CDP proiektuaren txostena

Bittor Alkain eta Maitane Martínez

***Laburpena*—Komunitate detekzioaren problema hainbat algoritmoren bitartez ebazten saiatu da. Kasu honetan Neural Infomartion Processing Systems kongresuan publikatutako autoreak komunitateetan banatzea da helburua. Horretarako bi algoritmo erabili dira: suberaketa simulatua eta algoritmo genetikoa. Lehenengoa soluzio bakarrean oinarrituta dago eta bigarrena, aldiz, populazioan. Bi algoritmoak egokitu dira problemara eta emaitzak konparatu dira.**

I. Sarrera

Gaur egun zientziarentzat interesgarriak diren sistema asko grafo bezala adierazi daitezke, modu erabilgarrian. Horien artean daude Internet, web-orriak, sare sozialak, sare biokimikoak, erakunde handiak, eta abar. Grafo horietako bakoitzak erpin multzo bat du, Interneteko ordenagailuak edo sare sozial bateko pertsonak adierazi ditzakeena, eta elkarren artean lotzen dituzten ertzak, ordenagailuen arteko konexioak edo pertsonen arteko adiskidetasunak adierazteko. Azken urteetako lanetan, grafoen ezaugarri batek garrantzia handia hartu du: komunitate-egiturak. Komunitatearen definizio orokor bezala, esan daiteke erpin multzo bat dela, non bertako erpinen arteko ertzen dentsitatea handiagoa den beste komunitateetarako ertzena baino. [1] [2]

Bilaketa Heuristikoak irakasgaiko proiekturako, komunitate-detekzioaren problema ebatzi dezaketen bi algoritmo inplementatzea eskatu zaigu, metaheuristikoak erabilita. Algoritmo batek soluzio bakarrean oinarritutakoa izan behar zuen, eta besteak populazioan oinarritutakoa. Gainera, elementu probabilistikoak erabili beharra zegoen, algoritmo batean gutxienez.

Soluzio bakarrekoa inplementatzeko suberaketa simulatua aukeratu dugu. Poblazionalerako, berriz, UMDA bat erabili nahi genuen, baina ez dugu ondo funtzionatzea lortu. Hori dela eta, beste bide batetik jo dugu: algoritmo genetiko bat idatzi dugu, gurutzatze bidezko ugalketarekin.

Txostenaren gainontzeko atalek honako egitura jarraitzen dute: 2. atalean optimizazio-problema azaltzen dugu, xehetasunak emanez erabilitako datuei, soluzioen kodeketari eta helburu-funtzioari buruz. 3. atalean suberaketa simulatua azaltzen da, eta 4.ean algoritmo genetikoa. 5. atalean egindako esperimentazioari buruz hitz egiten da, parametro-bilaketaren eta algoritmoen arteko konparazioaren emaitzak azalduz. Bukatzeko, 6. atalean ondorio orokorrak daude.

II. Optimizazio-problema

1) Datuak: Proiektu honetarako esperimentazio guztia instantzia bakarrarekin egin dugu: zehazki, Neural Information Processing Systems (NIPS) kongresuan publikatzen duten autoreen komunitateekin. Guztietatik, 2014 eta 2015 urteen bitarteko lanak aztertuko ditugu eta hauetan eman diren elkarlan komunitate desberdinak aurkituko ditugu.

Horretarako NIPSeko elkarlan-grafoa eraiki dugu eta eraikitako algoritmoek komunitate bakoitzean dauden autoreak zeintzuk diren esan beharko digute. Autore bakoitza komunitate baten kide izango da eta komunitate kopuru maximoa lehendik ezarriko da.

Elkarlan-grafoan, autore bakoitza erpin bat izango da. Bi autorek elkarrekin lan egin badute, haien erpinen artean ertz bat egongo da. Beraz, grafoa ez zuzendua izango da, eta ertzak ezingo dira erpin berean hasi eta bukatu. Gainera, ertzek pisua daukate. Izan ere, ez da berdina bi autoreren artean lan bakarra edo gehiago izatea. Beraz, ertzen pisuak autoreen arteko artikulu kopuruari erreferentzia egingo dio.

2) Soluzioen kodeketa: Soluzio bakoitza adierazteko zenbaki-lista bat erabiliko dugu. Erpin bakoitzeko zenbaki bat izango dugu, eta zenbaki horrek erpinari dagokion komunitatea identifikatuko du.

3) Helburu-funtzioa: Autore bat komunitate baten parte izango da artikulu asko idatzi baditu komunitatekoekin, eta ez hainbeste kanpokoekin. *Modularitatearen* [1] bidez neurtuko dugu zer neurritan betetzen den grafo osorako. Kontuan hartu behar da zenbatetan idatzi dituen artikuluak beste autore batzuekin. Izan ere, ez da nahikoa behin artikulu bat idaztea komunitatearen parte izateko.

A_{vw} sarearen auzokidetasun-matrizea definituko dugu:

$$A_{vw} = \begin{cases} pisua & v \text{ eta } w \text{ erpinak konektatuta badaude} \\ 0 & \text{bestela} \end{cases}$$

non pisua bi erpinen arteko artikulu kopurua den. Gainera, badakigu erpinak komunitatetan banatuta daudela, beraz, esan dezakegu v erpinaren komunitatea c_v dela. Elkarlan guztietatik, komunitate berean dauden autoreen arteko elkarlanen proportzioa hau 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 $\delta(i,j) = 1$ den baldin $i = j$ bestela 0, eta $m = \frac{1}{2}\sum_{vw} A_{vw}$ grafikoko ertzen pisuen batura den. Formula hau egokia izango da sarearen zatiketa onetarako, baina ez da komunitatearen egitura neurri ona erpin guztiak komunitate berekoak direnean, 1eko balioa hartuko duelako. Dena den, ausazko sare baten kasuan espero den balioa kenduz gero neurri erabilgarria lortuko dugu.

Erpin baten gradua haren ertzen pisuen batura izango da:

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

v eta w erpinen artean espero daitekeen pisua $k_v k_w/2m$ bidez kalkulatzen da. Beraz, Q modularitatea 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 elkarlan kopurua ausazko grafo batean espero dugunaren berdina bada, modularitatea 0 izango da. Zero ez diren balioek ausazkotik zenbat aldentzen den adierazten dute. Balio hori maximizatu egin nahi dugu.

Jarraian dagoen sasikodeak modularitatea kalkulatzeko prozesua laburbiltzen du:

```
modul = 0
m = pisuen_batura()
for u, v in konbinazio_guztiak(G) do
    if komunitate_berekoak(u, v) then
        A = pisua_lortu(G, u, v)
        ku = gradua(G, u)
        kv = gradua(G, u)
        modul+ = A − (ku ∗ kv/2m)
modul = modul/2m
```

III. Suberaketa simulatua

Suberaketa simulatua (*simulated annealing*, SA) soluzio bakarrean oinarritutako optimizazio-metodoa da. Bilaketa lokalean bezala, bilaketan zehar beti uneko soluzio bat dago, eta soluzioaren auzokideak ebaluatuz eta horietara mugituz egiten du aurrera algoritmoak. Baina, bilaketa lokal estandarrean ez bezala, batzuetan unekoa baino soluzio okerrago batera mugitzea onartuko da.

Lehen aldiz Scott Kirkpatrick, C. Daniel Gelatt eta Mario P. Vecchi-k [3] proposatu zuten 1983an, eta independenteki Vlado Černý-k [4] 1985ean.

Algoritmoaren egitura orokorra honakoa da:

```
s = sortu_hasierako_soluzioa()
T = hasierako_tenperatura()
while not bukaera_baldintza() do
    while not oreka() do
        s′ = ausazko_auzokidea(s)
        ΔE = f(s′) − f(s)
        if ΔE < 0 then
            s = s′
        else
            p = exp(−ΔE/T)
            if p > ausaz([0, 1]) then
                s = s′
    T = eguneratu(T)
```

Algoritmorako inspirazioa suberaketa da, industrian hainbat materialekin (beira, altzairua, etab.) erabiltzen den teknika: materiala lehenik berotu egiten da, eta gero, pixkanaka hoztu. Horrela, bere propietate fisikoak aldatzen dira. Izan ere, tenperatura jaisten den bakoitzean, partikulak energia baxuagoko egoeretan berrantolatzen dira, oreka termikora iritsi arte. Pixkanaka, kristal-egitura handiak sortzen dira. [5]

Bilaketa lokaleko aldaketak konparatu daitezke materialetako berrantolaketa mikroskopikoekin. Horrela, kostu-funtzioak energiaren papera beteko luke. Soluzio hobeak bakarrik onartzea tenperatura azkarregi jaistearen parekoa litzateke, emaitza okerragoak sortuz: bilaketaren kasuan, optimo lokal batean gelditzea. [3] Beraz, algoritmo honetan energia handiagoko soluzio batzuk onartuko dira, onartzeko $p = e^{\frac{-\Delta E}{T}}$ probabilitatea ΔE energia-aldaketak eta T tenperaturak baldintzatzen dutelarik.

Ikusten denez, hainbat irizpide finkatu beharra dago: nola sortu hasierako soluzioa, nola lortu hasierako tenperatura, zein diren soluzio auzokideak eta kostu-funtzioa, zenbatero eguneratu tenperatura, nola eguneratu eta noiz bukatu. Algoritmoaren egileek diote problema bakoitzerako parametroak *trial and error* bidez bilatu beharko direla. [3] Jarraian zenbait aukera planteatu eta hartutako erabakiak azalduko ditugu.

1) Hasierako soluzioa: Bi hasieraketa probatu ditugu: ausazkoa eta algoritmo eraikitzaile bidezkoa. Ausazkoan, erpin bakoitzerako 0 eta komunitate kopuruaren arteko zenbaki bat ausaz lortuko da. Eraikitzailean, berriz, erpin batetik hasita, grafoa zabaleran korrituz aurkitzen diren erpin guztiei komunitate bera esleitzen zaie, komunitatearen erpin-kopurua bete arte. Ondoren, berdin hurrengo komunitateekin. Komunitate guztiek tamaina bera izango dute, gehienez erpin bateko aldearekin. Erpinen zerrenda ausaz ordenatzen da, osagai konexu bakoitzeko abiapuntua exekuzio bakoitzean desberdina izateko.

2) Auzokideak: Orokorrean, bilaketa lokalean eta bere aldaeretan, auzokide batek uneko soluziotik hurbil egon behar du, hau da, antzekoa izan behar du. Gure kasuan, hurbil dagoela diogu baldin eta erpin bakarra aldatu bada komunitatez. Gainera, beste murriztapen bat ere jarri diogu ingurune-funtzioari: aldaketak grafoan auzokidea den komunitate batera bakarrik egin daitezke. Hau da, komunitateen mugetan dauden erpinak bakarrik aldatu ahal izango dira. Uste dugu horrek bilaketa eraginkorragoa egingo duela, ez baitu zentzu handirik, itxuraz, komunitate bati berarekin lotura zuzenik ez daukan erpin bat gehitzeak.

Algoritmo honek saiakera bakoitzerako auzokidea ausaz aukeratzea eskatzen du. Izan ere, aukerak ordena jakin bat jarraituz aztertuko bagenitu, bilaketaren hasieran ia aldaketa guztiak onartuko ditugunez, behin eta berriro nodo gutxi batzuk aldatzen arituko ginateke.

Idatzitako inplementazioan, lehenik ausaz aldatuko den nodoa aukeratzen da. Ondoren, bere ondoko nodoen artean beste komunitate bateko kiderik baldin badago, auzoko komunitate horien multzoa osatu eta haien artetik bat ausaz aukeratzen da, komunitate guztiek aukeratuak izateko probabilitate bera dutelarik. Aldiz, nodoa ez badago komunitateen arteko mugan, beste bat aukeratzen da.

3) Kostu-funtzioa: Modularitatea maximizatu egin nahi dugunez, azaldutako algoritmoan kostua edo energia ordezkatzeko $energia = -modularitatea$ erabiliko dugu.

4) Hasierako tenperatura: [6] artikuluan honakoa proposatzen dute hasierako tenperatura ezartzeko: hainbat mugimendu probatu, energia handitzen dutenen batez besteko ΔE kalkulatu, p_0 hasierako onarpen-probabilitatea ezarri (ia 1, hasieran ia mugimendu guztiak onartzeko), eta $T_0 = \frac{-\Delta E}{ln(p_0)}$ kalkulatu. ΔE hori lortzeko, hasierako soluziotik abiatuta egin daitezkeen mugimenduetatik batzuk probatzeko kodea idatzi dugu.

5) Tenperatura eguneratzea (hozte-funtzioa): Jatorrizko [3] artikuluan bezala, geometrikoki eguneratu dugu tenperatura $T_k = \alpha T_{k-1}$.

6) Oreka: Suberaketaren analogiari jarraituz, oreka termikora iristean eguneratu beharko litzateke tenperatura. Algoritmoa programatzeko orduan, hainbat irizpide sinple erabili ohi dira: onartutako saiakerak kopuru batera iristean hoztea edo saiakera-kopuru finko bat ezartzea (epoka-luzera). Doitu beharreko parametro kopurua murrizteko, balio finkoa eman diogu epoka-luzerari: $budget/5$. Horrela, tenperatura 4 aldiz eguneratuko da.

7) Bukaera: Guztira ebaluatutako saiakera kopuru maximo batera iristean, exekuzioa gelditu dugu. Horrek pixka bat aldatzen du algoritmo orokorra, bukaera-baldintza saiakera bakoitzaren ondoren egiaztatuko baita.

IV. Algoritmo genetikoa

Algoritmo genetikoak [7] (GA) eboluzio naturalaren teorian oinarrituz sortutako bilaketa metodoak dira. Naturaren teoriaren arabera, populazioek eboluzionatu egiten dute, indartsuenen hautespen naturalaren eta bizi-iraupenerako legeen arabera. Lege honek dio bere ingurunera ondoen egokitu diren kideek aukera handiagoak dituztela seme-alabak izan eta haien ezaugarriak hurrengo belaunaldietako kideei zabaltzeko.

Algoritmo genetikoak ezagunak egin ziren 1970eko hamarkadaren hasieran John Henry Hollanden lanaren bidez, eta bereziki Adaptation in Natural and Artificial Systems (1975) liburuan [15]. GAko ikerketak teorikoak izan ziren neurri handi batean 1980ko hamarkadaren erdialdera arte.

Algoritmo genetikoa hasierako populazio batez osatuko da eta populazioak eboluzionatu egingo du hobeak diren soluzioetara. Hasierako populazioa ausaz, kuasiausaz edo heuristikoen bitartez hasieratu daiteke. Ondoren, ugalketa prozesua dator.

Ugalketa prozesuan gurutzaketa eta mutazio eragiketak aplikatzen dira. Bi kide gurutzatzea edo crossover operadoreak [11] deskribatzen dira, hala nola puntu bakarrean oinarritutako gurutzaketa, n puntuetan oinarritutako gurutzaketa edo gurutzaketa uniformea. Kide bat mutatzen da haren gene bat aldatzen denean. Normalean oso probabilitate txikiz aplikatzen da. Mutazioari esker populazioko kideen inguruko beste kide batzuk aztertu ahal izango dira, dibertsitatea mantenduz.

Populazio berriaren kide onenak aukeratzen dira. Prozesu honetarako metodo desberdinak [12] azaltzen dira, esate baterako, erruletaren metodoa, ranking araberako aukeraketa edo lehiaketa bidezko aukeraketa.

Ondoren, belaunaldi berria sortzen da. Hainbat modu daude belaunaldi berria sortzeko: sortutako seme-alaba berriak besterik ez sartzea, bi seme-alabek populazioko bi kide txarrenak ordeztu edo gurasorik onenak zuzenean hurrengo belaunaldiko populazioan sartzea (Elitismoa). Azken kasuan erabaki behar da zenbat guraso eta seme-alaba sartu.

Bilaketa prozesua bi arrazoirengatik amaitu daiteke: belaunaldi kopuru maximora iristean edo populazioak konbergitzen duenean [8]. Gene guztiek konbergitu dutenean populazioak konbergitu duela esaten da. Populazioko kideen kodeketan, % 95ek baino gehiagok gene batean balio bera badu, orduan gene horrek konbergitu duela esaten da.

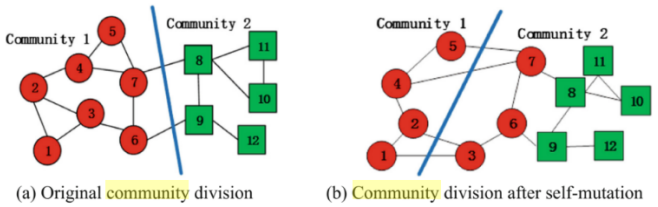
Gure algoritmoaren eskema [9] artikulutik atera da, "Algorithm 1" moldatuz gure kasura:

1: $P =$ hasieraketa(G, k, $size$)
2: $f =$ evaluate(P)
3: $evals = size$
4: **while** $max\_evals > evals$ **do**
5: P', $f' =$ new(P, pc)
6: P, $f =$ update(P, P', f, f')
7: $evals+ = size$
8: **if** ez_da_hobetu_g_belaunalditan **then**
9: break
10: **return** best_fitness(P, f)

Hasierako belaunaldia (pseudokodea 1. lerroa) metodo eraikitzaile batekin edo ausaz sortu daiteke. *size* aldagaiak populazioaren kide kopuruari erreferentzia egiten dio, eta *k* aldagaiak komunitate kopuruari. Hasieratze-metodoak aurretik *simulated annealing*-erako azaldutakoak dira.

Belaunaldi bakoitzean populazio berria sortzen da birkonbinazio edo mutazioa aplikatuz (pseudokodea 5. lerroa). *Pc* aldagaiak birkonbinaketa egoteko probabilitatea adierazten du. Birkonbinaketa gertatzen ez bada, mutatu egingo dira gurasoak. Gurasoak ausaz aukeratuko dira.

Puntu bateko birkonbinaketa aplikatzen da. Gurasoak puntu baten bidez gurutzatuko dira, lehenengo gurasoaren geneak hartuko dira hasieratik punturaino, eta bigarren gurasoren geneak puntutik bukaeraraino. Puntua ausaz aukeratuko da. Kideak binaka birkonbinatuko dira, bikoteak ausaz aukeratuta.



(a) Original community division (b) Community division after self-mutation

Irudia 1. Self-mutazio adibidearen diagrama.

Mutazioa [10] artikuluan deskribatzen den *Self-mutation operation*-ean oinarrituta dago. Ausazko nodoa aukeratzen da eta auzokidea den komunitate batera aldatzen da, baita nodo honen auzokideak ere. Adibidez, 1. irudian ikusi daitekeen bezala, 6. nodoa 1 komunitatetik 2 komunitatera mugitzen da.

Hurrengo belaunaldiaren populaziorako, gurasoak eta seme-alabak nahasi eta guztien artetik onenak aukeratzen dira.

Algoritmoa bi arrazoirengatik bukatu daiteke. Lehenengoa, ebaluazio kopurua gainditzen denean (pseudokodea 4. lerroa). Bigarrena, fitness hoberena max_g belaunalditan hobetu ez bada (pseudokodea 8. lerroa).

V. Esperimentazioa

Algoritmoen portaera aztertzeko, lehenik parametro-bilaketa bat egin dugu. *Grid search* edo lauki-sare bilaketa erabili dugu, algoritmoen exekuzio-denborak ez digulako metodo aurreratuagoak erabiltzeko aukerarik ematen. Aurreko ataletan azaldutakoari jarraituta, algoritmo bakoitzean finkatu gabeko bi parametro geratzen zaizkigu. SAren kasuan, p_0 hasierako probabilitatea eta α hozte-faktorea dira. GAn, berriz, pc birkonbinatzeko probabilitatea eta $size$ populazio-tamaina. Hemendik aurrera aipatzen ditugun exekuzio guztietan 10.000ko

Genetic algorithm, 10 komunitate

Pc \ Population size	100	200	300	400	500
0,1	0,8671	0,8712	0,8711	0,8710	0,8713
0,3	0,8715	0,8719	0,8734	0,8747	0,8745
0,5	0,8771	0,8757	0,8758	0,8766	0,8754
0,7	0,8845	0,8805	0,8747	0,8740	0,8723
0,9	0,8754	0,8717	0,8661	0,8660	0,8647

Genetic algorithm, 20 komunitate

Pc \ Population size	100	200	300	400	500
0,1	0,9070	0,9092	0,9115	0,9092	0,9085
0,3	0,9084	0,9134	0,9108	0,9127	0,9090
0,5	0,9149	0,9168	0,9098	0,9100	0,9075
0,7	0,9204	0,9122	0,9082	0,9040	0,9024
0,9	0,8842	0,8881	0,8770	0,8780	0,8787

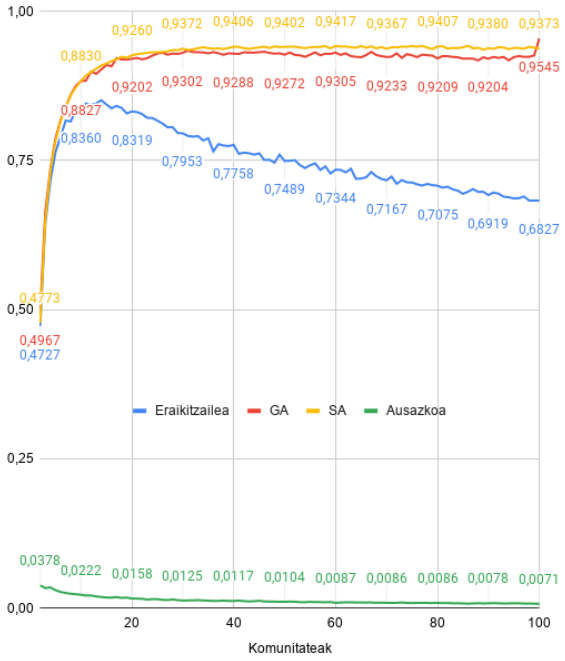
Simulated annealing, 10 komunitate

p0 \ hozte_faktorea	0,1	0,3	0,5	0,7	0,9
0,1	0,8862	0,8872	0,8884	0,8865	0,8863
0,3	0,8872	0,8872	0,8873	0,8872	0,8855
0,5	0,8823	0,8852	0,8864	0,8849	0,8764
0,7	0,8832	0,8855	0,8840	0,8765	0,8362
0,9	0,8800	0,8817	0,8659	0,8198	0,7838

Simulated annealing, 20 komunitate

p0 \ hozte_faktorea	0,1	0,3	0,5	0,7	0,9
0,1	0,9328	0,9363	0,9357	0,9365	0,9352
0,3	0,9292	0,9315	0,9305	0,9305	0,9279
0,5	0,9278	0,9272	0,9295	0,9281	0,9176
0,7	0,9232	0,9252	0,9275	0,9176	0,9053
0,9	0,9215	0,9237	0,9136	0,8624	0,8150

Irudia 2. *Grid search*-erako parametroak eta emaitzak.



Irudia 3. 2tik 100 komunitatera lortutako modularitateak.

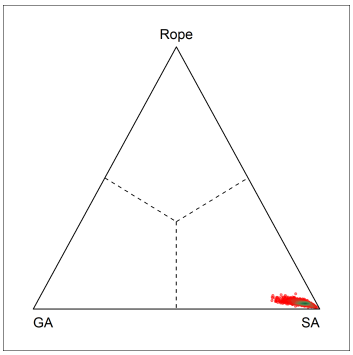
ebaluazio kopuru maximoa ezarri dugu, eta emaitza bakoitza 10 errepikapenetako modularitateen batezbestekoa da.

Bi algoritmoekin eraikitzaile bidezko hasieraketa erabili dugu. Hasiera batean 10 komunitaterekin egin dugu bilaketa, baina, batez ere GAren emaitzen arteko diferentzia txikiegia ote zen kezkagatik, 20 komunitaterekin ere probatu dugu. Probatutako parametroak eta lortutako emaitzak 2. irudiko tauletan agertzen dira. Emaitza horiek ikusita, hurrengo probetarako parametro hauek aukeratu ditugu: SArako $p_0 = 0,1$ eta $\alpha = 0,5$, eta GArako $pc = 0,7$ eta $size = 100$.

Eraikitzaile bidez hasieratuta emaitza hobeak lortzen direla erakusteko, bi algoritmoak ausaz hasieratuta exekutatu ditugu, aurretik finkatu ditugun parametroekin eta 20 komunitaterekin. SArako 0,8585 lortzen da batez beste, eta GArako 0,6969. Beraz, esan dezakegu eraikitzaile baten bidez hasieratzea egokiagoa dela.

3 irudiko grafikoan ikus daitekeen moduan, bi algoritmoak 2tik 100erako komunitate kopuru guztiekin exekutatu ditugu, emaitzak elkarren artean konparatzeko.[1] Ausazko bilaketa bidez lortutako emaitzak ere jarri ditugu, oinarri-lerro moduan. Gainera, metodo eraikitzaileak lortzen duen modularitatea ere adierazita dago. Ikusi daiteke bi algoritmoen emaitzak antzekoak direla, baina ausazko bilaketarekin alderatuta modularitate hobeagoa lortzen dute. Desbiderapenari dagokionez, errepikapenen arteko aldea oso txikia da. SAren desbiderapen maximoa 0,03koa izan da. GAn, aldiz, 0,07koa izan da. Metodo eraikitzaileari dagokionez, ikusi daiteke komunitate gutxirekin asko laguntzen duela, baina, 20 komunitateetatik aurrera okertzen doala.

[1]Probak egiteko kodea, emaitzak eta simulazio batzuk esteka honetan: https://github.com/b-alkain/CDP_proiektua



Irudia 4. Irabazi-galdu-berdindu probabilitateentzat kalkulatutako *a posteriori* banaketa irudikatzeko *simplex* diagrama.

Emaitzen analisi estatistikoa egiteko bayestar metodo bat erabili dugu, [13] artikulua eta bertako kodea adibide hartuta, [14] artikuluan oinarritutakoa. Aurretik aipatutako batez besteko modularitateak erabili ditugu, 2tik 100 komunitatera bilatuz lortutakoak ($-modularitatea$ erabili dugu, erabilitako kodeak balioak minimizatu nahi izatea espero baitu). Berdinketa bezala hartu dugu 0,001 baino diferentzia txikiagoa egotea. Emaitzak 4. irudiko *simplex* diagraman daude.

Puntu bakoitzak irabazi-galdu-berdindu probabilitateentzat kalkulatutako *a posteriori* banaketaren laginketa bat irudikatzen du. Lortutako puntu guztiak SA erpinetik hurbil daude, beraz, ziurtasunez esan dezakegu SA algoritmoarekin emaitza hobeak espero ditugula, egindako esperimentuen arabera. Lortutako probabilitateen batezbestekoak hauek dira: SA hobea izatekoa 0,9245, berdinketarena 0,0226 eta GA hobea izatekoa 0,0529.

VI. Ondorioak

Proiektu honetan bi algoritmo desberdin azaldu egin dira: soluzio bakarrean oinarritutako suberaketa simulatua eta populazioan oinarritutako algoritmo genetikoa. Algoritmoak NIPS arazoa ebazteko egokitu egin dira bai kodeketan, bai hasieraketan, bai hiperparametroen egokitzapenean. Azken honetarako grid search erabili da. Algoritmoak konparatzeko, 2tik 100erako komunitate kopuruak exekutatu dira. Oinarri lerrotzat ausazko bilaketa erabili da. Bi algoritmoek modularitate altuak itzultzen dituzte. Bayestar metodoa erabiliz ikusi da suberaketa simulatua egokiagoa dela problema honetarako.

ERREFERENTZIAK

[1] Clauset, A.; Newman, M. E. J. & Moore, C. *Finding community structure in very large networks*, Phyisical Review E 2004, 70, 066111

[2] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, *Defining and identifying communities in networks*. Proc. Natl. Acad. Sci. USA 101, 2658–2663, 2004

[3] Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science. 220 (4598): 671–680.

[4] Černý, V. (1985). *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*. Journal of Optimization Theory and Applications. 45: 41–51.

[5] Gutiérrez Andrade, Miguel Ámgel; de los Cobos Silva, Sergio Gerardo; Pérez Salvador, Blanca Rosa (1998). *Optimización con recocido simulado para el problema de conjunto independiente*. En Línea^2 (Universidad Autónoma Metropolitana) 3

[6] Park, M.W., Kim, Y.D., 1998. *A systematic procedure for setting parameters in simulated annealing algorithms*. Comput. Oper. Res. 25 (3), 207–217.

[7] *Genetic Algorithms in Search, Optimization and Machine Learning*. D. Goldberg. Addison-Wesley Publishing Company, 1989

[8] De Jong, 1975

[9] Chuan Shi, Zhentu yan, yi wang, yanan cai eta Bin Wu, *A genetic algorithm for detectiong communities in large-scale complex networks*, Advances in Complex Systems, Vol. 13, No. 1, 2010

[10] Xingming Sun, Han-Chieh Chao, Xingang You, Elisa Bertino, *Cloud Computing and Security*, Third International Conference, ICCCS 2017, Nanjing, China, June 16-18, 2017

[11] A.J. Umbarkar, P.D. Sheth, *Crossover operators in genetic algorithms: a review*, ICTACT Journal on soft computing Vol. 6, ISSUE: 1, 2015

[12] K. Jebari, M. Madiafi, *Selectin Methods for Genetik Algorithms*, Int. J. Emerg. Sci., 3(4), 333-344, 2013

[13] Joan Alza, Josu Ceberio, Borja Calvo, *Balancing the Diversification-Intensification Trade-off Using Mixtures of Probability Models*, 2018 IEEE Congress on Evolutionary Computation

[14] A. Benavoli, G. Corani, J. Demsar, and M. Zaffalon, *Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis*, Journal of Machine Learning Research, vol. 18, no. 77, pp. 1–36, 2017

[15] J. H. Holland, University of Michigan Press, Ann Arbor, *Adaptation in Natural and Artificial Systems*, 1975