Community Detection with ILS and GA-EDA

Aitor Zubillaga
UPV/EHU
azubillaga012@ikasle.ehu.eus

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

ABSTRACT

Komunitateak detektatzea, orokorrean grafo egiturak erabiliz, azkeen urteetan indar handia hartzen ari den problema bat da. Hala eta guztiz ere ez da lortu behar bezalako soluzio bat. Artikulu honetan, bi algoritmo proposatzen dira grafoetan komunitateak detektatzeko: Iterated Local Search-en aldaera bat, eta Genetic Algorithm eta Estimation Distribution Algorithm-en arteko konbinazio bat. Emaitzen egokitasuna egiaztatzeko baseline bezala erabili den Random Search-ekin alderatuz, emaitza oso onak lortu dira.

1 SARRERA ETA MOTIBAZIOA

Azken urteetan sare sozialek indar handia hartu dute. Arrazoi nagusienetako bat, gure interesak partekatzen dituzten pertsonak aurkitzeko ematen dituzten erraztasunak dira [2]. Interesak edo erlazioak partekatzen dituzten pertsonen multzoei komunitate deitzen zaie. Pertsona multzo batean komunitateak aurkitzeko gai izanez gero, posible da erabiltzaileari bere komunitate berdineko beste erabiltzaileei gogoko duten orrialdeen edo pertsonen gomendioak egitea, besteak beste. Hori dela eta, enpresak interes handia dute pertsonen artean komunitateak detektatzen.

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. 1. irudian ikus daiteke adibide bat.

Gure probleman, NIPS kongresuan publikatzen duten autoreen komunitateak aztertu nahi 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.

Lan honen helburua grafoetan komunitateak aurkitzeko bi algoritmo diseinatu eta aztertzea da. Bata soluzio bakarrean oinarritutakoa izango da, eta bestea poblazionala.

Bilaketa espazioa $\{1, ..., k\}^n$ 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.

Lortutako emaitzak aztertzeko modularitatea erabiliko da [2]. Modularitateak sare baten komunitate banaketaren egokitasuna neurtzen du. Banaketa egokia izango da komunitateen barruan ertz asko badaude, eta komunitate artean gutxi. Ausazko sare batean espero dezakegun modularitatea 0 da, eta modularitate maximoa 1 da. Beraz, gure helburua modularitatea maximizatzea izango da.

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

Lehenengo kantitateak i komunitatetik hasita j komunitatea lotzen duten ertzen pisuen frakzioa adierazten du:

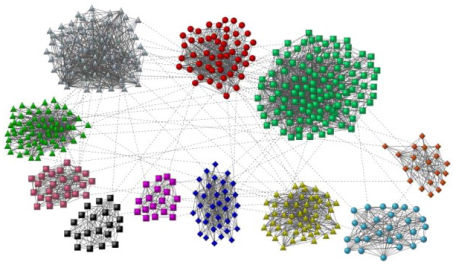

Figure 1: Komunitateak grafo batean

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

Bigarren kantitateak amaiera i komunitateko nodoetan duten pisuen frakzioa adierazten du:

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

Aurreko bi ekuazioetako aldagaien esanahia honakoa da:

- A_{vw} v nodotik w nodora doan ertzaren pisua da.
- c_v eta c_w v eta w nodoen komunitatea dira.
- $\delta(i, j)$ 1 izango da $i = j$ bada eta 0 bestela.
- $m = \frac{1}{2} \sum_{vw} A_{vw}$ grafoko ertzen pixuen batura da.
- $k_v = \sum_w A_{vw}$ gradua erpin horri lotutako ertzen pixuen batura da.

2 ALGORITMOEN DISEINUA

Proiektu honetan bi algoritmo desberdin inplementatu dira, hauetako bat soluzio bakarrekoa eta bestea poblazionala. Soluzio bakarreko algoritmoa ILS bat da. Algoritmo poblazionala aldiz GA-EDA algoritmoa inplementatu da. Bi algoritmoek hasierako soluzioak lortzeko metodo eraikitzaile estokastiko bat erabiltzen dute, Louvain algoritmoan oinarritzen dena. Beraz, hasteko Louvain azalduko da 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, hauek elkartzearen ondorioz modularitatea hobetzen bada bakarrik. Beraz, komunitate kopuru optimora iritsitakoan gelditu egiten da. Gure kasuan aldiz, emandako komunitate

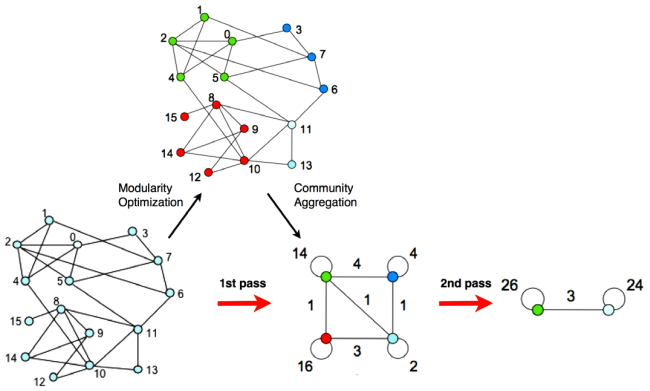

Figure 2: Louvain algoritmoa: Iterazio bakoitzak bi fase ditu, aldaketa lokala eta komunitateen agregazioa.

kopurura iristea interesatzen zaigu, nahiz eta beste komunitate kopuru batekin modularitate hobea lortzea posible izan. Hori dela eta aldaketak egin dira `best_partition` funtzioan, alde batetik, deseatutako komunitate kopurura iritsitakoan algoritmoa gelditzeko, eta beste aldetik, nahiz eta soluzio optimoa aurkitu, modularitatea ahal den gutxiena txartuz, desiratzen den komunitate kopurura iritsi ahal izateko.

Hala eta guztiz ere, ezin da lortu proiektu honetan erabili den instantziarekin komunitate kopurua 279tik jaistea, izan ere, komunitateen artean ertz gehiago ez baitago.

Hori konpontzeko, `best_fixed_partition` funtzioa inplementatu dugu. Honek, lehenengo `best_partition` funtzioari deituko dio. Honek, jasotzen duen soluzioari, soberan dituen komunitate txikienak hartu eta guztiak komunitate berean elkartuko dizkio. Izan ere, komunitateen artean konexiorik ez dagoenean, ahal den modularitate hoberena lortzeko modurik onena komunitate txikienak elkartzea da, komunitate handiak eragin handiago baitute modularitatean eta hauek txartzeak beraz, asko jaisten baitu fitnessa. Komunitate txikienak non elkartu aukeratzeko, aukera guztiak aztertu eta onena aukeratzen da.

2.2 ILS

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

Perturbatzeko, lehenik soluzioan 2 komunitate ausaz hartu eta elkartu egiten dira, eta ondoren komunitate bat ausaz hartu eta ausaz aukeratutako bi zatitan banatzen da. 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 erabiltzen du, emaitza txarragoak onartu ahal izateko. Gero eta tenperatura handiagoa, orduan eta probabilitate hanfiagoa dago emaitza txarrak onartzeko. Hasierako tenperatura eta tenperatura egokitzeko modua, 3.1 azpiatalean azalduko da.

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
```

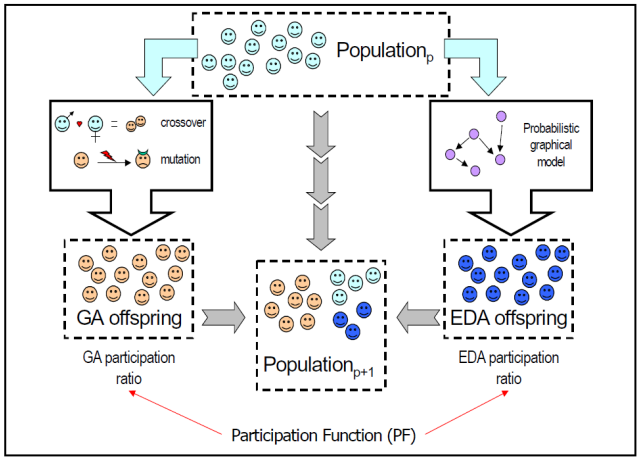

Figure 3: GA-EDA algoritmoan iterazio bat nola osatzen den ikus daiteke.

2.3 GA-EDA

GA-EDA [4] populaziotan oinarritutako algoritmoa inplementatu da. 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 daitezke. 2. pseudokodean eta 3. irudian ikus daitezke honen funtzionamedua.

Hasteko, hasierako populazioa eraikitzen da, ausaz edo moldatutako Louvain heuristikoa erabiliz. Gero, hainbat iterazio egiten dira, ebaluazio kopuru maximora iritsi arte. Iterazio bakoitzean pauso berdinak errepikatuko dira. Amaitzeko, populazioko soluzio onena itzultzen da.

- Selection eragiketan hainbat soluzio aukeratzen dira aurreko populaziotik. `best_selection` aukeraketan beti soluzio onenak aukeratzen dira. `roulette_selection` aukeraketan soluzioen probabilitateak kalkulatzeko fitness balioa erabiltzen da. `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 gordetzen dira. `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.

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. 4. irudian ikus daitezke 2 komunitaterako balioak.Beraz, esperimentazioaren hurrengo faserako parametro horiek erabiliko ditugu.

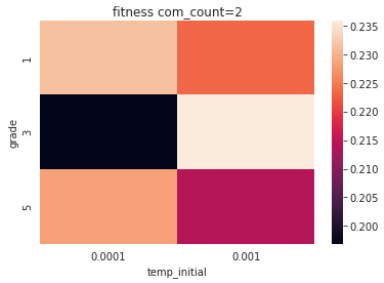

Figure 4: ILS algoritmoaren parametroen heatmapa 2 komunitaterekin.

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.

Gainerako parametroei dagokionez, balio estatikoak aukeratu ditugu. Offspring tamainak populazio tamainaren erdira finkatu ditugu, hau da, 15. Mutazio probabilitatea 0.01-en ezarri dugu. Hasierako populazioa sortzeko `louvain_population`, aukeraketarako `roulette_selection`, eguneratzeko `elitism_selection`, birkonbinaziorako `one_point_crossover` eta mutaziorako `bit_mutation`.

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. Fitness eta denbora emaitzekin boxplot motako grafikak atera ditugu, x ardatzean komunitate kopurua jarriz. Gainera, fitness emaitzen batazbestekoak 1. taulan ikus daitezke.

Fitness-ari dagokionez, ILS eta GA-EDA askoz hobeak direla ikus daiteke. Eta horien artean GA-EDA pixkat hobea da komunitate kopuru guztietarako. Orokorrean, bariantza txikia da eta horregatik kutxa batzuk ia ez dira ikusten. Ikusi 5. irudia.

Analisi estatistiko Bayesiarra ere gauzatu dugu ILS eta GA-EDA algoritmoak konparatzeko. Honetarako, bi algoritmoen artean berdinketa dagoela erabaki dugu aldea $5 * 10^{-3}$ baino txikiagoa denean. Analisiaren emaitzak 5. irudian ikus daitezke. Laburbilduz, GA-EDA hobea izateko probabilitatea 0.698 da, berdinak izatekoa 0.301 eta okerragoa izatekoa 0.000.

Denborari dagokionez, ILS eta RS askoz azkarragoak dira GA-EDArekin konparatuz. Komunitate kopurua handitu ahala, GA-EDAk behar duen denbora handitzen joaten da. Beste bi algoritmoetan, berriz, gutxi gorabehera mantendu egiten da. Ikusi 7. irudia.

Komunitateak	RS	ILS	GA-EDA
2	0.0021	0.1924	0.3131
10	-0.0038	0.7184	0.7344
20	0.0001	0.8082	0.8128
30	-0.0018	0.8552	0.8593
40	0.0008	0.8841	0.8866
50	0.0009	0.9000	0.9087
60	-0.0001	0.9154	0.9238
70	0.0001	0.9249	0.9342
80	0.0003	0.9348	0.9406
90	-0.0009	0.9389	0.9451
100	-0.0002	0.9387	0.9495

Table 1: Batazbesteko fitness balioak 3 algoritmoetan komunitate kopuru bakoitzerako.

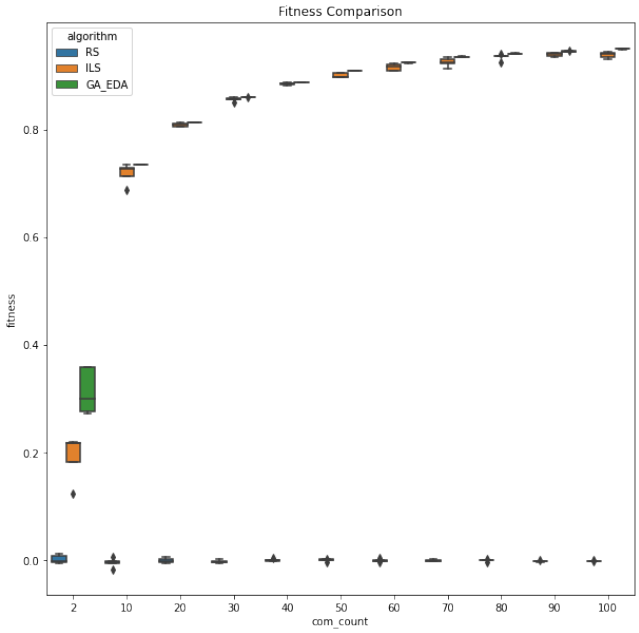

Figure 5: Fitness-aren boxplota 3 algoritmoekin komunitate kopuru bakoitzerako.

4 ONDORIOAK

Azken urteetan komunitateak detektatzeak indar handia hartu du, batik bat sare sozialen hazkundea dela eta. Artikulu honetan, bi algoritmo proposatu dira grafoetan komunitateak detektatzeko: bata ILS-ren aldaera, eta bestea GA-EDA. Emaitzen egokitasuna egiaztatzeko, Random Search erabili da baseline bezala. Honekin alderatuz emaitza onak lortu dira bi algoritmoekin. Alde batetik, GA-EDAk emaitza apur bat hobeak lortzen ditu, batik bat komunitate kopurua oso mugatua denean. Bestalde, ILS algoritmoa eraginkorragoa da denbora askoz gutxiago behar baitu soluzio egokietara iristeko.

Lan honen mugak kontuan hartuz etorkizunean hainbat hobekuntza egin daitezke. Algoritmoen parametroak kalibratzerakoan aukera gutxi probatu dira denbora murriztapenen ondorioz. Proba gehiago

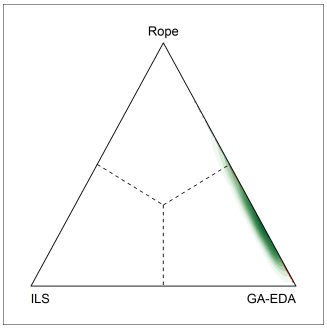

Figure 6: Simplex grafikoa ILS eta GA-EDA algoritmoen irabazteko, berdintzeko eta berdintzeko probabilitateekin.

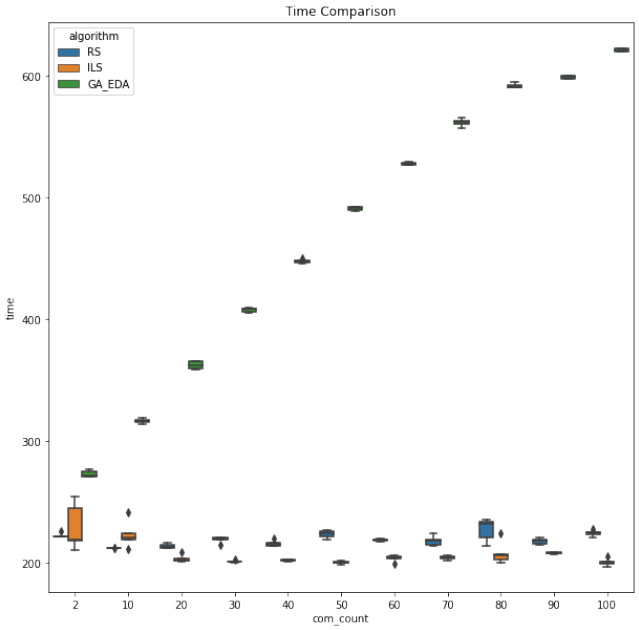

Figure 7: Denboraren boxplota 3 algoritmoekin komunitate kopuru bakoitzerako.

egiteko exekuzio denbora gehiago murriztea komeniko litzateke. Honetaz gain, 10^4 ebaluazio egin ordez 10^5 edo 10^6 ebaluazio egitea hobe izango litzateke. Era berean, errepikapen kopurua ere gutxienez 10era handitzea komenigarria izango litzateke.

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] J. M. Peña, V. Robles, P. Larrañaga, V. Herves, F. Rosales, and M. S. Pérez. 2004. GA-EDA: Hybrid Evolutionary Algorithm Using Genetic and Estimation of Distribution Algorithms. In *Innovations in Applied Artificial Intelligence*, Bob Orchard, Chunsheng Yang, and Moonis Ali (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 361–371.