eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA INGENIARITZAKO GRADUA
Konputazioa

GRADU AMAIERAKO PROIEKTUA

# *Bradley-Terry* ereduaren errendimenduaren analisia *Linear Ordering Problem*an

Egilea:
Joan Alza Santos

Zuzendariak:
Borja Calvo Molinos

Josu Ceberio Uribe

informatika fakultatea
facultad de informática

2017ko urria

# Laburpena

Azken hamarkadetan optimizazioan oinarritutako problemetan ikerketa asko egin dira, batez ere *Estimation of Distribution Algorithms* (EDA) algoritmoetan. Algoritmo ebolutiboen barruan sailkatzen dira, eta problema mota desberdinak ebazteko erabili ohi dira. Hala ere, algoritmo horiek ez dira asko garatu permutazio-problemak ebazteko. Horretarako, azken aldian, hobekuntza ugari egin dira eredu probabilistikoak erabiliz. Iterazio bakoitzean informazio garrantzitsuena duen soluzio-multzoa eskuratzeko erabiltzen dira ereduak. Ikasketa eta laginketa erabiltzen dira prozesu hori egiteko, non laginketa EDA algoritmoen bilaketa-espazioan zehar dauden soluzio onak bilatzeko erabiltzen den.

Gradu Amaierako Lan honetan, EDA algoritmoetan erabiltzeko eredu probabilistiko berri bat proposatu eta garatu da: *Bradley-Terry* eredu probabilistikoa. Bi jokalariren arteko lehian oinarritzen da eredua, non batak besteari irabazteko probabilitateak kalkulatzen diren. Hala, eredu hori permutazio-problemetan erabili nahi izan da, permutazioko elementuen probabilitateak kontuan hartuta soluzioa lortzeko. Eredu horren errendimendua ebaluatzeko optimizazio konbinatorioko problema ezagun bat erabili da, *Linear Ordering Problem*a, alegia. Hori guztia *R project*eko **metaheuR** paketea erabiliz kodetu da, eta, inplementazioarekin, paketea luzatu da.

Ereduaren ikasketa eta laginketa prozesuak kodetu dira, bigarrena egiteko bi algoritmo proposatuz. Eredua kodetu ondoren, bi esperimentazio egin dira haren efizientzia aztertzeko. Alde batetik, laginketarako proposatutako algoritmoen portaerak alderatu dira. Bestalde, LOPa ebazteko *Bradley-Terry* ereduaren EDAren efizientzia neurtzeko esperimentazioa egin da. Horretarako, proposatutako eredua inplementatuta dagoen beste eredu probabilistiko batekin alderatu da, *Plackett-Luce* eredu probabilistikoarekin, alegia. Proposatutako ereduaren portaera ez da beste ereduaren modukoa izan LOPa ebazteko.

Txostenak ondorengo egitura dauka: lehen atalean, oinarri teorikoak azaldu dira. Bigarren atalean, LOPa ebazteko proposatutako eredua erabiltzeko arrazoia aztertu da. Hirugarren atalean, aurreikusitako plangintza deskribatu da. Laugarren atalean, eredua lagintzeko proposatutako algoritmoak azaldu dira. Bosgarren atalean, esperimentazioak definitu eta haien emaitzak aztertu dira. Seigarren kapituluan, lortutako emaitzetatik ateratako ondorioak aipatu eta etorkizuneko lanak proposatu dira. Zazpigarren kapituluan, plangintzan aurreikusitako eta errealitatean egindako lan-karga alderatu eta proiektuaren ebaluazioa egin da. Bukatzeko, inplementatutako kodearen xehetasunak aztertu dira eranskin batean.

# Aurkibidea

# Irudien aurkibidea



# Taulen aurkibidea

# 1 Sarrera

Eguneroko bizitzan askotan erabakiak hartu behar dira. Erabakiak hartzeko prozesu horretan hainbat pauso desberdindu daitezke: problema formalizatu, matematikoki modelatu eta soluziorik onena bilatu. Soluzio bat baino gehiago dagoenean, soluzio horien kalitatea neurtzeko modu bat izanez gero, soluzio onena bilatzea izango da optimizazioaren helburua.

Optimizazio-problemak aspalditik aztertu diren arren, matematika aplikatuan optimizazioa duela ez asko bereizi den ikerkuntza-arloa da. Azken hamarkadetan, adimen artifizialak, ikasketa automatikoak edota ikerketa operatiboak optimizazio-problemak modu eraginkorrean ebazteko tresna espezifikoak proposatu dituzte.

Optimizazio-problemak bi motakoak izan daitezke: soluzioak definitzeko balio errealak erabiltzen dituztenak eta balio diskretuak erabiltzen dituztenak. Lehenengoei *optimizazio jarraitu*ko problemak deritze. $\mathbb{R}^n$ azpimultzo infinitu eta ez zenbagarri bateko zenbaki errealez osatutako bektore bat bilatzea da helburua, non $n$ probleman erabilitako aldagai kopurua den. Bigarrenei *optimizazio konbinatorio*ko problemak deritze. Multzo finitu edo infinitu eta zenbagarri bateko elementu bat bilatzen da; elementua zenbaki osoz osatutako bektore bat, grafo bat, azpimultzo bat edo permutazio bat izan daitezke. Soluzioaren karakterizazioak algoritmoen garapenean duen garrantzia dela eta, optimizatzeko proposatutako metodoak (heuristikoak zein metaheuristikoak), hainbat ikerketa-lerro jarraitu dituzte.

Proiektu honetan *optimizazio konbinatorio*ko problemak aztertu dira; zehazki, multzo zabal horren parte diren permutazio-problemak.

Konbinatorian, $\sigma$ permutazio bat, $\sigma : \{1, ..., n\} \rightarrow \{1, ..., n\}$ funtzio bijektiboa bezala deskribatzen da, hau da, $\{1, ..., n\}$ multzoaren ordenazioak dira, non $\sigma(i) = j$ *i.* posizioan *j.* elementua dagoela esan nahi duen. Gainera, $\sigma^{-1}$ adierazpenak $\sigma$ permutazioaren iraulia adierazten du.

Egun, permutazioak konbinazio egitura oso hedatuak dira [26] eta grafoen teorian, psikologia matematikoan, bioinformatika arloan eta, bereziki, problema logistikoetan azaltzen dira. Testuinguru bakoitzaren arabera *ranking* edo *ordenazio* terminoekin ere ezagutzen dira permutazioak. Arazorik ez egoteko permutazio terminoa erabili da memorian. Notazioari dagokionez $\sigma$ eta $\pi$ letra greziarrak erabili dira.

Permutazioetan oinarritutako problemetan soluzioak permutazioak dira, eta, beraz, bilaketa-espazioa $n$ tamainako permutazio guztien multzoa da ($\mathbb{S}_n$). Soluzio guztien artean helburu-funtzio zehatz bat ($f : \mathbb{S}_n \longrightarrow \mathbb{R}$) maximizatzen edo minimizatzen duen permutazioa aurkitu nahi da. Dokumentuan landutako problema $f$ maximizatzean oinarritzen da, hau da, $\sigma^*$ permutazioa bilatu nahi da non $f(\sigma) \leq f(\sigma^*)$ [4]. Permutazio-problemetan hiru problema mota bereizi ohi dira: esleipen-problemak [14], antolakuntza-problemak [12] eta garraio-problemak [27].

Permutazioetan oinarritutako problemetan bilaketa-espazioa faktoriala da ($|\mathbb{S}_n| = n!$, alegia). Adibidez, $n = 10$ kasuan, bilaketa-espazioan 3628800 permutazio daude; hala, $n = 20$ den kasuan, $2 \times 10^{18}$ soluzio baino gehiago. Ondorioz, permutazio-problemek duten konplexutasun-mailaren ondorioz NP-zailen multzoan sartzen dira (*NP-hard*, ingelesez) [11]. Beste era batean esanda, algoritmo zehatzak normalean ez dira bideragarriak (problema txikietan bakarrik).

## 1.1 Aurrekariak

*NP-hard* problemen barruan sartzen da ordenazio linealeko problema edo **LOP**a (*Linear Ordering Problem*, ingelesez) [19][25]. LOPan, zenbaki osoz osatutako $\mathbf{B} = [b_{i,j}]_{n \times n}$ matrizea emanda, matrizeko errenkada eta zutabe guztiak permutatzen dira aldi berean, diagonaletik gora geratzen diren osokoen batura maximizatzeko helburuarekin. LOPen soluzioak $n$ tamainako permutazio moduan kodetzen dira, non $\sigma(i)$ permutazioko $i$ posizioko zutabearen (eta errenkaden) erreferentzia den. LOP problemaren helburu-funtzioa horrela formalizatzen da:

$$f(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{\sigma(i),\sigma(j)} \quad .$$

Problema horrek duen konplexutasuna dela eta, soluzio optimoa topatzea ezinezkoa da normalean. Horregatik, ahalik eta soluzio onenak bilatzea izan ohi da helburua. Ildo horri jarraituz sortu ziren algoritmo heuristikoak. Algoritmo heuristikoak intuizioan oinarritutako metodoak dira, eta problema bakar baterako diseinatzen dira, ahalik eta emaitza onena lortzeko. Metodo heuristikoak oso interesgarriak izan arren, zailak dira beste problemetan berrerabiltzeko, ez baitira orokorrak. Birziklatze prozesu hori ahalbidetzeko daude algoritmo metaheuristikoak [24]. Horiek optimizazio-prozeduraren intuizioan oinarritzen dira, eta, horregatik, edozein problema ebazteko egoki daitezke. Adimen artifizialeko komunitateak soluzio onargarriak bilatzeko algoritmo metaheuristiko asko proposatu ditu, hala nola, *local search* (bilaketa lokala) , *tabu search* (tabu bilaketa), *simulated annealing* (suberaketa simulatua), *genetic algorithm* (algoritmo genetikoak) eta *Estimation of Distribution Algorithms* (EDA algoritmoak)[15][16][20][21]. Proiektuan azken algoritmo horiek landu dira.

EDA algoritmoak, konbinatoria-problemak ebazteko indar handia duen konputazio ebolutiboan oinarritutako algoritmoak dira. Algoritmoetan horietan, soluzio (edo indibiduo) multzo bat izanda (populazioa, alegia), prozesu iteratibo baten bidez populazioa eboluzionarazten da. Prozesuan hiru urrats daude: lehenengoan, soluzio onenak aukeratzen dira eta eredu probabilistiko bat ikasten da. Ondoren, eredua lagintzen da soluzio berriak sortzeko. Hirugarren urratsean, soluzio zaharren eta berrien artean onenak aukeratzen dira populazio berria sortzeko. Oinarrizko EDAren sasikodea 1. algoritmoan ikus daiteke.

**1. algoritmoa** EDA algoritmoaren sasikodea

1: $D_0 \leftarrow$ Ausaz $M$ indibiduo sortu (hasierako populazioa)
2: **for** $t = 1, 2, ...$ gelditze irizpidea bete arte **do**
3: $\quad D_{t-1}^{Se} \leftarrow$ Aukeraketa-metodoaren arabera $D_{t-1}$ populaziotik $N \leq M$ indibiduo aukeratu
4: $\quad p_t(x) = p(x|D_{t-1}^{Se}) \leftarrow$ Aukeratutako indibiduoen eredu probabilistikoa balioztatu
5: $\quad D_M \leftarrow M$ indibiduo lagindu (populazio berria) $p_t(x)$ erabiliz
6: $\quad D_t \leftarrow D_{t-1}$ eta $D_M$ indibiduoekin populazio berria sortu
7: **end for**

EDA algoritmoen esentzia, beraz, eredu probabilistikoa da. Ildo horretan, ereduak soluzio bakoitzari probabilitate bat esleituko dionez, soluzioen

kodeketa, eredua diseinatzeko orduan, puntu kritikoa da. Hala ere, soluzio-kodeketa batzuk eredu probabilistikoa definitzeko zailtasunak dituzte, permutazioak, esate baterako. EDA algoritmoentzako benetako erronka dira permutazioetan oinarritutako problemak, domeinu diskretuko edo jarraituko probabilitate banaketa klasikoak ezin baitira modu eraginkorrean egokitu.

Populazioa, permutazio multzo bat denean, posizio bakoitzeko bazter-probabilitateak bektore kategorikoetan bezala estimatzea posible da; baina, ondoren, eredua lagintzean, ez dira halabeharrez permutazioak lortuko, balio errepikatuak ager baitaitezke. Arazo hori saihesteko, soluzio berriak lagintzean, permutazioek duten murriztapena aintzat hartu behar da; baina hori egitean ikasitako eredua aldatzen da, algoritmoaren filosofia zapuztuz.

Permutazio-problemei soluzioa bilatzeko dauden arazoengatik eta permutazioen eredu probabilistikoak permutazio-espazioak modelatzeko dagoen eraginarengatik aurrerapen ugari ikertu dira EDA algoritmoen eraginkortasuna hobetzeko. Azken urteotan permutazioen domeinuko egokiak diren eredu probabilistikoak erabiltzea proposatu da, horien artean *Mallows*, *Generalized Mallows*, *Plackett-Luce* eta *Bradley-Terry*.

*Mallows model* [18] distantzian oinarritutako probabilitate-eredu esponentziala da. Permutazioen arteko $D$ distantzia, $\sigma_0$ permutazio zentrala eta $\theta$ dispertsio-parametroa dira ereduak erabiltzen dituen parametroak. Permutazioen arteko distantziekin lortutako probabilitateak horrela kalkulatzen dira:

$$P(\sigma) = \frac{1}{\psi(\theta)} e^{-\theta D(\sigma,\sigma_0)} \quad ,$$

non $\psi(\theta)$ normalizazio-konstantea den. $\theta > 0$ denean, $\sigma_0$ permutazio zentrala probabilitate altuenekoa da, eta hurrengo $n! - 1$ elementuen probabilitateak permutazio zentralaren distantziarekiko esponentzialki gutxiagotzen dira.

Normalean eredua *Kendall*en $\tau$ distantziarekin konbinatzen den arren, *Cayley*, *Ulam*, *Hamming* edo *Spearman* [7][10] distantziekin ere konbina daiteke.

Aurreko ereduaren orokortasun bezala *Generalized Mallows model* [10] proposatu da, baina, kasu honetan, distantzia $n-1$ elementutan deskonposatzen da, $S_j(\sigma, \sigma_0)$. Horrela, dispertsio-parametro bakarra erabili beharrean $\boldsymbol{\theta} = (\theta_1, ..., \theta_{n-1})$ diren $n-1$ dispertsio-parametro erabiltzen dira, eta bakoitzak soluzioko elementu bakarrari eragiten dio.

*Plackett-Luce* [22][17] eredua *Thurstone* motako eredu estatistikoen barruan sailkatzen da. Eredu horretan elementu onenen probabilitatea altua izatea eta elementu txarrenen probabilitatea txikia izatea bilatzen da. Aldibereko ordenazio baten bitartez kokatzen dira elementuak permutazioan, haien pisuak kontuan izanda. *Plackett-Luce*aren eredua honela definitzen da:

$$P(\sigma|\boldsymbol{w}) = \prod_{i=1}^{n-1} \frac{w_{\sigma(i)}}{\sum_{j=i}^{n} w_{\sigma(j)}} \quad ,$$

non $\boldsymbol{w}$ pisuen bektorea den. Definizioaren arabera, $\sigma(i)$ elementua $i$. posizioan hautatua izateko probabilitatea $j = i, .., n$ posizioetan dauden elementuen pisuen menpekoa da.

Aurreko hiru ereduak modu eraginkorrean aplikatu dira EDA algoritmoetan, baina badago beste antzeko eredu bat: *Bradley-Terry* eredua [9][3][13]. Eredu hori bi jokalarien arteko lehiakortasunean oinarritzen da. Jokalari batek bestea irabazteko probabilitatea jokalarien trebetasunarekin zerikusia dauka. Permutazioekin lan egin denez, elementu batek bestea irabazteko kontzeptua argitu behar da. Elementuen posizioarekin erlazioa dauka, non permutazioko $a$ elementua $b$ elementua baino lehen azaltzean $a$-k $b$ irabazi duela esaten den. Aipatutako elementuen irabazteko probabilitatea honela kalkulatzen da:

$$P(a \prec b) = \frac{w_a}{w_a + w_b} \quad ,$$

non $w_a$ eta $w_b$, $a$ eta $b$ elementuekin zerikusia duten pisu positiboak diren. $w_a$ balioa $w_b$ balioaren gain zenbat eta handiagoa izan, $a$ elementua $b$ elementuaren aurretik egoteko probabilitatea ere handiagoa izango da.

$\sigma$ permutazio bat eta $w = (w_1, ..., w_n)$ pisuak emanik, permutazioaren probabilitatea honela kalkula daiteke [6]:

$$P(\sigma) \propto \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} p_{(i \prec j)} \quad .$$

*Bradley-Terry* eredua EDA algoritmoetan erabili ahal izateko ikasketa eta laginketa metodo eraginkorrak diseinatu behar dira. Zentzu horretan, ikasketa-prozesuak eredu linealen oinarrituriko parametroak estimatzen ditu [23]. $n > 1$ tamainako permutazioek $\sum_{i=n-1}^{1} i$ konparazio egiten dituzte. Adibidez, $n = 4$ tamainako permutazioek 6 konparazio egin behar dituzte: 1-2, 1-3, 1-4, 2-3, 2-4 eta 3-4 (elementuen konparazioa).

Laginketari dagokionez, literaturan ez da oinarrizko metodorik proposatu. Hori dela eta, proiektu honetan bi irtenbide proposatu dira: *Metropolis-Hasting* algoritmoa [29] eta algoritmo heuristiko bat. Biak *4.* atalean azaldu dira.

# 2 Motibazioa

Aurreko kapituluan, eredu probabilistikoa EDA algoritmoen gakoa dela aipatu da. Ildo beretik, permutazio-problemetan erabili daitezkeen hainbat eredu azaldu dira. Noski, problema guztiek ez dituzte egitura berak, eta, era berean, eredu probabilistikoek ere murriztapen eta ezaugarri desberdinak dituzte. Hori dela eta, problema bat emanik, hori optimizatzeko egokiena den eredua aukeratzea ezinbestekoa izango da emaitza onak lortu nahi baldin badira.

Kapituluan LOP problemaren deskripzio sakona burutu da, eta, ondoren, *Bradley-Terry* ereduarekin konbinatzeko arrazoiak azaldu dira.

Hemendik aurrera $\sigma, \pi \in \mathbb{S}_n$ permutazioak izango dira, eta $e$ identitate permutazioa (123...$n$, $n$ permutazioaren tamaina izanda). Hori horrela, $i, j$ eta $z$ aldagaiak permutazioko posizioa adierazteko erabili dira; $k$ eta $l$ balioak permutazioko elementuei erreferentzia egiteko erabili dira.

## 2.1 LOP eta *Bradley-Terry*

Aurreko atalean azaldu den moduan, LOPan, $\mathbf{B} = [b_{k,l}]_{n \times n}$ zenbaki-matrizea emanda, $\mathbf{B}$ matrizeko errenkada eta zutabe guztiak permutatzen dira aldi berean diagonaletik gora geratzen diren osokoen batura maximizatzeko helburuarekin. Soluzioak $n$ tamainako permutazio moduan kodetzen dira, non $\sigma(i)$ permutazioko $i$ posizioko zutabearen (eta errenkadaren) erreferentzia den.

Jarraian, $\mathbf{B}$ matrizeko $b_{k,l}$ elementuen eta $\sigma$ permutazioaren arteko erlazioa ulertzen lagunduko duen hainbat zehaztapen eman dira:

1. $\sigma(i) = k, i = 1, ..., n$ elementuak $2(n-1)$ elementu erlazionatzen ditu ($k$ errenkadako $n-1$ elementu eta $k$ zutabeko beste $n-1$ elementu, $b_{i,i}$ ez da kontuan hartzen diagonal nagusian baitago). Adibidez, 1a irudian ikusi daiteke $\sigma(2)$ elementuak *2* zutabeko 16, 23, 22 eta 28 balioak eta *2* errenkadako 21, 14, 15 eta 9 balioak erlazionatzen dituela.

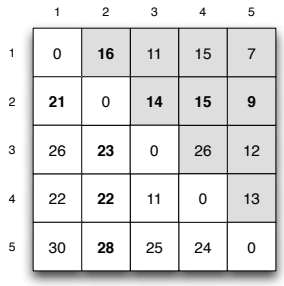
2. $\sigma(i) = k, i = 1, ..., n$ osatzen duten elementuak bikoteka multzoka daitezke. Izan ere, $k$ errenkadako edozein elementuk $k$ zutabean diagonal nagusiarekiko elementu simetrikoa dauka, hau da, $b_{k,\sigma(j)}$ eta $b_{\sigma(j),k}$ elementuak simetrikoak dira, non $j = 1, ..., n$. 1c irudian borobilduta ikus daitezke *2* errenkadako eta *2* zutabeko balio simetrikoak.

3. $\sigma(i) = k$ elementuak ez du esleitutako $\{b_{k,1}, b_{1,k}\}, ..., \{b_{k,n}, b_{n,k}\}$ bikoteen diagonalarekiko posizioaren berririk.

4. $b_{\sigma(i),\sigma(j)}$ guztiek bi elementuri erreferentzia egiten diete, $\sigma(i)$ eta $\sigma(j)$. Adibidez, 1a irudiko $b_{2,3} = 14$ balioa *2* errenkadan eta *3* zutabean kontuan hartzen da.

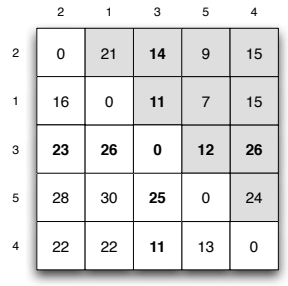
5. $\{b_{\sigma(i),\sigma(j)}, b_{\sigma(j),\sigma(i)}\}$ bikote guztietan parametro bat diagonal nagusiaren gainetik eta bestea azpitik dute. Kasu honetan ere 1c irudian borobilduta ikus daitezke balio simetrikoen posizioak.

Ezaugarriak ikusi ondoren permutazioko elementuen ekarpenen inguruan hitz egin da. $\sigma(i) = k$ elementuaren ekarpena diagonal nagusiaren gainetik



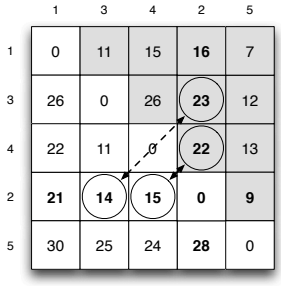
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | **16** | 11 | 15 | 7 |
| 2 | **21** | 0 | **14** | **15** | **9** |
| 3 | 26 | **23** | 0 | 26 | 12 |
| 4 | 22 | **22** | 11 | 0 | 13 |
| 5 | 30 | **28** | 25 | 24 | 0 |

**(a)** $e = 1\mathbf{2}345$,
$c(e,2) = 54$,
$c(e,3) = 63$,
$c(e,4) = 69$



| | 2 | 1 | 3 | 5 | 4 |
|---|---|---|---|---|---|
| 2 | 0 | 21 | **14** | 9 | 15 |
| 1 | 16 | 0 | **11** | 7 | 15 |
| 3 | **23** | **26** | **0** | **12** | **26** |
| 5 | 28 | 30 | **25** | 0 | 24 |
| 4 | 22 | 22 | **11** | 13 | 0 |

**(b)** $\pi = 21345$,
$c(\pi,2) = 59$,
$c(\pi,3) = 63$,
$c(\pi,4) = 80$



| | 1 | 3 | 4 | 2 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 11 | 15 | **16** | 7 |
| 3 | 26 | 0 | 26 | **23** | 12 |
| 4 | 22 | 11 | 0 | **22** | 13 |
| 2 | **21** | **14** | **15** | **0** | **9** |
| 5 | 30 | 25 | 24 | **28** | 0 |

**(c)** $\sigma = 134\mathbf{2}5$,
$c(\sigma,2) = 70$,
$c(\sigma,3) = 72$,
$c(\sigma,4) = 76$

**1. irudia:** Elementuen aldaketak ekarpenean duen eragina.

dauden $k$ ilarako eta zutabeko balioen batura maximizatzean datza. Hurrengo ekuazioarekin definitzen da $k$ elementuak helburu-funtzioan duen ekarpena:

$$c(\sigma, k) = \sum_{j=1}^{k-1} b_{\sigma(j),\sigma(i)} + \sum_{j=k+1}^{n} b_{\sigma(i),\sigma(j)} \quad .$$

$f$ helburu-funtzioari dagokionez, $c(\sigma, k)$ ekarpena aurreko $\{\sigma(1), ..., \sigma(i-1)\}$ eta hurrengo $\{\sigma(i+1), ..., \sigma(n)\}$ elementuek duten banaketak ezarriko du. Aldiz, aurretik dauden elementuen ordenak ez du eraginik, beti ere $i$. posizioa baino lehen kokatzen badira (eta berdina $i$.posizioaren ondorengo elementuekin); 1a eta 1b irudietan garbi ikusten da. Adibide horretan, 3 elementuaren aurreko eta ondorengo elementuen ordenak elkarrekin aldatu dira, eta ikusi daiteke aldaketek ez dutela 3 elementuaren ekarpena aldatu.

Hala ere, suposatuz $\sigma(j) = l$ elementua $k$ elementuaren aurretik dagoela, eragina al dauka permutazioan $l$ elementua $k$ elementuaren ondoren jartzea? Erantzuna baiezkoa da, $c(\sigma, i)$ aldatzen baita; hori dela eta, aurretik esan dena hedatu behar da.

LOP problemaren kontzeptuetan azaldu den moduan, $\{b_{\sigma(i),\sigma(j)}, b_{\sigma(j),\sigma(i)}\}$ bikote guztiek bi elementuei erreferentzia egiten diete, $k$ eta $l$ elementuei, alegia. Izan ere, $\sigma(i)$ eta $\sigma(j)$ elementuen artean egiten diren trukeak, bi elementuek helburu-funtzioan duten ekarpenean eragina du. Hala, $\sigma(i)$ $j$ posiziora mugitzeak, $i$ eta $j$ posizioen arteko elementuen ekarpeneko aldaketak izatea dakar. 1a eta 1c irudietan adibide bat ikus daiteke.

Xehetasun bezala, $i$ eta $j$ posizioen arteko elementuek ekarpenean izandako diferentziaren baturak finkatzen du $c(\sigma(i))$ berria. 1a eta 1c irudietako adibidean, $c(e, 3) = 63$ izatetik 72 izatera pasatzen da, bien arteko diferentzia 9 izanik. Halaber, $c(e, 4) = 69$ izatetik 76 izatera pasatzen da, bien arteko diferentzia 7 izanik. Hortaz, $9 + 7 = 16$ balioko diferentzia dago, hain zuzen ere, $c(e, 2)$ ekarpenen artean. Are gehiago, $\sigma(i)$ elementua $\sigma(j)$ elementuaren aurretik jartzearen ondorioz $b_{\sigma(i),\sigma(j)}$ parametroa $f$ helburu-funtzioaren kalkuluan erabiliko da, eta $b_{\sigma(j),\sigma(i)}$ ez, ordea.

LOParen ezaugarriak zehaztuta *Bradley-Terry* eredua erabiltzea egokia izan daitekeela pentsatu da. *Bradley-Terry* ereduaren ezaugarriak gogorarazteko, $\sigma$ permutazioaren probabilitatea elementuen arteko alderaketarekin kalkulatzen da. Alderaketa bi elementuren artean egiten da, eta permutazioan duten ordenak finkatzen du haien pisua. Hurrengo ekuazioa erabiltzen da horretarako:

$$P_{(i,j)} = \frac{w_i}{w_i + w_j} \quad ,$$

non $w_i$ eta $w_j$, $i$ eta $j$ elementuen pisu positiboak diren. Horrela, ba, intuizioz pentsa daiteke *Bradley-Terry* eredua egokia dela LOParen soluzioak modelatzeko. Planteamendu hori da hain zuzen ere gradu amaierako lan hau motibatu duena.

# 3 Proiektuaren kudeaketa plana

Atal honetan, Gradu Amaierako Proiektua burutzeko egindako kudeaketa plana azalduko da. Helburuak eta lan-paketeak azalduko dira, baita denboraren eta arriskuen kudeaketa-plana ere (*Gantt* diagrama eta LDE diagramak erabiliz).

## 3.1 Irismena

Ondoren proiektuaren helburuak, helburu horien kanpo geratutako jarduerak, lan-paketeen deskribapena eta emangarriak azalduko dira.

### 3.1.1 Proiektuaren helburuak

Esan bezala, permutazioetan oinarritutako optimizazio-problemen azterketan oinarrituko da proiektua, LOP problemetan zehazki. Hori horrela, bi helburu nagusi finkatuko dira:

- *Bradley-Terry* eredua lagintzeko metodoak proposatzea eta aztertzea.
- *Bradley-Terry* eredua EDA paradigman sartzea eta haren funtzionamendua LOP problemetan aztertzea.

Helburu horiez gain, helburu pertsonal bat ere ezarri da:

- Zientzia munduko ikerlan bat zer den ikastea (etorkizunerako interesa suspertzeko helburuarekin aukeratutako zientzia-proiektua da eta).

### 3.1.2 Irismenetik kanpo

Erabilitako R paketea luzatzea helburu nagusia ez izan arren, ondorengo lanetarako erabilgarria izatea bilatzen da. Rko klasea sortu beharra dago EDA algoritmoekin lan egiteko.

### 3.1.3 LDE diagrama eta lan-paketeen deskribapena

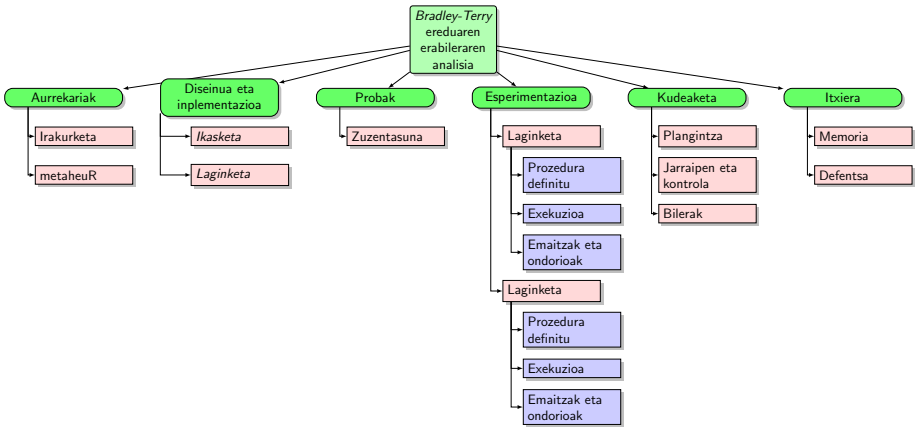
2. irudian, 6 ataletan banatuta, egin beharreko lanaren eskema ikusten da. Jarraian, atal bakoitzaren betekizunak aztertuko dira.

**Aurrekari**en barruan dagoen irakurketa atalak proiektua ulertzeko irakurri beharreko artikuluei erreferentzia egiten dio. 6 artikulu irakurriko diren arren [9][5][13][8][3][6], azpiataletan ez banatzea erabaki da. Bestalde, **metaheuR** [2] atalak R programazio lengoaiako paketea aztertzeari erreferentzia egiten dio. Horretarako, Bilaketa Heuristikoak ikasgaian erabilitako materiala berrerabiliko da (liburua [1], kodea...).

**Diseinu eta inplementazio** atala bi azpiataletan banatzen da: ikasketa eta laginketa. Lehen azpiatalak, R programazio lengoaiako **BradleyTerry2** paketea [28] erabiliz egindako prozesua azaltzen du. Bigarrenak, laginketa egiteko proposatutako bi algoritmoak kodetzeari erreferentzia egiten dio.

**Probak** atalean dagoen zuzentasun azpiatalak aurretik aipatutako prozesuen funtzionamendua egokia dela ziurtatzen du.

**Esperimentazioa**ren barruan bi atal banatu dira: laginketa-prozesuko datuen berreskurapena eta LOP problemak ebazteko EDA algoritmoen exekuzioa.




**2. irudia:** Lanen Deskonposaketa Ereduaren diagrama.

Hala ere, bi atal horiek hiru azpiatal berdin dituzte. Hiru atalak prozedurak definitu, exekuzioa eta emaitzak eta ondorioak dira. Prozedura definitu azpiatala egin beharreko prozesua definitzean datza. Exekuzioan, izenak dioen bezala, definitutako prozeduren exekuzioa sartzen da. EDA algoritmoen kasuan tutoreek pasatako instantziekin lan egingo da. Exekuzioen emaitzen analisiaren ondorioak azalduko dira 6. atalean.

**Kudeaketa** atalean proiektua aurrera eramateko beharrezko prozesuak azaltzen dira. Bertan plangintza, jarraipen eta kontrola eta bilerak bereizi daitezke. Bakoitzaren definizioa oso garbi dago. Aurreikuspen batetik egin beharreko lanaren jarraipen prozesuan oinarritzen da atala.

Seigarren eta azken atalean proiektuaren **itxiera** prozesuari erreferentzia egiten zaio. Bertan, egindako lan guztia jasotzen duen memoria eta tribunalaren aurrean azaldu beharreko aurkezpena azalduko dira.

### 3.1.4 Emangarriak

Proiektua aurreratu heinean emaitzak aztertzeko entregak egingo dira. Entregak tutoreekin ebaluatuko eta ondorioak aterako dira. Erabakiak alda litezke lortutako emaitzen arabera.

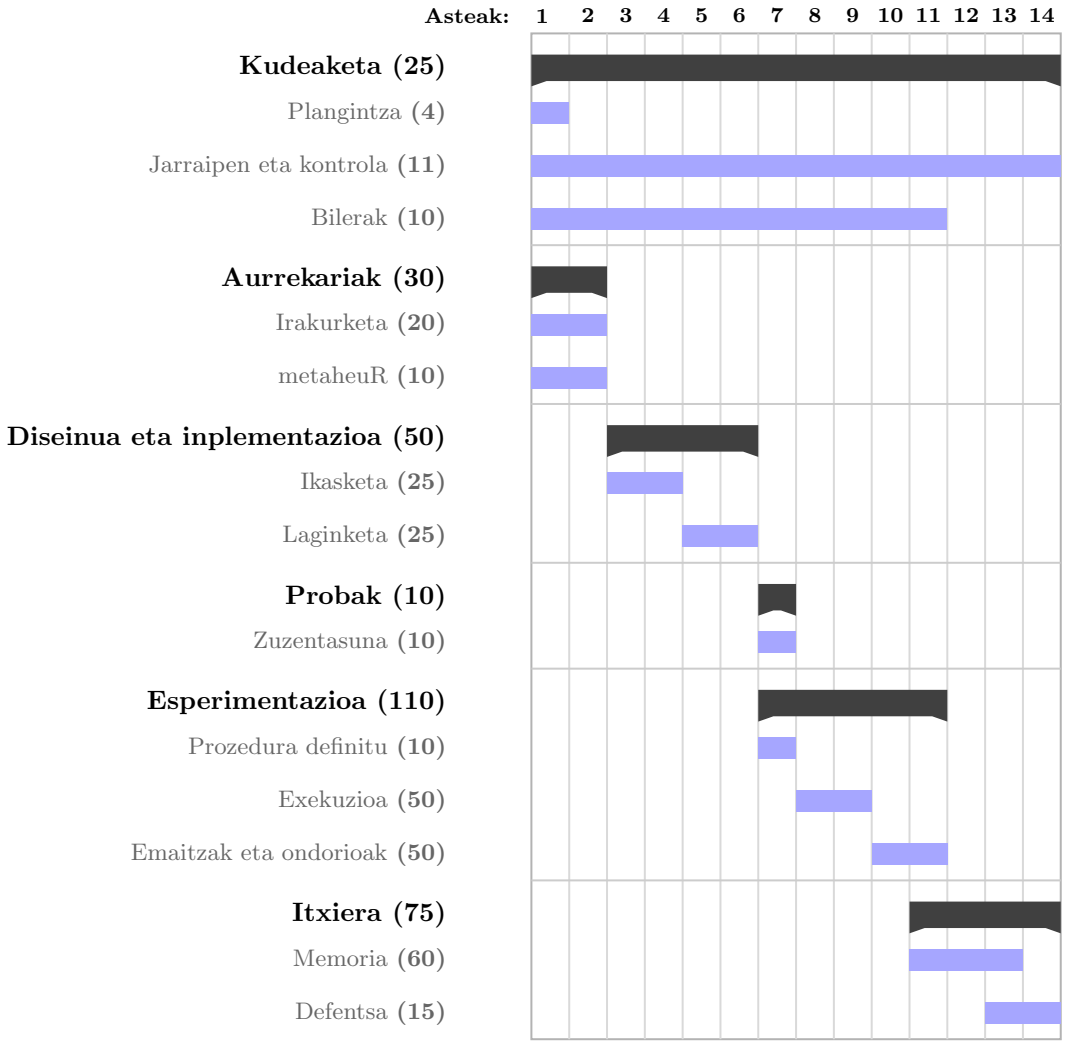
Horrez gain, proiektuan zehar egindako lan guztia txosten batean adieraziko da. Informazioaren laburpena, erabilitako prozeduren sasikodea (ez kode gordina), emaitzen analisia, etab. azalduko dira, atal bakoitzari kapitulu bat eskainiz.

Jabego Intelektualaren Legeak babesten du Gradu Amaierako Proiektua. Hortaz, memoriaren eta defentsa-aurkezpenaren eskubide guztiak erreserbatuta daude.

## 3.2 Denbora-kudeaketa

Proiektua egiteko hartu beharreko denboraren planifikazioa azaldu da hemen. Horretarako, lan bakoitzaren denbora-estimazioa eta mugarriak azalduko dira *Gantt* diagrama batean (3. irudia). Dokumentuaren amaieran

(7. atalean) xehetasun gehiagorekin azalduko da planifikazio orokorraren dedikazioa eta errealitatean egindakoaren arteko diferentzia.




**3. irudia:** *Gantt* diagrama. Parentesi artekoa atal zeregin bakoitzari eskaini beharreko denboraren aurreikuspena da.

### 3.2.1 Lan-paketeen dedikazioa

*Gantt* diagraman (3. irudia) entregak eta bukatze-datak ez dira egun batekin zehaztu, hau da, astetan banatu dira atalak. Memoriaren entrega mugarri nagusia da, eta hori egiteko egindako prozesuen datak ez dira finkatu, ordena baizik.

Proiektua osatzeko 300 ordu aurreikusten dira. Ordu horien distribuzioa 3. irudian parentesi artean azaltzen den dedikazioarekin osatuko dira. Proiektu osoa egiteko estimatutako denbora 14 astekoa da. Aste bakoitzean sartu be-

harreko lan karga ez da berdina izango, atal batzuk besteak baino errazagoak izatea aurreikusten baita.

## 3.3 Arriskuak

Ezustekoak azalduko dira atal honetan, baita horiek gertatzekotan egin beharrekoak ere. Bi ezusteko nagusi aurkezten dira:

- Laginketa-prozesua proposatzeko behar baino denbora gehiago pasatzea. Konponbidea emateko lana egin heinean erabakiko aa zer egin. Behar baino denbora gehiago hartzekotan esperimentazioa eta proiektuaren helburuak birfinkatuko lirateke; alderantziz, behar baino denbora gutxiago hartzekotan, esperimentazioan sakonduko litzateke proiektua.

- Egindako lan-prozesua galtzea. Lan-prozesua modu lokalean eta interneten gordetzea erabaki da, eskuragarritasuna hobetzeko asmoz. Horretarako *Github* plataforma erabiliko da kodea kudeatzeko, eta *Overleaf* plataforma memoriarako.

# 4 *Bradley-Terry* ereduaren laginketa

EDA algoritmoen mamia eredu probabilistikoak direla esan da aurreko ataletan. Ereduak erabiltzeko ikasketa eta laginketa metodo eraginkorrak diseinatu behar dira.

Atal honetan, proposatutako *Bradley-Terry* eredua lagintzeko erabilitako metodoak garatu dira haien egitura, sasikodea eta parametroak zehaztuz. Bi algoritmo proposatu dira: *Metropolis-Hasting* algoritmoa eta algoritmo heuristiko bat.

## 4.1 *Metropolis-Hasting* algoritmoa

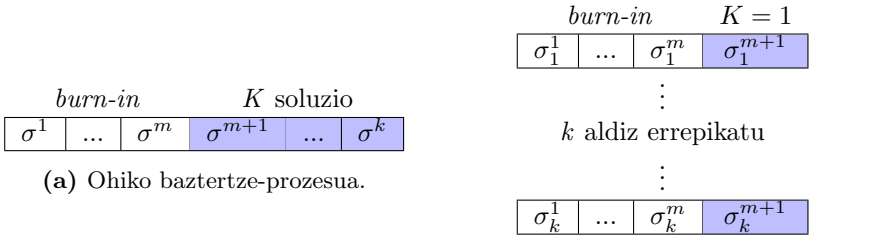
*Metropolis-Hasting* algoritmoa [29] banaketa probabilistikoen laginketarako erabili ohi den prozedura orokorra da. Eredu probabilistikoan oinarrituz, ausazko permutazio batetik hasita, iteratiboki permutazioan aldaketak egiten dira eta permutazio berria probabilitatearen arabera aukeratzen da. Hiru sarrera-parametro ditu algoritmoak: $P$ eredua, $K$ soluzio kopurua eta *burn-in* deritzon parametroa. Proiektu honetan *Bradley-Terry* eredua definitu da sarrera-parametro gisa. *Metropolis-Hasting* algoritmoaren sasikodea 2. algoritmoan ikus daiteke.

**2. algoritmoa** *Metropolis-Hasting* algoritmoaren sasikodea

1: $\sigma^1 \leftarrow$ $n$ tamainako ausazko permutazioa sortu
2: **for** $t = 1, 2, ...$ *burn-in* $+ K$ balioa arte **do**
3: $\quad \sigma' \leftarrow \sigma^t$ren bi ausazko balio trukatu
4: $\quad$ **if** $min\{\frac{P_{\sigma'}}{P_{\sigma^t}}, 1\}$ probabilitatearekin **then**
5: $\quad\quad \sigma^{t+1} \leftarrow \sigma'$
6: $\quad$ **else**
7: $\quad\quad \sigma^{t+1} \leftarrow \sigma^t$
8: $\quad$ **end if**
9: **end for**
10: **return** $\sigma^{burn\text{-}in+1}, ..., \sigma^{burn\text{-}in+K+1}$

Esan bezala, ausazko permutazio bat da algoritmoaren abiapuntua. 2. algoritmoan ikus daitekeen moduan, *burn-in* aldagaiak hasierako permutaziotik egin beharreko aldaketa kopurua adierazten du, non aldaketak ausaz aukeratutako bi elementuen artean egiten diren. Prozesu horrekin $P$ distribuziora konbergitzen da, baina hasierako permutazioak ezin dira distribuzioaren lagintzat hartu. Hori dela eta, sekuentziako lehenengo permutazioak baztertzen dira, *burn-in* lehenengo permutazioak, alegia.

Beraz, hasierako permutazioak alde batera utzi ondoren dauden hurrengo $K$ permutazioak lagintzat hartzen dira. Hala ere, algoritmoak, horrela definituta, arazo bat du: laginketa tamaina txikia denean lagindutako permutazioak oso antzekoak dira. Algoritmo ebolutiboetan hori arazo handi bat da, populazioaren dibertsitatea murrizten baita. Arazo hori ekiditeko prozesua aldatu da. $K$ indibiduo sortzeko ($K$ tamainako laginketa bat lortzeko, alegia) prozedura $k$ aldiz errepikatzen da, eta iterazio bakoitzeko $K = 1$ finkatu da (soluzio bakarra hartuz iterazio bakoitzeko). 4. irudian proposatutako aldaketa nabari ikus daiteke.




**(a)** Ohiko baztertze-prozesua.

**(b)** Proposatutako baztertze-prozesua

**4. irudia:** *Metropolis-Hasting* algoritmoaren baztertze-prozesua. $\sigma^1$ hasierako permutazioa, $m$ =*burn-in* eta $k$ lortu beharreko lagin kopurua dira.

Egia da konputazionalki garestiagoa dela, baina laginketaren dibertsitate hobea lortzen dira.

## 4.2 Algoritmo Heuristikoa

Jakin denez, *Bradley-Terry* eredua lagintzeko proposatzen den lehenengo algoritmo heuristikoa da hau. Proposatutako algoritmoak elementuak banan-bana $S$ bektorean txertatuz permutazioa eraikitzen du, betiere ereduak definitzen dituen probabilitateak kontuan hartuta. Hortaz, unean bektorea osatzen duten elementuek eta haien distribuzioak eragina dute elementu berria txertatzean.

**3. algoritmoa** Algoritmo heuristikoaren sasikodea

```
1: S ← 1..n bitarteko ausazko elementu bat aukeratu
2: while txertatu gabeko elementuak daude do
3:     k ← Sn ez daude {1, ..., n} elementuen artean ausaz bat aukeratu
4:     z ← Sko posizio bat ausaz aukeratu
5:     while gelditze-irizpidea bete arte do
6:         if w_k / (w_k + w_S(z)) probabilitatearekin then
7:             z ← z − 1
8:         else
9:             z ← z + 1
10:        end if
11:    end while
12:    S ← Txertatu k z posizioan
13: end while
14: σ ← S
15: return σ
```

3. algoritmoan ikusi daitekeen moduan, hasieran elementu bakarra dago $S$ bektorean, elementu guztien artean ausaz hautatutakoa. Bektorean elementu guztiak sartu arte, hurrengo prozesua jarraitzen da: gelditzen diren elementuen artean ausaz $k$ elementua hautatu eta $S$ bektorean sartu. Txertatzeko posizioa aurkitzeko, uneko bektoreko $z$ posizio bat ausaz aukeratzen da, eta hortik abiatuta, $S$ bektoreko elementuekin alderatzen da.

Alderatze-prozesua egiteko *Bradley-Terry* eredua erabiltzen da. *Bradley-Terry* ereduak $k$ elementua eta $S(z)$ elementuaren aurretik egoteko

$P_{(k,S(z))} = \frac{w_k}{w_k + w_{S(z)}}$ kodetzen du. Horrekin, konparazioetan erabiltzeko probabilitatea kalkulatzen da, hau da, aurretik edo ondoren jartzeko erabakia hartzeko.

Hala ere, algoritmoan azaltzen den parametro bat definitu beharra dago algoritmoa begizta infinitu batean ez sartzeko; parametro hori gelditze-irizpidea da. Txertatu beharreko elementua bektoreko elementuekin behin eta berriz ez alderatzeko pentsatuta dago. Horrela, bi elementuren artean txertatuko da $k$ elementua baldin eta bi elementuak aurretik konparatu diren. Badaude bi kasu hori betetzen ez dena: $S$ bektoreko lehen elementuaren aurretik edo azken elementuaren ondoren txertatzea. Kasu horietan ez dago arazorik elementua txertatzeko.

Laburbilduz, elementuak banaka txertatzen dira bektorean. Bektorean sartu ez diren elementuen artean bat hautatu ondoren bektoreko posizio bat ausaz aukeratzen da. Txertatu nahi den elementuaren eta bektorean dauden elementuen arteko probabilitateak kalkulatuz elementua bektorean gehitzea da helburua. Txertatzeko posizioa probabilitatearen arabera mugitzen da, eta alderaketak egiten bukatu ondoren gehitzen da elementua bektorean. Elementu guztiak bektorean txertatu ondoren bektorea permutaziotzat hartzen da.

Hona hemen adibide bat modu eskematiko batean. Suposatuz $n = 5$ tamainako permutazioekin lan egin dela:

- {1,...,5} bitarteko elementu bat ausaz aukeratu, 3 elementua, adibidez, eta $S$ bektorean txertatzen da, $S =$ ( 3 ).
- Txertatu gabeko elementuen artean bat ausaz aukeratzen da, 2 elementua, eta $S$ko posizio bat ere ausaz aukeratzen da (kasu honetan aukera bakarra dago, $z = 1$).
  - 2 eta $S(1)$ elementuen arteko probabilitatea kalkulatuta, $P_{(2,S(1))}$, 2 elementua $S$ bektorean txertatzen da. Suposatuz 3 elementuaren atzean txertatu behar dela, $S=$ ( 3,2 ).
- Oraingoan, 1 elementua txertatu nahi da, eta kasu honetan ere $z = 1$ aukeratzen da (biak ausaz hautatuta).
  - Suposatuz $P_{(1,S(1))}$ probabilitatearekin $S(1)$ elementuaren atzean txertatu behar da, baina $S(2)$ elementuarekin ere konparatu behar da, elementu horren atzean edo aurrean txertatzeko; hau da, $S=$ ( 3,1,2 ) edo $S=$ ( 3,2,1 ).
  - $P_{(1,S(2))}$ probabilitatearekin $S(2)$ elementuaren aurretik txertatzen da elementua $S$ bektorean, $S=$ ( 3,1,2 ).

Bost elementuak bektorean txertatu arte errepikatzen da prozesua.

# 5 Esperimentazioa eta emaitzak

Aurreko ataletan LOPa, EDA algorimoak, *Bradley-Terry* eredua eta laginketarako erabilitako bi algoritmoak azaldu dira. Atal honetan, diseinatutako eta inplementatutako prozeduren zuzentasuna eta errendimendua aztertu dira. Horretarako, hiru azpiataletan banatu da atala, lehena probak egiteko eta beste biak esperimentaziorako.

Hasteko, kodetutako *Bradley-Terry* ereduaren zuzentasuna frogatu da. Bertan, ikasketa- eta laginketa-prozeduran arazorik ez dagoela eta lortutako emaitzak logikoak direla ziurtatu da. Ondoren, laginketarako proposatutako bi algoritmoak kasu desberdinetan exekutatu dira eta lortutako emaitzen grafikoak komentatu dira. Atala ixteko, EDA algoritmoen ingurunearen antolaketa azaldu da eta *Bradley-Terry* ereduak EDA algoritmoen ingurunean duen errendimendua azaldu da. Grafikoak lortutako emaitzak biataratzeko erabili dira.

Inplementazioaren informazio zehatzagoa nahi izanez gero jo A. eranskinera.

## 5.1 Zuzentasunaren proba eta berreskuratze-prozesua

Kodetutako ereduaren ikasketa- eta laginketa-prozesuen zuzentasuna frogatzeko besterik ez da egin proba hau. Horretarako, adibide garbi bat garatu da R *script* batean.

Ikasketa-prozesua probatzeko aipatutako *script*ean $n = 5$ tamainako identitate permutazioez osatutako 100 tamainako populazio bat sortu da. Horrela, eredua ikastean, espero zen moduan, muturreko pisuak lortu dira, laginketako emaitzak identitate permutazioaren egitura izateko asmoz. Izan ere, muturreko pisuekin identitate permutazioaren probabilitatea ia 1ekoa izatea lortzen da.

Eredua ikasi eta pisuak kalkulatu ondoren laginketa-prozesuaren zuzentasuna aztertu da. Lehenengo, *Metropolis-Hasting* algoritmoaren funtzionamendua egokia dela probatu da. Horretarako, $burn - in = 1000$ balioarekin bost lagin (soluzio) sortu dira. Pisu horiekin *burn-in* baztertze parametro txikia erabili daiteke, berehala identitate permutazioaren egitura lortzen baita. Hala ere, *burn-in* balio txikiegia emanda ez dira identitate permutazioak lortzen, identitate permutazioa lortzeko behar baino aldaketa gutxiago egiten baitira. *burn-in* balioa 100 jarrita permutazio gehienek identitatearen egitura lortu dute, baina ez denek.

*Metropolis-Hasting* algoritmoaren zuzentasuna ziurtatu eta gero, proposatutako algoritmo heuristikoa probatu da. Kasu honetan ere bost lagin sortu dira. Metodo honek identitate permutazioen egitura aise aurkitu du, elementuen arteko lehentasunak oso erradikalak baitira. Denbora laburrean bost laginek identitate permutazioaren egitura lortu dute.

Ikasketa- eta laginketa-prozesuen funtzionamendua egokia dela egiaztatu ostean, haien efizientzia aztertzeko hirugarren proba bat egin da: prozesuaren egonkortasuna bermatzeko froga. Froga hori pisuen balioen berreskuratzean oinarritzen da, eta eredu bat ikastean, lagintzean eta berriro ikastean datza. Hasierako eta lagindu ondorengo pisuak berdinak izatea da helburua, horrela prozesuaren funtzionamendu egokia bermatzen baita. Berreskuratze-prozesuak ondo funtzionatzeko $K$ soluzio kopuru handia jartzea komeni da. Proposatutako adibidean ez dira pisuak berreskuratu soluzio gutxi erabili baitira, $K = 5$. Hala ere, hurrengo azpiataleko probetan ere berreskuratze-prozesua aztertu da.

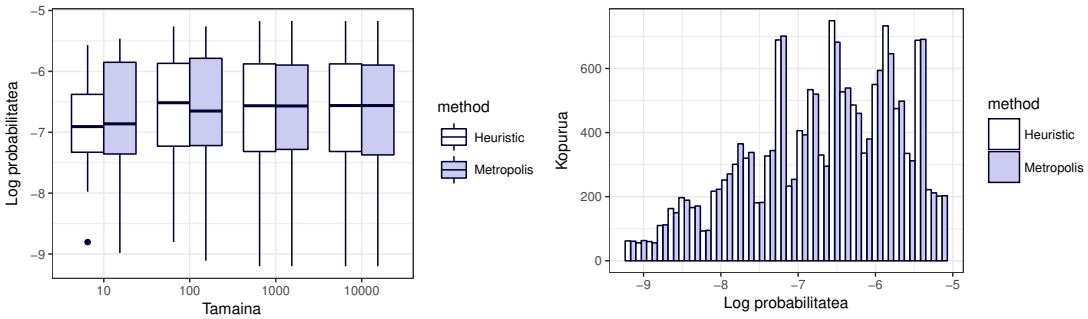
## 5.2 Laginketa-algoritmoen konparazioa

Algoritmoen zuzentasuna ziurtatu ondoren, laginketarako proposatutako bi algoritmoak alderatu dira, lortutako emaitzak eta exekuzio-denbora aztertuz. Horretarako, eredu bat ikasi eta lagindu ondoren lortutako soluzioak ikusi dira, eta, gainera, berreskuratze-prozesua kontuan izan da eredua berriz ikasiz. Eredua ikasita, laginketa-prozesuan 10.000 lagin sortu dira. *Metropolis-Hasting* algoritmoan, gainera, *burn-in* balioa 1ekoa jarri da, konplexutasuna murrizteko asmoz. Hala, alderaketa egiteko bost kasu desberdin planteatu dira, non kasu bakoitzaren ezaugarriak, helburua eta emaitzak azaldu diren.

Kasu bakoitzeko emaitzak azaltzeko bi grafiko sortu dira: kaxa-diagramak eta histogramak. Alde batetik, kaxa-diagramek sortutako lehen 10, 100, 1.000 eta 10.000 laginen probabilitateak deskribatzen dute, lagin kopurua handitzeak duen portaera azalduz. Bestetik, histogramek laginen probabilitatearen joera azaltzen dute.

Hori horrela, emaitzen oparotasuna bilatzeko tamaina desberdineko permutazioak eta populazio desberdinak erabili dira. Esan bezala, bost kasu desberdin erabili dira bi algoritmoak alderatzeko. Hona hemen bi algoritmoak alderatzeko proposatutako bost esperimentuak:

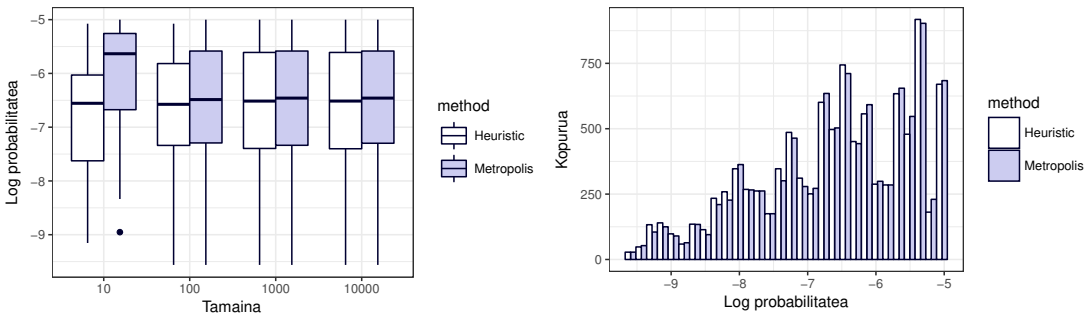
1. Lehen esperimentuan $n = 5$ tamainako 100 permutazio sortu dira populaziorako, non horietako 30 identitate permutazioa ($e$) eta 70 ausazko permutazioak diren. Ausazko permutazioek duten eragina aztertu da esperimentu honetan. 5. irudian ikus daitezke esperimentu honen emaitzak.




**5. irudia:** $n = 5$ tamainako 100 permutazioekin osatutako populazioa (30 $e$ permutazio eta 70 ausazko).

Laginen arteko probabilitatea aldakorra dela antzematen da bi grafikoetan. Kaxa-diagraman ikusi daiteke lagin kopurua handitzen den heinean probabilitatea egonkortzen dela. Limitean bi algoritmoen soluzioen probabilitatearen distribuzioa antzekoa da. Histograman probabilitate antzekoa duten permutazio-multzo asko ikus daitezke, non bi algoritmoen proportzioak antzekoak diren.

2. Kasu honetan, 30 identitate permutazio erabili beharrean, iraulia erabili da ($e^{-1}$). Hortaz, populazioa 30 $e^{-1}$ permutazioz eta 70 ausazko permutazioz osatu da. Permutazioaren egiturak garrantzia aztertu da bigarren esperimentu honetan. 6. irudian ikus daitezke lortutako emaitzen grafikoak.



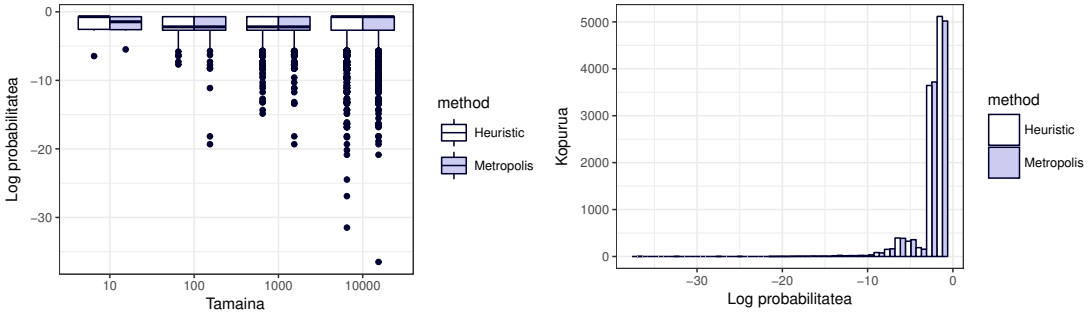

**6. irudia:** $n = 5$ tamainako 100 permutazioekin osatutako populazioa (30 $e^{-1}$ permutazio eta 70 ausazko).

Espero zen moduan, lehenengo esperimentuko grafikoekin alderatuta ez daiteke diferentzia handirik antzeman. Hortaz, pisuen berreskuratze-prozesua betetzen dela ziurtatu da. Pisuak espazio logaritmikoan adierazi dira (horren zergatia A.1.2. eranskinean irakur daiteke):

```
> abilities.true
       1        2        3        4        5
0.0000000 0.4836977 0.5935661 0.9399548 1.3744978
> abilities.metropolis
       1        2        3        4        5
0.0000000 0.4596880 0.5477201 0.9271403 1.3355586
> abilities.heuristic
       1        2        3        4        5
0.0000000 0.4915002 0.5887273 0.9453197 1.3546830
```

non `abilities.true` hasierako pisuak diren, eta `abilities.metropolis` eta `abilities.heuristic` laginketa-prozesuaren ondoren algoritmo bakoitzarekin sortutako eredua ikastean lortutako pisu berriak diren. Lortutako pisuak hasierakoen oso antzekoak direla ikusten da.

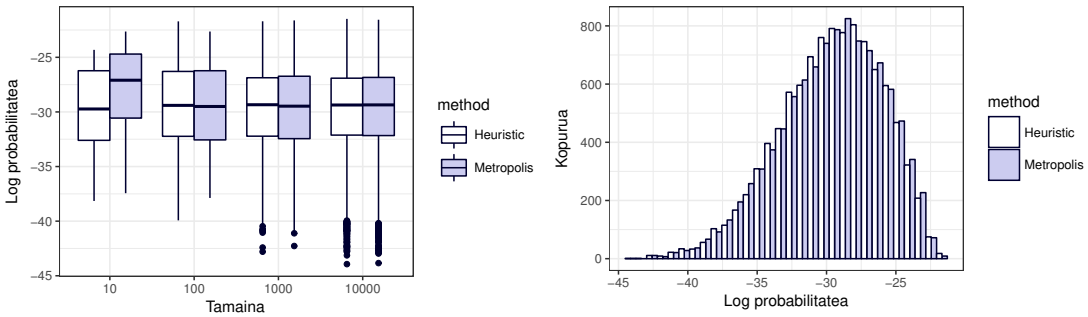
3. Kasu honetan ere lehen esperimentuaren antzeko prozesua egin da, baina, oraingoan, 30 identitate permutazio erabili beharrean 90 erabili dira. Hortaz, populazioa 90 $e$ permutazioz eta 10 ausazko permutazioz osatzen da. Ausazko permutazio kopurua gutxitzeak duen eragina ikusi nahi da. Grafikoak 7. irudian ikus daitezke.




**7. irudia:** $n = 5$ tamainako 100 permutazioekin osatutako populazioa (90 $e$ permutazio eta 10 ausazko).

Aurreko esperimentuekin alderatuta ikus daiteke lagin kopurua handitu heinean permutazioen probabilitate antzekoa lortzen dela. Kaxa-diagraman kasu berezi asko ikusi daitezke, eta histograma aztertuz ikus daiteke permutazio gehienek probabilitate berdina dutela. Identitate permutazioaren probabilitatea bilatzen dute lagin gehienek, ikasten den ereduaren aldakortasuna askoz txikiagoa baita.

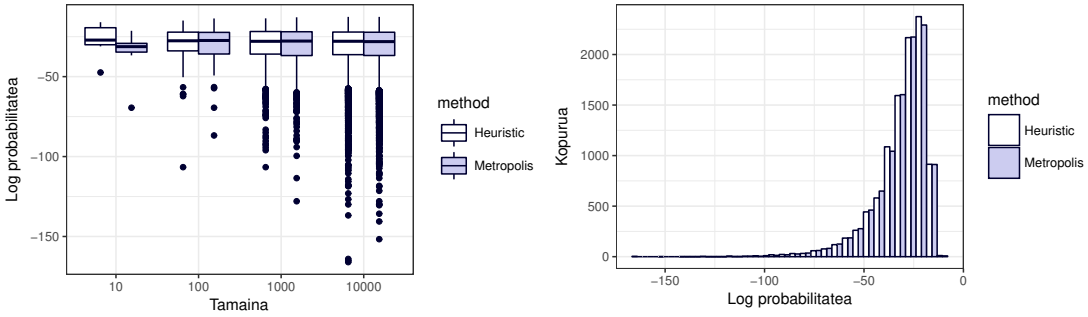
4. Bigarren esperimentuan oinarritzen da laugarren esperimentua, baina kasu honetan populazioa aldatu beharrean permutazioen tamaina aldatu da. Permutazioak $n = 5$ tamainakoak izatetik $n = 8$ izatera pasako dira, elementuen arteko probabilitatea moldatuz. Probabilitate desberdin gehiago egotea eta *Metropolis-Hasting* algoritmoaren eta algoritmo heuristikoaren arteko desberdintasuna handiagoa izatea bilatzen da. 8. irudian laugarren esperimentuko grafikoak ikus daitezke.




**8. irudia:** $n = 8$ tamainako 100 permutazioekin osatutako populazioa (30 $e^{-1}$ permutazio eta 70 ausazko).

Emaitzen heterogeneotasuna nabarmentzen da histogramaan, kanpai itxurako grafikoa lortuz. Bigarren esperimentuarekin alderatuz, kaxa-diagramaren lehen kasuan kasu bereziak daudela ikus daiteke, baina gehienek probabilitate antzekoa dutela. Kaxa-diagramako kaxen tamaina oso gutxi aldatzen da lagin kopuruaren arabera.

5. Bosgarren eta azken esperimentuan hirugarren esperimentuko datu berdinak erabili dira, baina kasu honetan ere $n = 8$ tamainako permutazioak erabili dira. Hirugarren esperimentuko histograma lortu nahi da, baina kanpai itxura gehiago nabarmenduta. 9. irudian ikus daitezke esperimentu honetan lortutako grafikoak.




**9. irudia:** $n = 8$ tamainako 100 permutazioekin osatutako populazioa (90 $e$ permutazio eta 10 ausazko).

> Elementuen pisua txikiagoarekin kanpai itxura lortu da, baina muturreko pisuak erabili direnez permutazio gehienek probabilitate antzekoa dute. Kaxa-diagrametan sakabanatzea kaxaren bi alboetan dagoela ikus daiteke, hirugarren esperimentuan ez bezala.

Bi algoritmoek grafikoetan duten portaera alderatuta ikus daiteke lortzen diren emaitzak oso parekoak direla bukaeran, baina hasieran, lehen 10 eta 100 laginekin, bi algoritmoen emaitzak desberdinak dira. Hala ere, ezin da ondorio garbi bat atera esperimentuaren arabera bata bestea baino hobea baita. Gogoratu *Metropolis-Hasting* algoritmoan *burn-in* balioa 1ekoa izan dela kasu guztietan. Horrek eragina dauka lortutako soluzioetan, baita haren konplexutasunean ere.

Halaber, exekuzio-denboran ere eragina dauka, ez baitira oso parekoak izan. Lehen hiru kasuetan, bi grafikoak lortzeko 50 segundo inguru behar dira. *Metropolis-Hasting* algoritmoak 30 segundo inguru behar ditu, eta algoritmo heuristikoak 20 segundo inguru. Beste bi esperimentuak 2 minutu inguruan egin dira. Kasu honetan bi algoritmoen arteko aldea nabaria da: *Metropolis-Hasting* algoritmoak minutua baino gehiago behar du datuak lortzeko, eta algoritmo heuristikoak, berriz, minutua baino gutxiago.

## 5.3 *Bradley-Terry* eredua EDA algoritmoetan

Aurreko azpiatalean emaitzak alderatu ondoren, algoritmo heuristikoa aproposagoa dela ondorioztatu da, eta hori izan da hain zuzen ere EDA algoritmoekin erabili dena. *Bradley-Terry* eredua erabiltzen duen EDA (BTEDA) baten bitartez LOPa ebaztea eta emaitzak *Plackett-Luce* ereduaren EDAren (PLEDA) emaitzekin alderatzea da helburua esperimentazio honetan.

Jarraian instantzien egitura eta erabilitako datuak azaldu dira, ondoren, *Bradley-Terry* eta *Plackett-Luce* ereduen emaitzak alderatzeko.

Ezer baino lehen, esperimentuaren diseinua azaldu da:

- Tamaina desberdineko sei instantzia erabili dira exekuzioa egiteko.
- Exekuzio bakoitza $1.000n$ ebaluazio ondoren geratuko da, non $n$ problemaren tamaina den (instantziaren arabera desberdina izango da).
- *Bradley-Terry* eta *Plackett-Luce* ereduek erabiltzen duten EDA algoritmoen parametroak ere definitu behar dira:
  - Populazioaren tamaina $10n$ da.
  - Onenen arteko %10 aukeratuko da laginketako iterazio bakoitzeko aukeraketan.
  - Laginketan $10n$ soluzio berri sortuko dira.
- Algoritmoak 10 aldiz exekutatu dira instantzia bakoitzeko, non hasierako populazioak ausazkoa den.

Esperimentua diseinatu eta exekutatu ondoren lortutako soluzioak gordetzeko *csv* fitxategiak erabili dira. Instantzia bakoitzaren exekuzioen datuak gorde dira, hala nola, uneko balioa, denbora, ebaluazio eta iterazio kopurua. Beste *csv* fitxategi batean instantzia guztien datuak bildu dira, 1. taularen egitura bera duena.

| Instantzia | Algoritmoa | Balioa | Denbora | Iterazioa |
|---|---|---|---|---|
| ... | Bradley-Terry | ... | ... | 1 |
| | Bradley-Terry | | | 2 |
| | ⋮ | | | ⋮ |
| ... | Plackett-Luce | ... | ... | 1 |
| | ⋮ | | | ⋮ |

**1. taula:** Emaitzen *csv* fitxategiaren egitura.

Esperimentazioaren diseinua eta emaitzen egitura ikusi ondoren, exekutatu beharreko instantziak deskribatu dira. Hauek dira erabilitako instantziak eta haien tamainak eta orain arte ezagutzen diren emaitza onenen helburu-funtzioaren balioak:

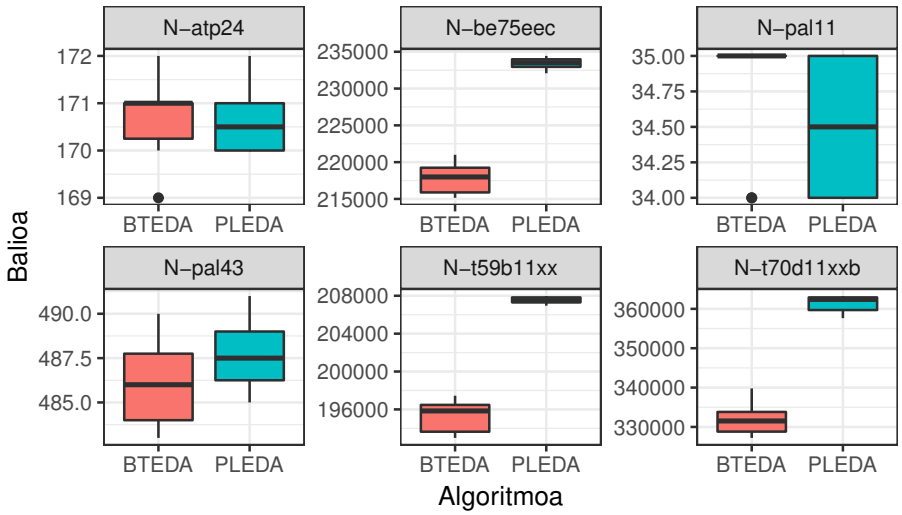
- **N-pal11**: 11 tamaina, balio onena 35.
- **N-atp24**: 24 tamaina, balio onena 172.
- **N-pal43**: 43 tamaina, balio onena 597.
- **N-t59b11xx**: 44 tamaina, balio onena 209.320.
- **N-t70d11xxb**: 44 tamaina, balio onena 366.469.
- **N-be75eec**: 50 tamaina, balio onena 236.464.

Jarraian exekuzioa aurrera eramateko kodearen adibide bat aurkezten da, *Bradley-Terry* ereduaren EDA algoritmoarena (BTEDA), alegia. *Plackett-Luce* ereduarena (PLEDA) antzekoa izango litzateke. Hori horrela, hasierako populazioa definitu eta instantzia bakoitzaren LOPa sortu ondoren (A. eranskinean azalduta dagoen moduan), exekutatu beharreko R kodea hurrengoa da:

```
args<- list()
args$evaluate <- problem$evaluate
args$initial.population <- initial.population
args$selectSubpopulation <- elitistSelection
args$selection.ratio <- 0.1
args$learn <- bradleyTerry
args$non.valid <- "discard"
args$resources <- cResource(evaluations = 1000 * problem$size)
object <- do.call(basicEda, args)
```

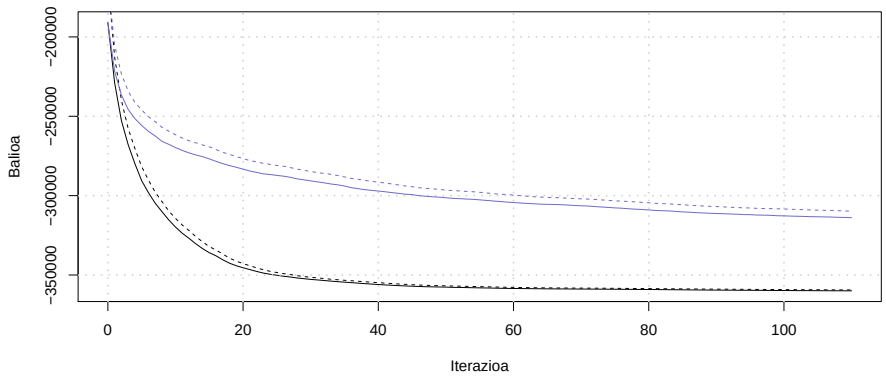
non `initial.population` $10n$ tamainako ausazko permutazio-multzoa den. Esan den moduan, `learn` parametroan `plackettLuce` aldagaia jarriz PLEDA exekutatzen da.

10. irudian instantzia guztien exekuzio-emaitzak kaxa-diagrametan irudikatu dira bi algoritmoen portaerak alderatuz. Ikusi daitekeen moduan instantzia txikiekin lan egitean lortzen diren emaitzak nahiko parekoak dira (**N-pal11**, **N-pal43** eta **N-atp24** instantzietan, alegia). Instantzia handietan, ordea, *Plackett-Luce* ereduarekin alderatuz emaitza txarragoak lortzen dira.




**10. irudia:** Bi ereduen portaera proposatutako instantziekin. 19 ordu behar izan dira exekuzioa egiteko.

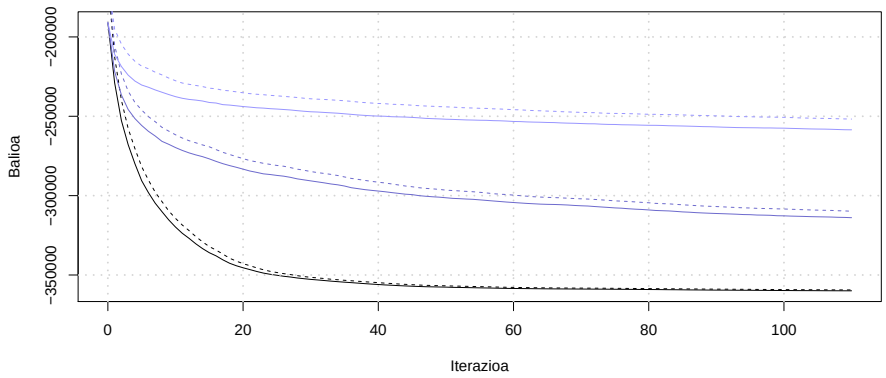
Emaitza horien zergatian sakontzeko, 11. irudian, *Bradley-Terry* eta *Plackett-Luce* ereduen EDA algoritmoek 100 iterazioetan lortutako emaitza onena eta populazioaren batezbestekoa irudikatu da, zehazki **N-be75eec** instantziarena. Argi ikus daiteke *Bradley-Terry* ereduren EDA algoritmoaren konbergitzea motelagoa dela *Plackett-Luce*arena baino (funtzioaren malda mantentzen da BTEDAn, eta, hortaz, algoritmoak soluzioak hobetzen jarraitzen du). Bestalde, media urrutiago dago balio onenetik alderantziz baino.




**11. irudia:** *Plackett-Luce* eta *Bradley-Terry* ereduek **N-be75eec** instantzian zehar izandako portaera. Urdina *Bradley-Terry* eredua da, eta beltza *Plackett-Luce* eredua.

EDA algoritmoetan ereduaren laginketa-prozesuaren garrantzia ikusteko asmoz, *Bradley-Terry* ereduaren laginketa aldatu eta alderatu da. Laginketa-prozesu berritzat *Metropolis-Hasting* algoritmoa proposatu da. Hala ere, bi algoritmoak alderatu direnean, *Metropolis-Hasting* algoritmoaren konplexutasuna oso handia dela aipatu da. Hori horrela, problema exekutatzen jarri orduko emaitzak lortzea oso garestia dela ondorioztatu da ($n = 44$ tamainako instantziako balio bakoitza kalkulatzeko 5 minutu baino gehiago behar ditu). Hortaz, ondoriozta daiteke *Metropolis-Hasting* metodoa EDA algoritmoekin lan egiteko ez dela bideragarria.

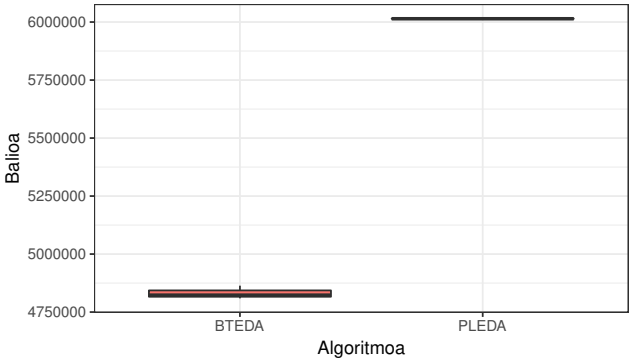
Ikusita *Metropolis-Hasting* metodoarekin lan egitea ezinezkoa dela, eredua ausazko aldatzea proposatu da. $n = 44$ tamainako instantziarekin lortutako emaitzak 12. irudian grafikoki ikus daitezke.




**12. irudia:** **N-t70d11xxb** instantzian zehar ereduek izandako portaera. Onena, beltza, PLEDA da, erdikoa BTEDA eta kolore argienekoa ausazko EDA. 2 ordu behar izan dira exekuzioa egiteko.

Ikusi daitekeen moduan, proposatutako ereduak baino emaitza txarragoak lortzen dira ausazko EDArekin, espero zen moduan. Hortaz, *Bradley-Terry* ereduak ausazko EDAk baino emaitza hobeak lortzen dituela jakitea ondorio positiboa da; alegia, ereduak eragin positiboa du bilaketan.

Ausazko EDAren eta BTEDAren arteko diferentzia ikusi ondoren, instantzia handiagoekin proba bat egin da emaitzak errepikatzen diren ikusteko asmoz. $n = 150$ tamainako LOP instantzia batekin [5] lortutako emaitzak azaldu dira 10. irudian. Kasu horretan, prozesua ez luzatzeko, BTEDA eta PLEDA algoritmoak 4 aldiz exekutatu dira instantzia horretan. Aurreko kasuetan bezala, *Bradley-Terry* ereduarekin lortutako emaitzak txarragoak izaten jarraitzen dute 13. irudian ikus daitekeen moduan.




**13. irudia:** *Plackett-Luce* eta *Bradley-Terry* ereduen EDAek **diff150_0** instantzian izandako portaera. 19 ordu behar izan dira instantzia hau exekutatzeko.

# 6 Ondorioak eta etorkizuneko lanak

Txostenaren hasieran proiektuaren egituraren oinarri teorikoak azaldu dira, EDA algoritmoaren eta *Bradley-Terry* ereduaren zehaztasunak aipatuz, azken finean, LOPa ebazteko erabili baitira. Horrez gain, ereduan garatutako bi laginketa-algoritmoak azaldu eta alderatu dira, bakoitzaren funtzionamendua eta emaitzak aztertuz. Aurreko atalean ereduaren inguruko esperimentazioak egin dira, bai laginketaren portaera ikusteko baita LOPa ebazteko ereduaren EDAen aztertzeko ere. Atal honetan aurreko esperimentazioen ondorioak azaldu dira eta, horrez gain, etorkizuneko lan interesgarriak proposatu dira, proiektua sakontzeko nahiarekin.

Lehenik eta behin, kodetutako ereduaren ikasketa- eta laginketa-prozesuen zuzentasuna probatzean ondorio bat atera da. Ikasketa-prozesuan identitate permutazioak bakarrik erabili dira, baina populazioaren tamainak eragin handia dauka. Proposatutako esperimentazioan $n = 5$ tamainako $e$ permutazioz osatutako populazio batetik eredua ikasi eta 5 lagin lortu dira. Populazioa identitate permutazioarekin osatuta egon arren, populazioaren tamainak garrantzia du ereduaren pisuetan. *Metropolis-Hasting* algoritmoan, adibidez, emaitza berdinak lortu dira *burn-in* balioa gutxituta eta populazioa handituta. Kasu horretan 1.000 tamainako populazioa finkatu da eta *burn-in*= 100. Beraz, populazioa handituz elementuen arteko probabilitateak handitzea lortzen da, soluzioen egonkortasun egokia bilatuz.

Ondorio horren ondoren, laginketarako proposatutako bi algoritmoak alderatzean lortutako emaitzak aztertu dira. 5.2. ataleko grafikoak ikusita hainbat puntu nabarmen daitezke. Oro har, permutazioen tamaina handitzean heterogeneotasuna handitzen dela antzeman daiteke, kanpai itxura zehatzagoa lortuz; garbi ikus daiteke bigarren eta laugarren esperimentazioetan lortutako histogramak alderatzean. Horren ildotik, ereduaren pisuak muturrekoak direnean probabilitate berdina duten permutazio kopurua handitzen dela ikusten da, hirugarren eta bosgarren esperimentazioetan ikusi daitekeen moduan. Bi kasuetan azken kaxa-diagramak txikiak direla nabarmen daiteke, eta kasu berezi batzuk bereizten dira. Gainera, permutazioaren tamaina handitzean prozesuak eredua ikasteko eta lagintzeko denbora gehiago hartzen duela ondoriozta daiteke.

Horrez gain, bigarren esperimentazioan egin den berreskuratze-prozesutik ere ondorioak atera daitezke. Ikus daiteke lortutako emaitzak nahiko parekoak direla, baina antzekotasuna gehiago manten daiteke. Eredua ikasi eta lagindu ondoren informazioa galtzeko joera dago, eta hori ekiditeko lagin kopuru infinitu batekin lan egin beharko litzateke. Hortaz, pisuen antzekotasuna gehiago mantentzeko laginketan sortu beharreko lagin kopurua handitu beharko litzateke.

Aurreko guztia ondorioztatu ondoren, *Bradley-Terry* ereduaren porrotaren zergatia aurkitu behar da, eta horretarako inplementatutako *Bradley-terry* eredua aztertu beharra dago. Esan behar da LOPak duen kokapen denotatiboa eta *Bradley-Terry* ereduaren parekatze metodoa konbinatzeak zentzua duela. Hala, eredua aztertzeko honako etorkizuneko lanak proposatu dira:

- Laginketan sortutako soluzio kopurua handituz esperimentazioa errepikatzea.
- *Metropolis-Hasting* algoritmoa ausazko permutazioarekin hasi beharrean momentuko onenarekin hastea. Era berean, 4. atalean proposatu den

baztertze-prozesuaren funtzionamendua aztertu.

- *Bradley-Terry* ereduaren ikasketa-prozesuaren funtzionamendua zehazki aztertzea, hau da, probabilitateen eboluzioan sakondu.

Horrez gain, proposatutako eredua alderatzeko beste hainbat prozesu egin daitezke:

- Eredu gehiagorekin alderatu, hala nola, *Mallows* eredua *Hamming* distantzia erabiliz.
- *Bradley-Terry* eredua beste problema batzuetan erabili, $c - GAP$ problema, adibidez.

Proiektuan 300 ordu aurreikusi direnez ez da aurreko zerrendako lanik egin, baina proposatutako lanak egitekotan *Bradley-Terry* ereduaren inguruan informazio baliagarria lortuko litzateke.

# 7 Jarraipen eta Kontrola

Proiektuan zehar egindako lana azaldu ondoren, atal honetan, aurreikusitako planifikazioarekiko izandako desberdintasunak eta ikasitako lezioak azaldu dira. Gai bakoitzari azpiatal bat eskaini zaio.

## 7.1 Jarraitutako planifikazioa

Proiektuaren aspektu guztiak diseinatu eta azaldu ondoren, 2. taulan errealitatean emandako denbora eta aurreikusitako alderatu dira.

| **Ataza/zeregina** | **Aurreikusitako denbora** | **Denbora erreala** |
|---|---|---|
| Kudeaketa | 25 | 27.5 |
| Planifikazioa | 4 | 4 |
| Jarraipen eta kontrola | 11 | 11 |
| Bilerak | 10 | 12.5 |
| Aurrekariak | 30 | 24 |
| Irakurketa | 20 | 20 |
| metaheuR | 10 | 4 |
| Diseinua eta inplementazioa | 50 | 40 |
| Ikasketa | 25 | 12.5 |
| Laginketa | 25 | 27.5 |
| Zuzentasuna | 10 | 12.5 |
| Esperimentazioa | 110 | 129 |
| Prozeduraren definizioa | 10 | 15.5 |
| Exekuzioa | 50 | 63.5 |
| Emaitzak eta ondorioak | 50 | 50 |
| Itxiera | 75 | 85 |
| Memoria | 60 | 70 |
| Defentsa | 15 | 15 |
| **GAP** | **300** | **318** |

**2. taula:** Jarraipenaren taula. Plangintzan azaldutako eta errealitatean emandako ordu kopuruak azaltzen dira bertan.

Orokorrean, aurreikusitako denborarekiko izandako desberdintasunak ondo banatu dira, atal batzuetan denbora gehiago eskaini baita, eta beste batzuetan gutxiago; baina proiektuaren ordu kopurua errespetatu da. 300 orduko lan batean 20 orduko diferentzia egotea txikikeria da. Atal bakoitzaren diferentzia aztertu da jarraian, bakoitzaren zergatiak azalduz.

**Kudeaketan**, **aurrekarietan** eta **zuzentasunean** eskainitako denborak nahiko parekoak dira. Lehenengo atala bilerengatik luzatu da, aurreikusitakoak baino gehiago egin baitira. Bigarrenean behar baino denbora gutxiago eskaini da **metaheuR** paketea *Bilaketa Heuristikoak* ikasgaian landu baitzen. Hirugarren atala gutxi luzatu da, baina ez zaio garrantzi handirik eman.

Beste hiru ataletan denbora diferentziak handiagoak izan dira. **Inplementazioan** eskainitako denbora murriztu da eredua ikasteko **BradleyTerry2**

paketeko funtzioak erabili baitira. Laginketa-prozesuari aurreikusitakoaren oso antzekoa eskaini zaio.

Hala ere, **esperimentazio** atala izan da desberdintasun handiena izan duena. Alde batetik, esperimentazioa egiteko kodeketan behar baino denbora gehiago eskaini da. Bestetik, behin baino gehiagotan egin behar izan da exekuzioa emaitzak esperotakoak ez zirelako (exekuzio bakoitzak ia 20 ordu hartu ditu).

Proiektuaren **itxieran** ere denbora aurreikusitakoa baino denbora gehiago eskaini zaio, egindako guztia txosten batean sartzeak duen zailtasuna dela eta.

Bukatzeko, aldaketa horiek kontuan hartuta, proiektuaren kalitatea azaldu da. Proiektu zabal bat egin da, eredu bat proposatu, kodetu eta probatu baita. Egia da esperimentazioko emaitzak ez direla esperotakoak izan, baina horregatik proposatu dira etorkizuneko lanak.

## 7.2 Ikasitako lezioak

Proiektua egin heinean ateratako ondorioak zerrenda batean azaldu dira atal honetan. Hauek dira proiektuan zehar ikasitako lezioak:

- Denbora kudeaketa ondo maneiatzea. Atal batean aurreikusitakoa baino denbora gutxiago ematekotan, beste atal batean gehiago sakondu (eta berdina alderantziz).
- Materiala ikasteko denbora eskaini. Proiektu honetan paketeak erabiltzen jakiteko hainbat artikulu irakurri dira, eta, gainera, interneteko estekak erabili dira arazoei aurre egiteko.
- Kodearen konplexutasunak garrantzi handia du exekuzioan. Horretarako, R lengoaiak eskaintzen dituen funtzioak erabili begizten eta baldintzen ordez.
- Memoria idaztea lan neketsua da. Horregatik, proiektuarekin aurrera egin ahala prozesua idaztea komeni da. Exekuzioaren denbora ere probetxuzkoa izan daiteke beste ataletan aurreratzeko.
- Ezustekorik ez gertatzeko hautabideak izan. Momenturen batean ordenagailu pertsonalean lan ez egitekotan, kodea eta memoria eskuragarri egon dira interneten.
- Eredu probabilistiko bat diseinatzea eta inplementatzea. Proiektuaren helburuetan azaldu den moduan, ikerlan bat zer den ikasi da, probabilitate kalkulu eta inferentziako kontzeptu aurreratuak landu baitira.

# Erreferentziak


[1] Borja Calvo, Usue Mori, and Josu Ceberio. *Bilaketa heuristikoak. Teoria eta Adibideak R Lengoaian*. Euskal Herriko Unibertsitatea, 2017.

[2] Borja Calvo, Usue Mori, and Josu Ceberio. *metaheuR: Package including classical metaheuristics and combinatorial optimization problems*, 2017. R package version 0.35.

[3] Francois Caron and Arnaud Doucet. Efficient bayesian inference for generalized bradley–terry models. *Journal of Computational and Graphical Statistics*, 21(1):174–196, 2012.

[4] J Ceberio. *Solving permutation problems with estimation of distribution algorithms and extensions thereof*. PhD thesis, PhD thesis, Faculty of Computer Science, University of the Basque Country, 2014.

[5] Josu Ceberio, Alexander Mendiburu, and Jose A Lozano. The plackett-luce ranking model on permutation-based optimization problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 494–501. IEEE, 2013.

[6] Douglas E Critchlow, Michael A Fligner, and Joseph S Verducci. Probability models on rankings. *Journal of mathematical psychology*, 35(3):294–318, 1991.

[7] Persi Diaconis. Group representations in probability and statistics. *Lecture Notes-Monograph Series*, 11:i–192, 1988.

[8] David Firth et al. Bradley-terry models in r. *Journal of Statistical software*, 12(1):1–12, 2005.

[9] David Firth and Heather L Turner. Bradley-terry models in r: the bradleyterry2 package. *Journal of Statistical Software*, 48(9), 2012.

[10] Michael A Fligner and Joseph S Verducci. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 359–369, 1986.

[11] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[12] Jatinder ND Gupta and Edward F Stafford. Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711, 2006.

[13] David R Hunter. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406, 2004.

[14] Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

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

[16] Jose A Lozano. *Towards a new evolutionary computation: advances on estimation of distribution algorithms*, volume 192. Springer Science & Business Media, 2006.

[17] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

[18] Colin L Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[19] Rafael Martí, Gerhard Reinelt, and Abraham Duarte. A benchmark library and a comparison of heuristic methods for the linear ordering problem. *Computational optimization and applications*, 51(3):1297–1317, 2012.

[20] Martin Pelikan and David E Goldberg. Hierarchical problem solving and the bayesian optimization algorithm. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pages 267–274. Morgan Kaufmann Publishers Inc., 2000.

[21] Martin Pelikan, Kumara Sastry, and Erick Cantú-Paz. *Scalable optimization via probabilistic modeling: From algorithms to applications*, volume 33. Springer, 2007.

[22] Robin L Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[23] Filip Radlinski and Thorsten Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 570–579. ACM, 2007.

[24] Colin R Reeves. Improving the efficiency of tabu search for machine sequencing problems. *Journal of the Operational Research Society*, pages 375–382, 1993.

[25] Tommaso Schiavinotto and Thomas Stützle. The linear ordering problem: Instances, search space analysis and algorithms. *Journal of Mathematical Modelling and Algorithms*, 3(4):367–402, 2004.

[26] Richard P Stanley. Enumerative combinatorics, wadsworth publ. *Co., Belmont, CA*, 1986.

[27] P Toth and D Vigo. The vehicle routing problem, ser. *Monographs on Discrete Mathematics and Applications. SIAM*, 2001.

[28] Heather Turner and David Firth. Bradley-terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, 48(9):1–21, 2012.

[29] Ilker Yildirim. Bayesian inference: Metropolis-hastings sampling. *Dept. of Brain and Cognitive Sciences, Univ. of Rochester, Rochester, NY*, 2012.

# A Inplementazioa

Proiektu honetan kodetutako xehetasunak aztertu dira eranskin honetan. Bi azpiataletan banatu da eranskina: alde batetik, erabilitako pakete nagusien deskribapen labur bat egin da; bestetik, sortutako *Bradley-Terry* klasearen egitura aipatu da.

## A.1 Paketeen deskribapena

Proiektuan, Rko bi pakete erabili dira nagusiki: **metaheuR** [2] eta **BradleyTerry2** [28]. Lehena irakaskuntzarako paketea da, EHU unibertsitateko kideek garatutakoa. Pakete hori izan da, hain zuzen, *Bradley-Terry* ereduarekin luzatu dena. Beste paketea, *Bradley-Terry* eredua konputatzeko paketea da. **BradleyTerry** paketearen [8] luzapena da, interfaze hobea eta ereduen sorta zabalagoa eskaintzen duena. Ondoren, aipatutako bi paketeak laburki deskribatu dira.

### A.1.1 metaheuR paketea

**metaheuR** [2] algoritmo metaheuristiko eta problema ugari inplementatuta dituen Rko paketea da. *Github* plataforman dago eskuragarri. Hortaz, lan egiteko biltegia adarkatu [1] eta Rko lan espaziora gehitu behar da. Ohiko paketea hedatu da esteka horretan, eta, bertan, proiektuan zehar egindako aldaketa guztiak daude.

Proiektua egiteko paketeko hainbat funtzio erabili dira. LOP problema paketeko `lopProblem` funtzioarekin sortzen da. Proiektuan instantziekin lan egin denez, R espazioan instantziak kargatu eta horiekin problema sortu da.

Horrez gain, paketeak eskaintzen duen `basicEDA` algoritmo metaheuristikoa erabili da EDA algoritmoarekin lan egiteko. EDA algoritmoetan *Bradley-Terry* eredua erabiltzeko banaketa probabilistikoen klase berri bat sortu behar da (`BradleyTerry` klasea). EDA algoritmoen barruan distribuzioak erabiltzeko klaseak dira, non ikasketa eta laginketa-prozesuak inplementatzen diren (A.2. azpiatalean sakondu da horren inguruan).

### A.1.2 BradleyTerry2 paketea

Elementuak binaka alderatzeko eredua sortzen duen Rko paketea da **BradleyTerry2** [28]. Eredua bi jokalariren arteko irabazteko probabilitatean oinarritzen da, baina, permutazioekin lan egin denez, elementuen arteko konparaketa gisa interpretatu behar dira; hau da, $\sigma(i)$ elementua $\sigma(j)$ elementuaren aurretik egoteko probabilitatea $i$ elementuak $j$ elementuari irabazteko probabilitatea da.

**BradleyTerry2** paketeak [28] permutazio-populazio batetik 3. taula moduko irabazte-taula bat erabiltzen du eredua sortzeko. Sortutako irabazien taulan populazioko elementuen arteko irabazien eta galeren datuak azaltzen dira. Hori horrela, eredua sortzeko paketeak duen `BTm` funtzioa erabiltzen da, eta funtzioak itxura hau dauka Rn:

[1] `https://github.com/joanalza/BradleyTerry`

| Elementua 1 | Elementua 2 | Irabaziak | Galtzeak |
|---|---|---|---|
| 1 | 2 | 0 | 4 |
| 1 | 3 | 1 | 3 |
| 2 | 3 | 3 | 1 |

**3. taula:** Irabazien taula.

```
BTeredua <- BTm (cbind(win1,win2), player1 = Elementua 1, player2 =
    Elementua 2, formula = ~ Elementua, id = "Elementua", data =
    Irabazte taula)
```

Eredua sortzean pisuak lortzen dira *coefficients* atalean. Elementu eta populazio kopuru txikiekin lan egitean erabilitako prozedura egokia izan daiteke. Baina, zer gertatzen da kopuruak handitzean? Permutazioaren tamaina handitzean probabilitateak gero eta txikiagoak dira; horrela, arazo numerikoak izan daitezke . Hori ekiditeko espazio logaritmikoan lan egitea proposatzen da literaturan, eta **BradleyTerry2** paketeak [28] ere espazio horretan lan egitea eskaintzen du. $w$ pisuak $a = log(w)$ *ability*ak izatera pasatzen dira. *Ability*ak paketeko `BTabilities` funtzioarekin lortzen dira, funtzioari sortutako eredua parametro bezala emanda.

## A.2 *BradleyTerry* klasea

Proiektuaren mamia **metaheuR** paketean [2] sortutako klasean dago. EDA algoritmoen barruko distribuzioa erabiltzeko sortutako klasea da `BradleyTerry`. Hauek dira klasearen 4 parametroak:

- *abilities*: ikasketa-prozesuan lortutako *ability*en *slot*ak (Rko atributuak).
- *sampling*: laginak lortzeko erabiliko den metodoa. *Heuristic* eta *Metropolis* dira balio posibleak.
- *indices*: elementuen arteko konparaketarako erabilitako bikote guztiak.
- *burn-in*: *Metropolis-Hasting* algoritmoaren *burn-in* parametroa.

Azken bi parametroak *Metropolis-Hasting* algoritmoan erabiltzen dira; *indices* permutazio-probabilitateen kalkulua azkartzeko eta *burn-in* parametroa hasieran sortutako permutazioak baztertzeko.

Parametroak azaldu ondoren klasearen egitura azaldu da. Klasean bi funtzio nagusi bereizi dira: funtzio eraikitzailea eta *simulate* funtzioa. Lehena, ikasketa-prozesurako erabili da, eta, bigarrena, laginketa-prozesurako.

Funtzio eraikitzailean **BradleyTerry2** paketeak eskaintzen duen eredu sortzailea erabili da eredua ikasteko. Irabazien taula sortu behar da gero eredua sortu ahal izateko. Adibidez, $(213; 231; 321; 231)$ permutazioz osatutako populazioaren taula 3. taulan ikus daiteke. Taulak elementu guztien arteko irabazte- eta galtze-kopuruak gordetzen ditu. Zehaztasun bat nabarmen daiteke: errenkada bakoitzeko irabazte- eta galtze-kopuruen batura permutazio kopuruaren berdina da. Azaldutako adibidean populazioa 4 permutazioz osatuta dagoenez batura guztiak 4 dira.

*Simulate* funtzioan, aldiz, proposatutako bi laginketa metodoak inplementatu dira: *Metropolis-Hasting* eta algoritmo heuristikoa. Erabili beharreko algoritmo aukeratzeko *sampling* parametroa erabiltzen du. Bi algoritmoen funtzionamendua 4. atalean azaldu da.