BHLN: Helburu anizkoitzeko optimizazioa

Borja Calvo, Josu Ceberio

Laburpena

Orain arte optimizazio problemak nola ebazten diren ikusi dugu, baldin eta soluzioak ebaluatzeko era bakar bat badugu baina, zer gertatzen da ebaluazio funtzio bat baino gehiago daukagunean? Kapitulu honetan helburu anizkoitzeko optimizazioan agertzen diren kontzeptuak aztertuko ditugu, menderatzea, batik bat. Horrez gain, helburu funtzio bat baino gehiago duten problema ebazteko algoritmo ebolutibo bat aztertuko dugu: MOEA/D.

1 Helburu funtzioak eta menderatzea

Optimizazio problema arruntetan (helburu funtzio bakarra dutenak, alegia) bi soluzio dituguneen zein den hobea jakitea erraza da. Helburu funtzioa (f) minimizatu nahi badugu, x y baino hobea izango da baldin eta bakarrik baldin $f(x) < f(y)$ bada. Helburu funtzio bat baino gehiago daukagunean, berriz, gauzak ez dira hain errazak. Demagun bi funtzio ditugula, f_1 eta f_2, eta hiru soluzio, x_1, x_2 eta x_3, zeinen ebaluazioa hauxe den:

$$f_1(x_1) = 10;\ f_2(x_1) = 524$$
$$f_1(x_2) = 5;\ f_2(x_2) = 224$$
$$f_1(x_3) = 3;\ f_2(x_3) = 1524$$

Bi helburu funtzioak minimizatu nahi baditugu, x_2 soluzioa x_1 baino hobea dela argi dago, $f_1(x_2) = 5 < f_1(x_1) = 10$ eta $f_2(x_2) = 224 < f_2(x_1) = 524$ direlako baina, zer gertatzen da hirugarren soluzioarekin? Lehenengoarekin alderatzen badugu, $f_1(x_3) = 3 < f_1(x_1) = 10$ dela ikusiko dugu baina, aldi berean, $f_2(x_3) = 1524 > f_2(x_1) = 524$ da. Hortaz, ezin dugu esan x_3 x_1 baino hobea denik; x_2 soluzioarekin alderatzen badugu ere gauza bera gertatuko da.

Horrelako egoerak aztertu ahal izateko kontzeptu berri bat definitu behar dugu: menderatzea.

1.1 Menderatzea

Menderatze kontzeptua oso sinplea da: soluzio batek, x, beste bat, y, menderatzen du, baldin eta bakarrik baldin helburu funtzio guztietarako x y berdina edo hobea bada eta, gutxienez batentzako, hobea bada. Hau da, matematikoki x soluzioak y menderatzeko bi baldintza hauek bete behar dira:

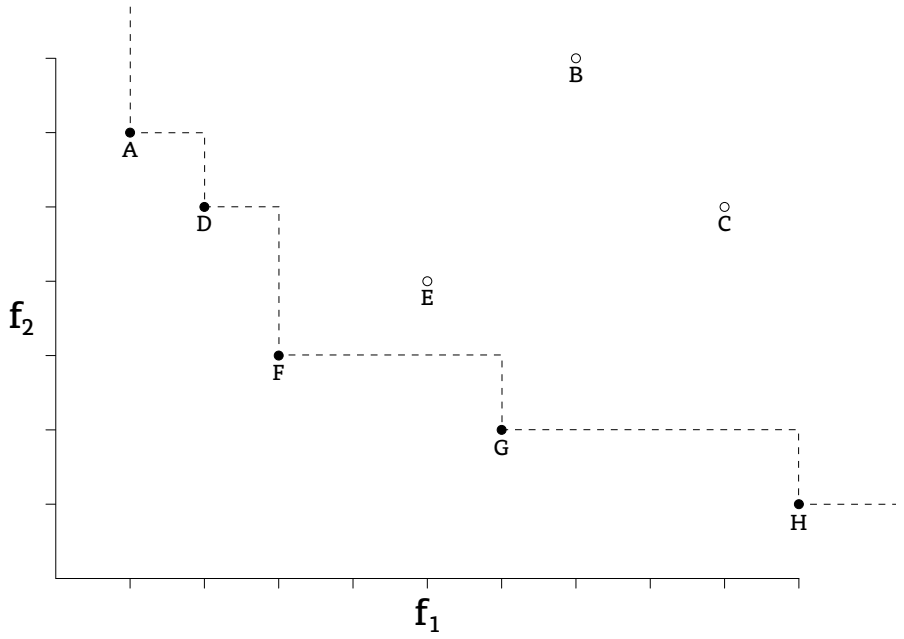
$$\forall i\ f_i(x) \leq f_i(y)$$
$$\exists j\ f_j(x) < f_j(y)$$

non f_i problemaren helburu funtzioak diren eta gure xedea denak minimizatzea den.

Demagun minimizatu nahi dugun bi helburu funtzio ditugula, eta ondoko soluzio hauek:

Ikasturtea: 2013/2014
Irakaslea: Borja Calvo, Josu Ceberio
Telefonoa: 943 01 50 13
E-posta: borja.calvo@ehu.es

GIPUZKOAKO CAMPUSA
CAMPUS DE GUIPUZCOA
Paseo Manuel de Lardizabal, 1
20018 Donostia (Gipuzkoa)



Irudia 1: Helburu anizkoitzeko optimizazio adibide bat. Grafikoan zazpi soluzio, bakoitza bi funtzio (f_1 eta f_2) erabiliz ebaluatua.

	f_1	f_2
A	1	6
B	7	7
C	9	5
D	2	5
E	5	4
G	6	2
H	10	1

1 irudian soluzioak helburu funtzioen espazioan kokatuta daude. E soluzioa hartzen badugu, F soluzioa hobea dela argi dago, bi helburu funtzioentzako F-ren balioa E-rena baino txikiagoa baita. Era berean, E soluzioa B soluzioa baino hobea da bi funtzioetarako. Beraz, F soluzioak E menderatzen du eta, aldi berean, E soluzioak B soluzioa menderatzen du. E eta B soluzioak menderatuta daude, baina F soluzioa menderatzen duen soluziorik ez dugu; F soluzioa Pareto-optimoa dela esango dugu.

Definizioa 1.1 *Izan bedi helburu anizkoitzeko problema baterako x soluzioa. Bilaketa espazioan x soluzioa menderatzen duen soluziorik existitzen ez bada, x Pareto-optimoa da.*

Helburu funtzio bat baino gehiago ditugunean aurreko kapituluetan erabili dugun optimotasun kontzeptua ezin da aplikatu. Beraz, bere ordez Pareto-optimotasun kontzeptua erabiliko dugu. Aintzat hartzekoa da helburu funtzio bat baino gehiago dugunean bi soluzio Pareto-optimoak izan daitezkeela, nahiz eta oso ezberdinak izan – ikusi grafikoan, adibidez, A eta H soluzioak –.

Adibideko irudian, esate baterako, A, D, F, G eta H soluzioak Pareto-optimoak dira, ez baitago soluziorik hauek menderatzen dituena – eta beraien artean ere menderatzen ez direlako, noski –. Pareto-optimoak diren soluzio multzoari Pareto-multzoa – *Pareto set*, ingelesez – deritzo, eta Pareto-multzoan dauden soluzioen ebaluazioari, berriz, Pareto-frentea – *Pareto front*, ingelesez –.

Helburu anizkoitzeko optimizazio problema bat ebatzi behar dugunean gure helburua Pareto-multzoa estimatzea da. Alabaina, Pareto-optimoak diren soluzio guztiak topatzea ezinezkoa denez, bilaketan menderaturik gabe dauden soluzioak jasoko ditugu[1]

Helburu funtzio bat baino gehiago izateak ez du esan nahi problema helburu anizkoitzekoa izango dela; Pareto-multzoan soluzio bakarra baldin badaukagu, optimoa topatzeko nahikoa izango da edozein helburu funtzio optimizatzea, eta ez denak aldi berean.

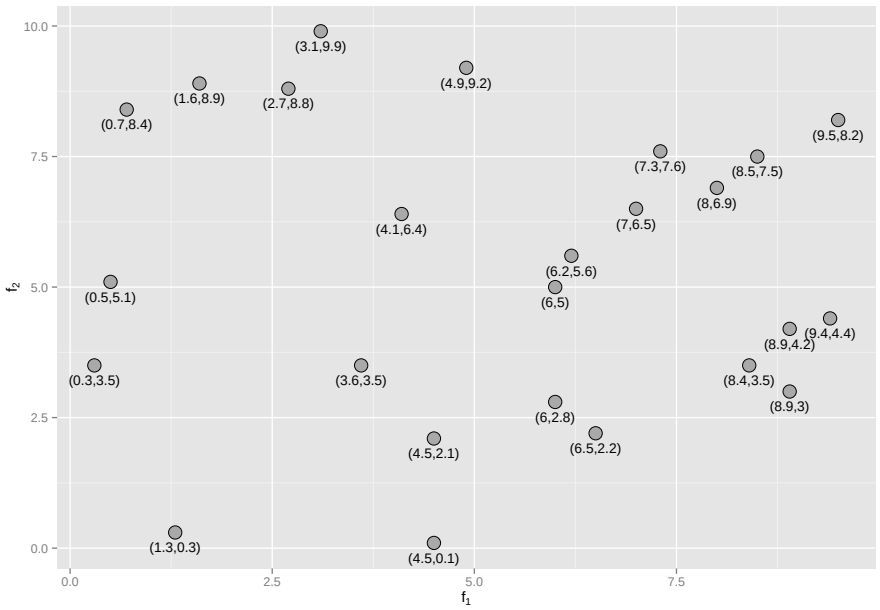
1.2 Ariketak

Ariketa 1.1 *Izan bitez ondoko soluzioak, 3 helburu funtzio dituen optimizazio problema baterako:*

Soluzioa	f_1	f_2	f_3
A	*92*	*19*	*76*
B	*54*	*19*	*76*
C	*7*	*80*	*22*
D	*49*	*16*	*82*
E	*59*	*3*	*25*

- *Zein soluzio daude Pareto-multzoan ondoko egoeratan?*
  - *Helburu funtzio guztiak minimizatu nahi ditugu*
  - *Helburu funtzio guztiak maximizatu nahi ditugu*
- *Demagun f_1 eta f_2 minimizatu eta f_3 maximizatu nahi dugula. D soluzioak A soluzioa menderatzen du? eta alderantziz, A-k D?*

Ariketa 1.2 *Izan bedi ondoko soluzio multzo, bi helburu funtzio dituen optimizazio problema baterako*



- *Markatu grafikoan zein soluzio dauden Pareto-multzoan ...*
  - *... f_1 helburu funtzioa maximizatu eta f_2 minimizatu nahi badugu*
  - *... f_2 helburu funtzioa maximizatu eta f_1 minimizatu nahi badugu*
- *Demagun soluzio berri bat sartzen dugula, $(0.35, 10)$. Nola aldatzen dira aurreko ataleko Pareto-multzoak?*

[1] Helburu funtzio bakarra genuen bezala, orokorrean soluzio zehatza lortzea ezinezkoa da, problemak NP-zailak izan ohi direlako

2 MOEA/D algoritmoa

Helburu anizkoitzeko problemak ebazteko algoritmo asko proposatu dira. Atal honetan algoritmo ebolutibo bat ikusiko dugu, *Multiobjective Evolutionary Algorithm based on Decomposition* edo MOEA/D deritzona.

Algoritmoaren atzean dagoen kontzeptua oso erraza da; izan ere, helburu bat baino gehiago dugunean hurbilketarik sinpleena du oinarrian: helburu funtzioak konbinatu. Demagun TSP motako problema bat dugula non, ibilbidearen kostua zein denbora minimizatu nahi dugula. Aukera sinple bat bien bataz bestekoa minimizatzea izango zen baina, zergatik suposatu bi helburuek garrantzi berdina dutela? Zergatik ez aldatu bakoitzaren ekarpena?

MOEA/D algoritmoak horixe bera egiten du. Populazio bat izango dugu – ez ahaztu algoritmo ebolutibo bat dela –, baina populazioko indibiduo bakoitzari helburu bakarreko azpi-problema bat esleituko diogu.

Demagun bi helburu funtzio ditugula, f_A eta f_B, eta 101 indibiduo sortu nahi ditugula. Lehenengo indibiduoari helburu funtzio berri bat esleituko diogu: $f_1 = 1.00f_A + 0.00f_B$. Hurrengoari $f_1 = 0.99f_A + 0.01f_B$ funtzioa esleituko diogu, hirugarrenari $f_1 = 0.98f_A + 0.02f_B$ funtzioa etab; azken bi soluzioei $f_{100} = 0.01f_A + 0.99f_B$ eta $f_{101} = 0.00f_A + 1.00f_B$ funtzioak esleituko dizkiegu.

Algoritmoaren iterazio bakoitzean azpi-problema guztiak heuristikoren bat – bilaketa lokal bat, esate baterako – erabiliz ebatzi behar dira. Iterazio batetik bestera soluzioak aldatzen ez badira, litekeena lehenengo iterazioan algoritmoa trabaturik gelditzea da. Hori ez gertatzeko, iterazio bakoitzean heuristikoa bi soluzio hartuz eta gurutzatuz hasieratzen da. Gurutzatuko ditugun soluzioak aukeratzerakoan soluzio horiek uneko azpi-problemarako onak izatea espero dugu. Hori ziurtatzeko MOEA/D algoritmoak ideia sinple batean oinarritzen: Soluzio bat problema baterako ona bada, «gertu» dauden problemetarako ere ona izango da.

Laburbilduz, iterazio batean indibiduo bakoitzeko ingurunean dauden bi indibiduo ausaz aukeratuko ditugu eta, gurutzatu ondoren, areagotze prozesu bat – bilaketa lokal bat, esate baterako – lortutako soluzioarekin abiatuko dugu; prozesu hau indibiduoari dagokion azpi-problemaren helburu funtzioak gidatuko du. Prozesu honek itzultzen duen soluzioa azpi-problemarako ona izango da eta, unekoa baino hobea bada, ordezkatuko dugu. Era berean, soluzioa inguruneko azpi-problemetarako ere ona izan daiteke. Horrela balitz, inguruneko soluzioak ere ordezkatuko genituzke.

Hona hemen algoritmo originalaren deskripzioa – sinplifikatua –:

- **Sarrera:**
  - Helburu funtzio zerrenda
  - Gelditzeko baldintza bat
  - Azpi-problema kopurua, N – edo populazio tamaina –
  - Helburu funtzioen pisuak azpi-problema bakoitzeko
  - Ingurunearen tamaina T
- **Irteera:**
  - Pareto multzoaren estimazioa, EP
- **1 - Hasieraketa:**
  - $EP = \emptyset$
  - Ezarri, azpi-problema bakoitzeko, zein dira ingurunean daudenak
  - Azpi-problema bakoitzeko sortu – ausaz, adibidez – hasierako soluzioa
- **2 - Eguneraketa:** i bakoitzeko, 1etik Nra
  - **Gurutzatzea** - i problemaren ingurunetik bi soluzio ausaz aukeratu eta gurutzatu, soluzio berri bat, y, sortzeko
  - **Hobekuntza** - y soluzioa zuzendu/hobetu heuristiko bat erabiliz
  - **Eguneraketa** - Beharrezkoa bada, i indibiduoaren inguruan daudenak lortutako soluzioarekin ordezkatu; eguneratu EP, soluzio berria sartuz dominatuta ez badago eta, beharrezkoa bada, soluzio berriak dominatzen dituen soluzioak EP multzotik ezabatuz
- **3 - Gelditzeko baldintza** Algoritmoaren gelditzeko baldintza betetzen ez bada, 2. puntura itzuli