$stop\_criterion()$ **do**
    while !$equilibrium()$ **do**
        s' = random_neighbor(s)
        $\Delta E = f(s') - f(s)$
        if $\Delta E < 0$ **then**
            $s = s'$
            if $f(s) - f(s^*)$ **then**
                $s^* = s$
            end if
        else
            $s = s'$ with $e^{-\frac{\Delta E}{T}}$ probability
        end if
    end while
    T =update_temperature(T)
end while
s^* = Best solution

4 ESPERIMENTAZIOA

Inplementatutako algoritmoekin esperimentazioa egin aurretik honen diseinua azalduko da. Ondoren, bi hiposesi proposatu dira eta esperimentazioa hauen gain egin da. Amaitzeko, emaitzak aztertuko dira Random Search algortimoa baseline bezela erabilita eta amaierako ondoriak atera dira.

4.1 Esperimentazioaren diseinua

Aurretik aipatu den bezela, gure problemarako datu base bakarra dugu zeinetik grafo bakarra sortzen den. Baina, esperimentazioa egiteko egokiagoa da hainbat problema desberdinetan probatzea. Orduan, ausazko instantzia generaodre bat (Random instance generator) erabili da datu base bakarretik hainbat grafo desberdin

sortzeko. Grafoaren tamaina aldakorra izan daiteke eta ez da osotara ausaz sortuko. Hau da, banaketa probabilistiko bat erabili da lotura gehien duten nodoei pisu gehio emanez. Amaieran lortuko diren emaitza eta ondorioak generadoreak jarraitzen duen banaketa probabilistikoaren honen gainera hedatuko dira.

Ondoren, parametroen egokitzapena egin da (parameter tuning). Aurretik ikusi den bezela, algoritmoak hainbat parametroeren menpe daude eta parametro hauen balioak algoritmoen portaera baldintzatzen dute. Beraz, garrantzitsua ahalik eta balio hoberenak aukeratzea. Honetarako erabili da parameter tuning.

Konputazionalki baliabide gutxi dugunez, algoritmo bakoitzeko bi parametro hartu dira Grid Search aplikatzeko (ahalik eta garrantzitsuenak eta independienteak izatea saiatu da). Parametro bakoitzeko hiru balio hartu dira eta aukera bakoitzeko 5 aldiz errepikatu da exekuzioa (media kalkulatzen da kasu bakoitzerako). Honela, aukera guztiak begiratuko dira balio hoberena lortzeko helburuarekin. Gainerako parametroak finkoak dira ahalik eta balio egokiena aukeratuz bakoitzarentzako. Generadorea erabiltzen dugunez, kasu bakoitzeko instantzia berri bat sortu da exekuziorako. Grafoaren tamaina 600 erpinekoa izan da kasu guztietan.

Bestalde, gelditze irizpidea beti berdina izan da bi algoritmoetan: instantzia bakoitzean-helburu funtzioa 5000 aldiz ebaluatzean amaitu. Azkenik, esperimentazio guztia makinan lokalki exekutatu da Jupyter Notebook ingurunean.

Ondoren, parameter tuning-ek algoritmo bakoitzean eman dituen emaitzak bistaratuko dira:

Algoritmo Genetikoaren kasuan aukeraketa eta populazio tamaina aukeratu dira parametroen egokitzapenerako. Zehazki, populazio tamainarako $\{40, 60, 80\}$ balioak hartu dira eta aukeraketa tamainarako $\{20, 30, 40\}$. Parametro finkoen balioak hurrengoak izan dira: 20 komunitate, off_size 40 eta mutazio probabilitatea 0.02. Lortutako heatmap grafika hurrengoa da:

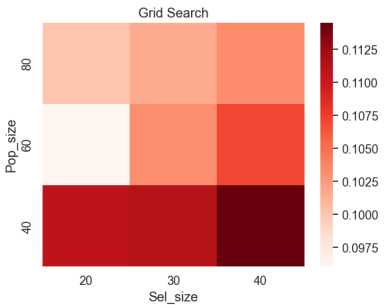

Figure 1: Parameter Tuning Heatmap Genetic Algorithm

1. irudian ikusi daiteken bezela, populazio tamaina 40 denean ematen ditu emaitza hoberenak. Aukeraketa tamainaren kasuan, berriz, hiru kasuetan antzeko ematen ditu (sel_size=40 denean hoberen). Errepikapen gehiagorekin exekutatuta emaitza hobeak lortuko lirateke.

Bestalde, Simulated Annealing-en kasuan, hasierako tenpertura eta oreka baldintzaten duen p probabilitatea aukeratu dira parametroen egokitzapenerako. Zehazki, hasierako tenpertura $\{0.2, 0.4, 0.6\}$ balioak hartu dira eta orekaren p probabilitatea $\{0.1, 0.3, 0.5\}$. Lortutako heatmap grafika hurrengoa da:

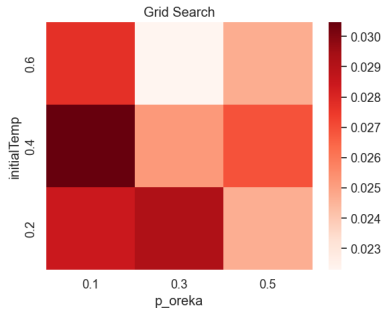

Figure 2: Parameter Tuning Heatmap Simulated Annealing

2. irudian ikusi daiteke oreka baldintzatzen duen probabilitatea txikia denean emaitza hobeak lortzen direla. Bestalde, hasierako tenperatura 0.4 inguruan denean emaitza hobeak lortzen dira. Balio honetatik hurruntzean balioak okertu egiten dira. Honek aurretik hasierako tenperaturaren hurbilketa lortzeko teknikak ongi funtzionatzen duela adierazten du (0.4 inguruko balioa ematen du).

Dagoeneko bi algoritmoentzat balio egokiak ezarri dira. Hurrengo pausua algoritmoen errendimendua begiratzea da. Horretarako, bi hipotesi poposatu dira eta hauen analisia egin da.

4.2 A hipotesia

Hipotesia

Algoritmo Genetikoak eta Simulated Annealing-ek emaitza hobeak lortzen ditu Random Search-ek baino instantzien tamaina desberdinetarako.

Esperimentuaren deskribapena

Esperimentu honetan hiru algoritmoak $\{50, 100, 600, 1000, 1500\}$ tamaineko instantziekin exekutatu dira. Kasu bakoitzeko 6 aldiz errepikatu da eta aldiro generadorearekin instantzia berri bat sortu da. Gelditze irizpidea 10^4 ebaluazio izan da algoritmoen exekuzio bakoitzeko eta komunitate tamaina finko mantendu da 20 balioan.

Emaitza

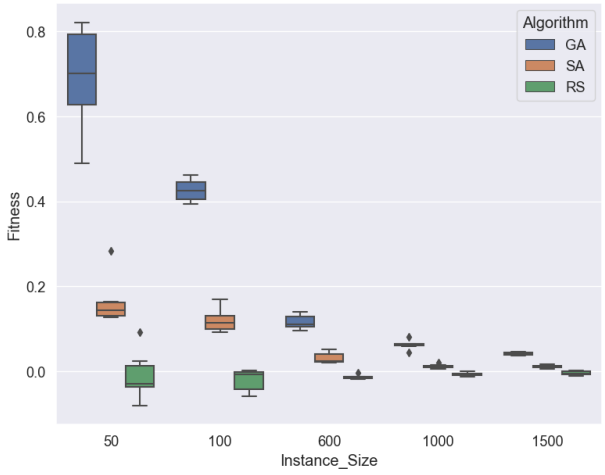

Figure 3: Hiru algoritmoen konparaketa instantzien tamainaren arabera boxplot erabiliz

Ondorioa

3. irudian garbi ikusi daiteke Algoritmo Genetikoak emaitza hoberenak lortzen dituela. Instantzia tamaina txikia denean diferentzia handia dago, baina GA-k ematen dituen balioak aldakorragoak dira besteak baino (errepikapen gutxi egitearen arazoa izan daiteke). Bestalde, instantzia tamaina handitu ahala, GA eta SA-ren emaitzak okertu egiten dira eta balioen aldakortasuna txikitu egiten da. Nolanahi ere, GA eta SA Random Search-ek baina emaitza hobeak lortzen dute kasu guztietan. Beraz, hipotesia bete dela zihurtatzen da.

4.3 B hipotesia

Hipotesia

Algoritmo Genetikoak eta Simulated Annealing-ek emaitza hobeak lortzen ditu Random Search-ek baino komunitate tamaina desberdinetarako.

Esperimentuaren deskribapena

Esperimentu honetan hiru algoritmoak {5, 10, 20, 50, 100} tamaineko komunitateekin exekutatu dira. Kasu bakoitzeko 6 aldiz errepikatu da eta aldiro generadorearekin instantzia berri bat sortu da. Gelditze irizpidea 10^4 ebaluazio izan da algoritmoen exekuzio bakoitzeko. Instantzia guztien tamaina 600 erpinekoa izan da.

Emaitza

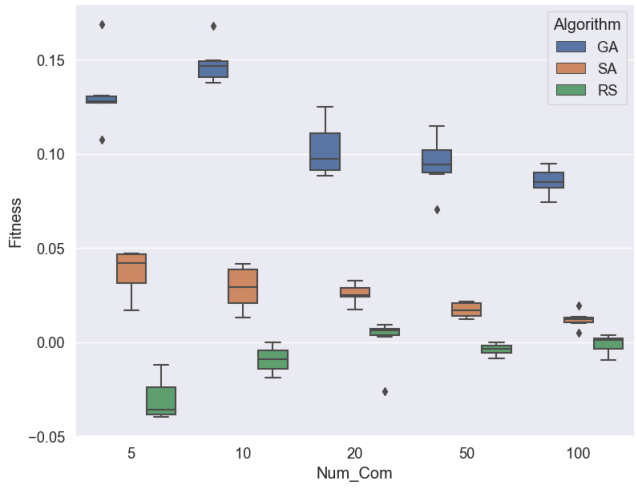

Figure 4: 3 algoritmoen konparaketa komunitate kopuruaren arabera boxplot erabiliz

Ondorioa

4. irudia garbi ikusten da Algoritmo Genetikoak emaitza hoberenak lortzen dituela. Komunitate kopurua handitu ahala GA eta SA-ren emaitzak pixka bat jeitsi egiten dira. Nolanahi ere, GA eta SA Random Search-ek baina emaitza hobeak lortzen dute kasu guztietan. Beraz, hipotesia bete dela zihurtatzen da.

5 ONDORIOAK ETA ETORKIZUNEKO HOBEKUNTZAK

Artikulu honek Community Detection Problem-a ebazteko bi algoritmo metaheuristiko proposatu ditu: Algoritmo Genetikoa eta Simulated Annealing. Artikulu honetan inplementazioan eta experimentazioaren diseinuan sakondu da. Experimentalki frogatu da bi algortimoek emaitza hobeak lortzen dutela Random Search-ek baino problemaren instantzia eta komunitate kopuru guztietarako. Gainera, hipotesi berdinetarako GA-k problema honetarako SA baino hobe funtzionatzen duela ere ikusi da.

Baina, naiz eta emaitzak baseline baino hobeak izan, state-of-the-art metaheuristikoekin alderatuta oso atzean daude. Hala ere, hemen proposatutako algoritmoek hobetzeko marjina handia dute. Alde batetik, ahalmen konputazionala oso eskasa izan denez, etorkizunerako hobekuntza bezela kodea paralelizatzeak eta makina lokalean ez exekutatzeak eraginkortasunean lagunduko luke. Honela, algoritmoak 10^5 edo 10^6 gelditze baldintzarekin probatu daiteke eta bakoitzeko gutxienez 10 errepikapen. Era berean, ahalmen konputazionala handituz parametroen egokitzapena hobea egin daiteke. Guzti honek emaitza hobeak emango ditu.

Beste aldetik, etorkizuneko hobekuntza bat analisi estadistikoa egitea da. Honek, algoritmoen konparaziorako eta portaera hobe ulertzen lagunduko luke.

REFERENCES

[1] B. S. Khan and M. A. Niazi., "Network community detection: A review and visual survey," 2017. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1708/1708.00977.pdf

[2] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, no. 70, 066111, 2004. [Online]. Available: https://arxiv.org/pdf/cond-mat/0408187.pdf

[3] M. Tasgin, A. Herdagdelen, and H. Bingol, "Community detection in complex networks using genetic algorithm," *Corrosion Science*, 2007. [Online]. Available: https://arxiv.org/ftp/cond-mat/papers/0604/0604419.pdf

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *American Association for the Advancement of Science*, 1983. [Online]. Available: http://www2.stat.duke.edu/~scs/Courses/Stat376/Papers/TemperAnneal/KirkpatrickAnnealScience1983.pdf