eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Optimizazio heuristikorako R pakete baten hedapena

Egilea

Onintza Aracama Eceiza

Zuzendaria

Borja Calvo Molinos

informatika fakultatea
facultad de informática

2017ko otsailaren 2a

Laburpena

Memoria-txosten hau, Onintza Aracama Eceizak, 2016-2017 ikasturtean, UPV/EHU unibertsitateko Donostiako Informatika Fakultatean Informatika Ingeniaritzako Graduari amaiera emateko egindako Gradu Amaierako Proiektua dokumentatzeko egin dut. Proiektuaren zuzendaria Borja Calvo da, Donostiako Informatika Fakultateko irakaslea da.

Proiektua Bilaketa Heuristikoak irakasgaiari zuzendurik egin dugu. Bilaketa Heuristikoak Konputazio Zientzia eta Adimen Artifiziala Saileko laugarren mailako hautazko irakasgaia da. Irakasgai honen aztergaia optimizazio-problemen ebazpena da. Horretarako, problema horietara egoki daitezken algoritmo orokorrak aztertzen dira. Problemek, soluzio ugari eduki ditzakete, baina beraien kalitatea neurtzeko erarik izanez gero, soluziorik onena bilatuko genuke, eta horixe izango da optimimizazioaren helburua. Hau da, aukera guztien artean soluziorik onena topatzea. Bilaketa Heuristikoak irakasgaian metaheuR izeneko R pakete bat garatzen da optimizazio-problemekin probak egiteko, pakete horretan zenbait algoritmo eta problema daude inplementaturik.

Gradu amaierako proiektu honetan bi gauza lortu ditugu. Alde batetik, aurretik aipatutako R pakete horren hedapena egin dugu, hau da, problema eta algoritmo gehiago txertatu ditugu. Beste aldetik berriz, R pakete horren erabilera hobeto uler dadin, web-aplikazio bat diseinatu eta eraiki da, esperimentazio sinpleak egiteko laguntza bat oso erabilgarria baita. Horrela, pakete horretako algoritmo eta problemen emaitzak eta progresioak, modu bisualago batean ikusi ahal izango ditu Bilaketa Heuristikoa irakasgaia ikasten ari den ikasleak.

Eskerrak

Lehenik, eskerrak eman naiz dizkiot nire proiektuaren zuzendari izan den Borja Calvori, proiektu osoan zehar emandako animo, aholku eta orientazioagatik.

Bestetik, azken urte hauetan nire bidelagun izan diren klasekide eta gainerako lagunei, pasa ditugun eta pasako ditugun momentu guztiengatik.

Azkenik, familiari eta Iñakiri, egun ilunak argitzen laguntzeagatik.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Proiektua

1.1 Sarrera

Buruhaustez beteriko mundu honetan, egunerokotasunean, arazoei aurre egin nahi izaten diegu, ahalik eta soluzio onena emanez geure problemei. Adibide bat jartzearren, produktu bat erosterakoan, zeri begiratzen diogu? Produktuaren kalitateari? Produktuaren prezioari? Ahal izanez gero, behintzat, ahalik eta kalitate oneneko produktua nahiko genuke, ahalik eta merkeen. Baina hori posible ote da? Horrexegatik, geu konturatu gabe, geure iritziez baldintzatzen gara erosterako orduan eta azkenik baldintza horietara egokitzen den produktua erosiko dugu. Antzeko problemak planteatzen dira informatika eta beste hainbat zientzietan.

Problemek soluzio bat baino gehiago izaten dituzten gehienetan. Baina zein izango da soluzio horien artean hoberena? Emaitzarik onena emango diguna? Soluziorik onena topatzea helburu duten problema hauei, optimizazio-problema deitzen zaie.

Problemen soluziorik onena aurkitzen ari bagara, problema horiei optimizazio-problema deituko diogu. Problema hauen kalitatea helburu- funtzio bidez neurtzen da; eta horrela, funtzio hori minimizatuz, edo maximizatuz, soluziorik onenarekin topatuko gara. Soluzio maximo edo minimo honi beraz, soluzio optimoa deitzen zaio.

Definizioa 1.1. ***Minimizatze-problema*** *Izan bedi S soluzio bideragarrien multzoa eta $f : S \to \mathbb{R}$ helburu-funtzioa. Minimizazio-problema optimo globala $s^* \in S$ topatzean datza non $\forall s \in S, f(s^*) \leq f(s)$*

Gure helburu-funtzioa maximizatu nahi izanez gero, berriz, funtzio berri bat deinituko dugu: $g = -f$.

Bilaketa Heuristikoak [Calvo et al., 2015] irakasgaian, problema hauek aztertzen dira eta irakasgaiaren helburua, optimizazio-problemen ebazpena ikastea da. Horretarako, ikasturtean zehar, problema mota ezberdinak dira hizpide eta problema horien ebazpenerako aplikatu beharreko algoritmo heuristikoak azaltzen dira.

Ikasitako problema eta algoritmoek osatutako, metaheuR [Calvo et al., 2014] izeneko R pakete bat erabiltzen dute ikasleek, barneratutakoa praktikan jartzeko probak eginez. Hala ere, jakina da problema mota asko daudela, eta hori horrela, pakete horretan ez daude problema guztiak inplementaturik. Horrexegatik, proiektu honetan eginbeharrekoetako bat horixe izan da: bi problema gehiago inplementatzea eta problema horiekin esperimentazioa gauzatzea. Hau honela izanik, metaheuR paketean edozein momentutan gehitu ahal izango dira problema berriak eta algoritmoak.

Beste aldetik eta bukatzeko, emaitzak bisualagoak izan daitezen, eta ikasleek hobeto uler dezaten, klasean ikasi eta ikusitakoa, web-aplikazio sinple bat egitea izan da proiektuaren bigarren eginbeharrekoa, inplementatu berri diren problema berriekin. Aurrez aipatu bezala, hemen ere, web-aplikazioa prest egongo da, edozein unetan kode berriak gehitu ahal izateko.

1.1.1 Motibazioa

Proiektu hau, esan bezala, Bilaketa Heuristikoak irakasgaiari loturikoa da. Interes nagusiak bi izan dira proiektu hau garatzeko: Bata irakasgaiarena eta bestea egilearena.

Irakasgaian, problema eta algoritmoak biltzen dituen R pakete bat erabiltzen da eta jakina da, hau nahi hainbeste luza daitekeela. Horregatik, bertan erabiltzen den R paketea garatzea helburuetako bat duenez, garapen hori ondo datorkio irakasgaiari. Ildo beretik, egingo den web-aplikazioa ere erabilgarri izango dute irakasgai hori ikasten dutenek, horrexegatik, interesatuak irakasgaiari lotutako denak direla esan daiteke.

Bestalde, proiektu hau Gradu Amaierako Proiektu bat denez, bigarren interesatu nagusia egilea da, hau da, ni neroni. GAPa egiteaz gain, Bilaketa Heuristikoak irakasgaiari buruz ikasi dut, irakasgaia ez bait dut eman graduan zehar, eta orain artean ikasi dudana praktikan jartzeko aukera ezin hobea izan da.

1.1.2 Memoriaren egitura

1. **Proiektua**: Lehenengo kapituluan, proiektuaren nondik norakoak azaltzen ditugu. Baita proiektu hau aukeratzera bultzatu gaituzten arrazoiak ere.

2. **Proiektuaren Kudeaketa Plana**: Proiektuaren kudeaketaren nondik norakoak azaltzen dira: proiektua nola egituratzen den eta nola inplementatzen den. Helburuak, irismena eta emangarriak deskribatzen dira, interesatuen paperekin eta betebeharrekin batera.

3. **metaheuR paketea eta haren hedapena**: Kapitulu honetan paketearen deskribapen zehatza egiten da eta ondoren inplementatu diren problemak azaltzen ditugu.

4. **Esperimentazioa eta erabilera kasua**: Gure proiektuko atal praktikoa agertzen da atal honetan. Hasieran, erabili ditugun instantziak azaltzen dira eta ondoren, algoritmo bakoitza, bere parametroak, lortutako emaitzak eta ateratako ondorioak azaltzen ditugu.

5. **Ondorioak eta jarraipena**: Atal honetan, proiektuan zehar egindako lana berrikusten da, hasierako plangintza eta helburuak, lortutako emaitzarekin alderatuz eta amaitzeko proiektuaren osotasunetik ateratako ondorioak azaltzen dira.

2. KAPITULUA

Proiektuaren Kudeaketa Plana

Bigarren kapitulu honetan, Gradu Amaierako Proiektu hau burutu dadin egindako kudeaketa plana azalduko da. Proiektuaren helburuak lortu ahal izateko egindako plangintzan, proiektuaren atazak hartu dira kontutan eta ataza horiek burutzeko egutegi bat zehaztu da. Gainerako planak ere egin dira: Kalitate-plana, komunikazio-plana eta arriskuen kudeaketa-plana, hain zuzen ere.

2.1 Helburua

Proiektuaren helburuak bi multzotan banatzen dira: alde batetik, proiektuaren egilearen helburuak eta, bestetik, Bilaketa Heuristikoak irakasgaiaren helburuak.

2.1.1 Ikaslearen helburuak

Proiektuak hainbat helburu ditu: helburu nagusiak, hau da, produktuarekin lotutakoak eta helburu osagarriak, helburu nagusiak konplimentatzeko bete beharreko helburuak.

- Helburu nagusiak:
  - Orokorrak:
    * Graduari lotutako konpetentziak aplikatzea. Azken urteetan ikasitakoa barneratzea eta horiekin loturiko galderei erantzun ahal izatea.

        * Hautatutako lerroan, hau da, Konputazio espezialitatean ikasitakoa praktikan jartzen ikastea.
    - Espezifikoak:
        * Bilaketa Heuristikoak irakasgaian ikasten eta osatzen den metaheuR izeneko paketea luzatu ahal izateko irakasgaian ikasiten diren kontzeptuak ulertu eta ikastea.
        * Problema mota desberdinak eta beraiek ebazteko algoritmo metaheuristikoak bereiztea, ulertzea eta ikastea.
        * Web-aplikazio bat eraikitzen ikastea, orain arte erabili gabeko teknologia batekin: Shiny.
        * Sortutako web-aplikazioaz baliatzea, aurretik paketean zeuden nahiz gehitutako problemen ebazpenerako. Horrela experimentazio bat egin eta emaitza erabakigarriak lortzea.

- Helburu osagarriak:
    - Dokumentazio aberats bat idaztea egindako lan guztia azaltzeko. Gainera horretarako erabilitako teknologiaren erabilera barrentzea: LaTeX.
    - Bakarrik lan egiteko gaitasuna indartzea.
    - Paketea editatu ahal izateko lengoaia eta programazio-ingurunea sakontzea: R lengoaia eta RStudios ingurunea hain zuzen ere.

2.1.2 Irakasgaiaren helburuak

Irakasgaiak ere hainbat helburu ditu proiektu honetan:

- Bilaketa Heuristikoak ikasten duten ikasleek, ikasitakoa praktikan jar dezaten, eta ulerterrazagoa suerta diezaieten, web-aplikazio bat izatea.
- Paketea lluzatzea.

2.2 Irismena

Aurretiaz azaldu bezala, proiektu honen helburua bikoitza da. Alde batetik, zenbait optimizazio-problemak eta optimizazio-algoritmoak aukeratzea da, ondoren Rn inplementatuz eta paketean integratuz; horrela kodea probatzeko algoritmoak esperimentalki alderatu ahalko

dira. Bestaldetik, esperimentuak sortzeko web-asistente bat sortzea da, era bisual batean problemak eta algoritmoak konfiguratzeko eta R kodea sortzeko. Beraz, helburu horiek kontutan hartuz, irismena definitu dugu eta hori horrela, betekizunak, mugak eta lan metodologia azaltzen ditugu.

2.2.1 Betekizunak

Proiektuak Donostiako Informatika Fakultateak zehaztutako arauak bete behar ditu. Hurrengo zerrendan garrantzitsuenak zerrendatu ditugu:

- Proiektua defendatzeko, graduko gainontzeko kreditu guztiak gaindituta izan behar dira.
- Gradu Amaierako Proiektuak 12 kreditu betetzen ditu; hau da, 300 orduko dedikazioa.
- Proiektuko zuzendaria Informatika Fakultateko irakaslea izan behar du.
- Proiektuak Informatika Fakultatean eskaintzen diren espezialitateko baten ingurukoa izan behar du; kasu honetan, konputazio espezialitatekoa.

2.2.2 Mugak

Esan bezala, proiektuak 300 ordu bete behar ditu eta, beraz, denbora mugatuta dugunez, atal bakoitzarentzat mugak jartzea komenigarri suertatu da. Hala ere, erabili beharreko teknologia berria denez, ezin dugu jakin zehatz mehatz zenbateko dedikazioa eskainiko zaion ulertzeari eta inplementatzeari; eta beraz, gerta daiteke muga hauen datak aldatu behar izatea.

Hala ere, Gradu Amaierako Proiektua denez, gutxienez 300 ordu lan egitea aurreikusten da.

2.2.3 Lan-metodologia

Proiektu hau, hasieratik adostu da era autonomoan garatuko dela, beraz, irakasleen laguntzarik gabe, ikasleak berak egingo du. Hala ere, oso beharrezkoa den garaian, argibideren bat behar izanez gero, zuzendariarengana joko du ikasleak zalantzak argitzeko.

2.3 Entregatu beharrekoak

Proiektua amaitzean, hauek dira entregatu beharreko emangarriak:

- Memoria: Proiektutik sorturiko emangarri nagusia da memoria. Dokumentu honek Gradu Amaierako Proiektu baten memoriaren gidalerroari jarraitu behar dio, formatu egokian egon behar du, eta egindako lan guztia azaldu behar du. Euskal Herriko Unibertsitateko ADDI plataformara igo behar da otsailaren 2a baino lehen, zuzendarien oniritzia jaso ondotik.
- Aurkezpena: Proiektuaren defentsa egiteko prestatutako gardenkiak, nahiz eta aurkezpena entregatzea ez den beharrezkoa.
- R paketea eta web-asistentea: Irakasgaiko arduradunari metaheuR R pakete eguneratua eta sorturiko web-aplikazioa eman behar zaio.

Gradu Amaierako Proiektua Jabego Intelektualaren Legeak babestuta dago; beraz, proiektuaren jabea sortzailea da (ikasleak kontrakoa esaten ez badu). Hortaz, memoriaren eta aurkezpenaren eskubide guztiak erreserbatuta daude.

2.4 Antolakuntza

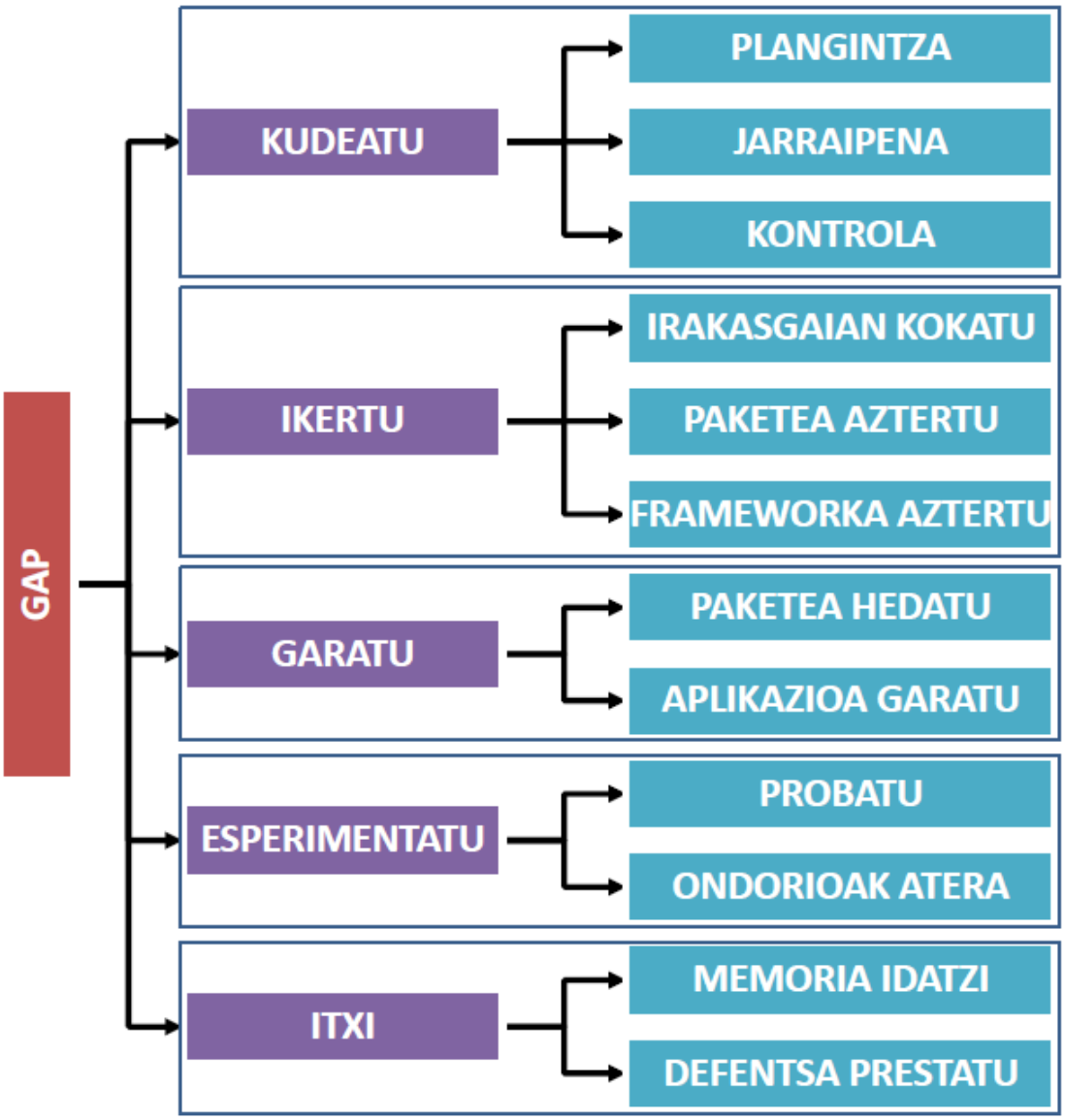
Hurrengo pausuan, Lanaren Deskonposaketa Egitura edo LDEa eraiki dugu, horretarako, aurretiaz helburuak finkatuak baititugu. Helburuetako ataza horiek zati txikiagotan banatu ditugu eta zati horiei aurreikusitako denbora bat balioetsi diegu diagramari laguntzeko.

2.4.1 Lanaren Deskonposaketa Egitura (LDE)

Azpiatal honetan, proiektuaren bizi-zikloa osatzen duten ataza nagusiak identifikatzen dira: kudeaketa, ikerketa, garapena, esperimentazioa eta proiektua ixtea. Ataza bakoitzak bere barnean hainbat zeregin izango ditu, proiektuaren LDEan ikus daitezkeenak, 2.1 irudian.

Kudeaketa

Kudeaketa hiru azpiataletan banatu dugu.



2.1 Irudia: LDE diagrama.

- Plangintza: Proiektua aurrera eramateko egin behar diren zenbait aurreikuspen: egutegia eta orduen estimazioa, beharrezko baliabideak aurreikustea, proiektuaren kalitatea definitzea, arriskuak aurreikustea eta kontingentzia-plana prestatzea.
- Jarraipena: Proiektuaren zuzendariekin egindako bilerek osatzen dute, nagusiki. Elkarrekin hartutako erabakien arabera plangintza eguneratu behar da.
- Kontrola: Lanaren kalitatea, arriskuen eta aldaketen kontrolarekin zerikusia duten atazak bilduko ditugu.

Ikerketa

Ikerketa atala honela banatuko dugu:

- Bilaketa Heuristikoak irakasgaiari zuzenkin loturiko proiektua denez, eta nik irakasgai hori eman ez dudanez, bertan ikasten den materiala ezagutzea proiektua garatu ahal izateko.
- metaheuR R lengoaiaz idatziriko paketea ezagutzea bertan problema eta algoritmo gehiago inplementatu ahal izateko.
- Shiny frameworka ezagutzea web-aplikazioa garatu ahal izateko.

Garapena

Garapenak bi zati desberdin izango ditu:

- Problema eta algoritmoak: metaheuR paketeari bi optimizazio-problema gehituko zaizkio eta horiek ebazteko algoritmo metaheuristikoak diseinatu eta inplementatuko dira.
- Shiny frameworka erabiliz web-aplikazio bat diseinatu eta inplementatuko da.

Esperimentazioa

Atal honetan, berriki sorturiko problemak nahiz aurretik inplementatuta zeuden problemak erabiliz esperimentazio bat egingo dugu. Honela, algoritmo ezberdinekin probatuz, lortu ditzakegun emaitzak alderatuko ditugu eta horietatik ondorioak atera.

Proiektua ixtea

Azkenik, amaierako atalean, bi multzo bereiztu ditugu:

- Memoria: Egindako lana, sortu diren arazoak eta hartutako erabakiak, proiektuan zehar ikasi dena eta ateratako ondorioak batzen ditu.
- Aurkezpena: Defentsa egiteko sortuko den materiala eta saioa prestatu.

2.4.2 Egutegia

Proiektuaren txostena entregatzeko azken eguna otsailaren 2a da. Beraz, ordurako proiektu osoak bukatuta egon behar du eta hurrengo bi asteetan, defentsa prestatzeko aukera izango dut. Plangintza zehaztuagoa eta egutegia hurrengo atalean azaltzen dira.

2.5 Proiektuaren atalak

Proiektua lau atal nagusitan banatu dugu: Hasieratzea, Plangintza, Gauzatzea eta Ixtea.

Hasieratzea: Atal hau, irakaslearekin proiektua hitzartzen den unean hasten da. Proiektuaren nondik norakoak, ulertu, adostu, zalantzak argitu eta zereginak definitzean datza.

Plangintza: Behin proiektua zehazturik, eta datozen ilabeteetan egingo duguna argi daukagularik, lortu nahi dugun emaitza ahalik eta hoberena izateko, planifikatzea da hurrengo eginbeharrekoa. Beraz, zereginak, atazak eta ekintzak identifikatu behar ditugu eta horiek betetzeko aurreikusten den denbora edo esfortzua balioetsi behar dugu. Horretarako egutegi bat egingo dugu ditugun datu guztiak biltzeko. Gainera, arriskuen plana ere egin behar da aurreikusitako arazoei aurre egin ahal izateko.

Gauzatzea: Plangintzan bildutako eginbeharrekorik nagusiena proiektua gauzatzea da, edo hobeto esanda, produktua. Hau da atal honetan egin beharrekorik nagusiena. Horretarako aurreko atalean esandako hainbat gauza hartu behar dira kontuan. Geure burua kokatzea da lehenengoz egingo duguna. Garapenarekin hasi aurretik, ikerketa bat egingo dugu beharrezko materialarekin, horrela behin garapenarekin gaudenean, zalantza gutxiago izango ditugu gaiaren inguruan. Behin prest gaudelarik, produktua garatzen has gaitezke. Prozesu osoan zehar, jarraipen eta kontrola egingo dugu, bukaeran ondorioak atera ditzagun.

Itxiera: Proiektuaren bizi-zikloaren amaieran, txostena idatzi eta entregatu behar dugu, aurkezpenaren prestaketarekin batera.

2.6 Proiektua gauzatzeko plana

Plan honetan, LDEa kontuan izanik, egutegi bat eta GANTT-diagrama bat egin dira.

2.6.1 Egutegia

Jarraian, 2.2 irudian, ataza nagusien karga ehunekotan ikus ahalko ditugu. 2.3 irudian, aldiz, zeregin bakoitzaren estimazioen informazio gehiago ikusiko dugu: beharrezko denbora, hasiera- eta bukaera-data.

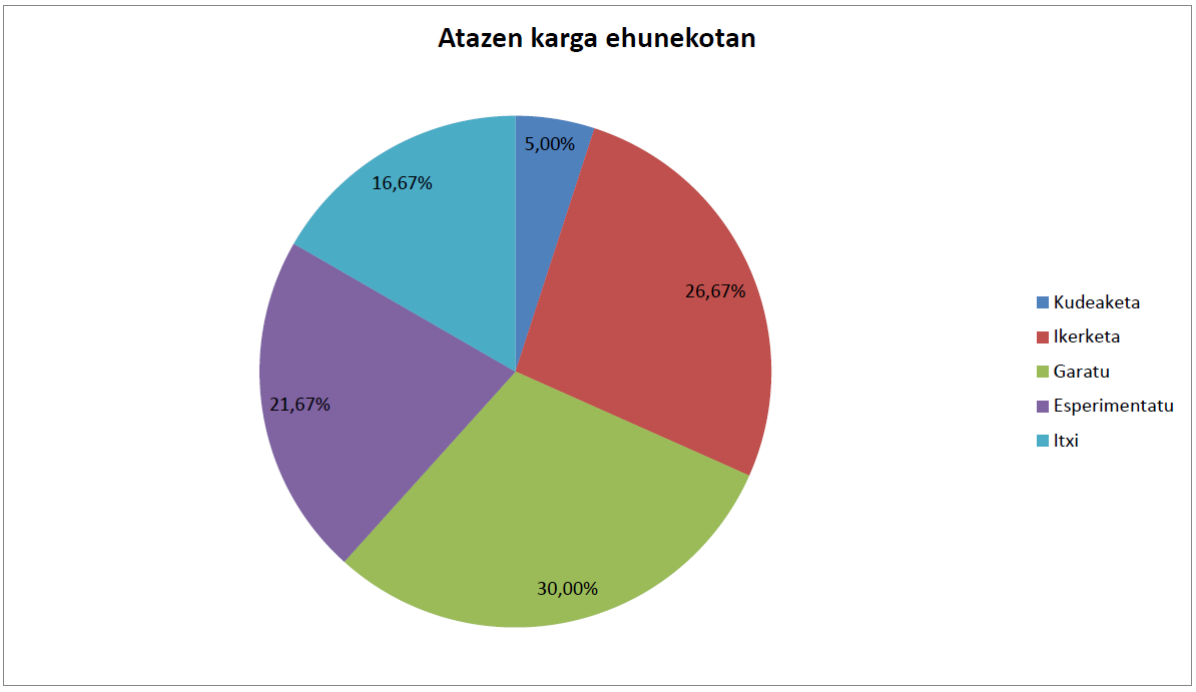
Bi grafiko hauetan fijaturik, garbi ikus daiteke kudeaketarako finkatu dugula kargarik txikiena, edo hobeto esanda, ordu kopururik txikiena. 2.3 taulan ikusten den bezala, ataza honen barruan plangintza, jarraipena eta kontrola kokatu ditugu. Hauek proiektua egiterako garaian oso garrantzitsuak diren arren, denbora oso gutxi eskaintzea komeni zaigu, beste atazei denbora gehiago eskaini ahal izateko. Kudeaketaren atal honek, proiektua hasten denetik, memoria entregatzen den arte iraungo du.

Hurrengo ataza, ikerketari dagokio; hemen, hiru azpiataza banatu ditugu. Hasteko eta behin, lehenengo eginbeharrekoa, Bilaketa Heuristikoak irakasgaian kokatzea izango da. Denbora asko eskaini ez arren, beharrezko denbora eskaintzea komenigarria litzateke, gainerako ataza eta azpiatazekin hasi aurretik. Bigarren eta hirugarren azpiatazak, proiektuaren helburuekin zuzenki erlazionaturik daude. Bi helburu dituenez, bi ataletan banatu dugu gainerako ikerketa ataza ere, helburu bakoitzarentzako azpiataza bat hobeto esanda: paketearen hedapena egin ahal izateko, paketearen azterketa eta aplikazioa garatu ahal izateko berriz, frameworkaren azterketa. Paketearen azterketari lan karga haundiagoa emango zaio frameworkaren azterketari baino, azkenean bata oso teorikoa izango baita eta denbora gehiago beharko baita berau ikasteko; bestea, berriz, inplementatu heinean ikasiko da.

Garapenaren atazari ikerketaren atazari baino denbora gehixeago estimatea estimatu dugu; hori bai, bata bestearen menpekoak dira, ikertu gabe ezingo baitugu garatu edo inplementatu. Paketearen ikerketa egin ondoren, atal hori garatzen has gaitezke, eta berdin frameworkaren atalarekin. Kasu honetan, metaheuR paketearen garapenari denbora gutxiago eskaintzea estimatu dugu, aplikazioa garatzeari baino.

Behin aplikazioa martxan dagoenean, esperimentazio ataza hasiko da. Ataza honi, ikerketari eta garapenari baino karga pixka bat txikiagoa estimatu diogu, azkenean garatzeko orduan ere probak egiten baitira. Ataza honek ordea, garrantziarik handienetakoa du egia esan proiektu honetan. Bertan, probak egingo dira, algoritmo desberdinekin problemak ebaztea lortuko da eta konparaketak egingo dira bildutako datuekin. Azkenik ondorioak aterako dira, bukaeran memorian azaldu ahal izateko.

Azken ataza itxiera deiturikoa dugu. Izenak dioen moduan, proiektua ixteko ataza da. Memorian, ordurarte hartutako apunteak, datuak, planak eta abar idatziko dira, egindako lan osoa ere azalduko delarik bertan. Ataza hau, gainerako dena egin ostean osatuko da, eta entrega eguna baino lehenago amaitua izan behar da. Ondoren, defentsa aurkezpena egiteko beharrezko materiala prestatzea ere ataza honetan sartu dugu. Beste ataza batzuk bezala, karga nahikoa izango du proiektu osoan zehar.



2.2 Irudia: Atazen karga ehunekotan.

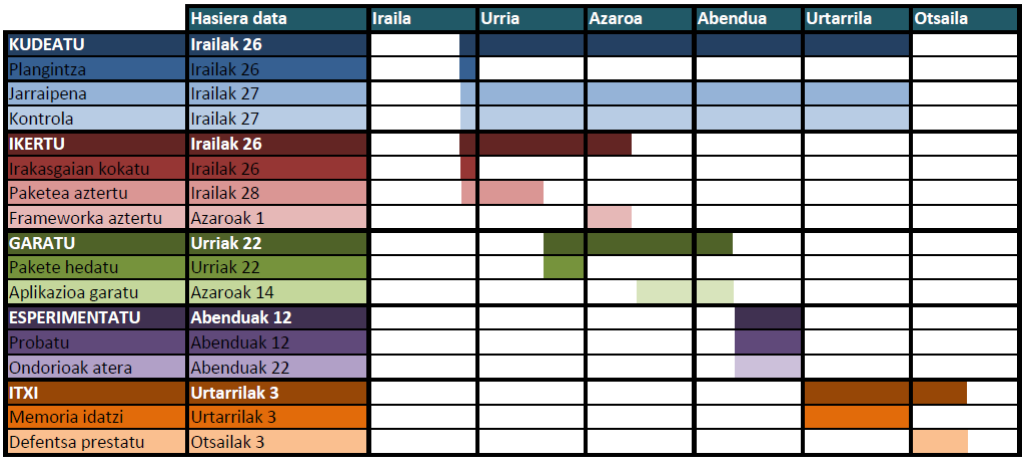
2.6.2 GANTT-diagrama

2.3 irudiko taulan ikusi ditugun datei jarraituz, proiektuko atazekin sortu dugun Gantt-diagrama 2.4 irudian ikus daiteke. Hemen, aurretik azaldutakoa, hobetoago ikusi daiteke,

Kodea	Ataza/Zeregina	Denbora	Hasiera	Bukaera
-	-	300	Irailak 26	Otsailak 14
1	KUDEATU	15	Irailak 26	Otsailak 1
1.1	Plangintza	4	Irailak 26	Irailak 26
1.2	Jarraipena	7	Irailak 27	Otsailak 1
1.3	Kontrola	4	Irailak 27	Otsailak 1
2	IKERTU	80	Irailak 26	Azaroak 13
2.1	Irakasgaian kokatu	10	Irailak 26	Irailak 27
2.2	Paketea aztertu	50	Irailak 28	Urriak 21
2.3	Frameworka aztertu	20	Azaroak 1	Azaroak 13
3	GARATU	90	Urriak 22	Abenduak 11
3.1	Pakete hedatu	30	Urriak 22	Urriak 31
3.2	Aplikazioa garatu	60	Azaroak 14	Abenduak 11
4	ESPERIMENTATU	65	Abenduak 12	Urtarrilak 2
4.1	Probatu	58	Abenduak 12	Abenduak 31
4.2	Ondorioak atera	7	Urtarrilak 1	Urtarrilak 2
5	ITXI	50	Urtarrilak 3	Otsailak 14
5.1	Memoria idatzi	35	Urtarrilak 3	Otsailak 1
5.2	Defentsa prestatu	15	Otsailak 3	Otsailak 14

2.3 Irudia: Jarraipen taula.

diagraman datak eta koloreek lagundurik. Estimazioen eta benetako lan-orduen konparazioa Jarraipen eta Kontrola kapituluan ikus daiteke.



	Hasiera data	Iraila	Urria	Azaroa	Abendua	Urtarrila	Otsaila
KUDEATU	**Irailak 26**						
Plangintza	Irailak 26						
Jarraipena	Irailak 27						
Kontrola	Irailak 27						
IKERTU	**Irailak 26**						
Irakasgaian kokatu	Irailak 26						
Paketea aztertu	Irailak 28						
Frameworka aztertu	Azaroak 1						
GARATU	**Urriak 22**						
Pakete hedatu	Urriak 22						
Aplikazioa garatu	Azaroak 14						
ESPERIMENTATU	**Abenduak 12**						
Probatu	Abenduak 12						
Ondorioak atera	Abenduak 22						
ITXI	**Urtarrilak 3**						
Memoria idatzi	Urtarrilak 3						
Defentsa prestatu	Otsailak 3						

2.4 Irudia: Gantt-diagrama.

2.7 Kalitate-plana

Proiektuaren kalitate maila minimoa betetzen dela ziurtatzeko eta estimatutako egutegia bete nahi denez, izan ditzakegun mehatxuak kontutan hartzea ere beharrezkoa da, horrela, proiektuak kalitaterik gal ez dezan. Horrexegatik, hainbat arrisku aurreikusi ditugu eta arrisku horiei soluzioa proposatu diegu:

2.7.1 Arriskuak kudeateko plana eta Kontingentzia plana

- Ordenagailua matxuratzea: Ordenagailuak duen denbora dela eta, edozein momentutan funtzionatzeari utz dezake. Horri aurre egiteko aukera bat lehenbailehen beste ordenagailu bat lortzea litzateke.

- Aurreko arazoak, hainbat arazo ekar ditzake, baina garrantzitsuena, ordurarte egindakoa galtzea litzateke. Hori gerta ez dadin, segurtasun kopiak egitea beharrezkoa da; hala-nola, kodea Github plataforman gordetzea erabaki da, bertsioen arteko kontrola eta antolaketa ere erraz dezakegu honekin eta dokumentazioari dagokionez, berriz, Google Driven.

- Irismen aldaketak izatea: Hau gerta ez dadin, komenigarria da hasieran irismena eta helburuak ahalik eta ondoen zehaztea.

- Teknologia berriak ez ulertzea: Honi aurre egiteko modurik egokiena, informazio gehiago aztertzea eta ikertzea litzateke, baina kontutan izanda, behar baino denbora gehiago ez pasatzea horretan. Bestalde, azken aukera bezala, tutoreari edo beste norbaiti laguntza eskatzea litzateke.

- Denbora falta: Estimatutako denbora baino gehiago pasaz gero atazen batean, gainerako atazei ez eragitea lortu beharko dugu eta hala balitz, denbora gehiago eskeini beharko diogu behar duen atazari.

Proiektuaren bizi-zikloan zehar aurreko atalean aipatutako arazoren bat gertatzen bada, proiektuaren irismena arriskuan jarriz, lehenengo egitekoa arazoa konpontzen saiatzea da. Esate baterako, programazio-arazorik izanez gero, inguruko norbaiti laguntza eska diezaiokegu, edo, makinaren batek funtzionatu ezean, beste makina batean jarri.

Arazoa edozein dela ere, proiektuaren zuzendariarekin bilera bat antolatu behar da, arazoari buruz hitz egin eta konponbidea aurkitzeko.

2.8 Proiektuaren hartzaile edo interesatuak

Proiektuak maila desberdinetako interesatuak ditu. Hasteko, interesatu nagusia ni neroni naiz, nire eginbeharra baita proiektu hau burutzea eta emaitza erabilgarriak lortzea.

Bigarrenik, Bilaketa Heuristikoak irakasgaiari lotutako irakasle eta ikasleak daude. Proiektu honen emaitza interesatzen baitzaie, izan ere, emaitzak erabilgarriak baitira beraientzat.

Beste aldetik, proiektu honen zuzendaria dago, Borja Calvo hain zuzen ere. Proiektua zuzentzea da bere betebeharra eta horretaz gain, behar izanez gero, zalantzak argitu eta emaitza arrakastatsua dela ziurtatzea.

Azkenik, proiektua ebaluatuko duten, irakasleek osatutako epaimahaia dago. Hauek proiektua kalifikatuko dute eta ikaslearekin izango duten kontaktu bakarra defentsaren egunean izango da.

Interesatu hauez gain, gerta litekeena da, beste norbaiti gaia interesatzea eta proiektuari berari buruz informatu nahi izatea, defentsaren egunean edo beste momenturen batean.

2.8.1 Komunikazio plana

Proiektuan zehar zuzendariarekin komunikatu ahal izateko, aurrez aurreko bilerak egitea adostu dugu. Gainera, bilera horiek deitzeko edo zalantza sinpleak argitzeko posta elektronikoa erabiltzea ere erabaki dugu.

3. KAPITULUA

metaheuR paketea eta haren hedapena

Kapitulu honetan, proiektuaren zatirik nagusiena izango da gai nagusia. Bertan, proiektuak izan dituen atalak azalduko dira. Esan bezala, proiektuak eginbeharreko bi helburu nagusi izan ditu. Bata *metaheuR* paketea hedatu, eta bestea aplikaziotxo bat diseinatu eta inplementatu.

Hasteko beraz, *metaheuR* paketea izango da protagonista nagusia, bertan, paketearen ezaugarriak, bertako problemak eta ebazteko dauden algoritmoak azalduko ditugu.

Ondoren, inplementatu ditugun problema berrien nondik norakoak argituko ditugu; beraien nolakotasunak, zertarako erabiltzen diren, eta batez ere beraien inplementazioa.

Azkenik, paketean bertan inplementatu dugun web-asistentearen garapena eta erabilera azalduko dugu, lortu ditugun emaitzak irudietan azalduz.

3.1 Paketearen deskribapena

metaheuR irakaskuntzan erabiltzeko erreminta bat da, hain zuzen ere, Euskal Herriko Unibertsitateko, Informatika Ingeniaritzako Graduko, Bilaketa Heuristikoak irakasgairako garatutako pakete bat da. Pakete hori ez da erabiltzen benetako optimizazio problemak ebazteko, bertako algoritmoak ez baitira inplementatu efizienteenak izateko, baizik eta ikasleak problema horiek erabiliz, eta probak eginez irakasgaia ikasteko.

Proiektu originala *GitHub* plataforman dago eta proiektua garatzen hasteko, *fork* bat egin nuen nire biltegian: https://github.com/oniar/metaheuR . *GitHub* horrelako proiek-

tuak taldean inplementatzeko eta konpartitzeko plataforma egokia da; beraz, kodea inplementatuz joan naizen heinean, bertan gordetzen joan naiz proiektua.

Paketean bi elementu nagusi daude: problemak eta algoritmoak. Jarraian elementu hauek aztertuko ditugu.

3.1.1 Problemak

Ikerketaren atalean, *metaheuR* paketea izan zen aztertu beharreko zatirik handiena. Paketeak optimizazio-problema ezberdinak inplementatzen ditu eta ondorengo taulan problema garrantzitsu edo ezagunenen hainbat ezaugarri azalduko dira.

Problema	**Adibideak**	**Ezaugarriak**
Garraio-problemak	Travelling Sallesman Problem	Merkantzien edota pertsonen garraioarekin zerikusia duten optimizazio-problemak
Esleipen-problemak	Quadrantic Assignment Problem	Matematikako eta, batez ere, Ikerkuntza Operatiboko oinarrizko problemak dira.
Azpimultzo-problemak	Motxilaren problema	Aukeraketak ematen dizkigun onurak maximizatzean datza, definitutako murriztapenak betez
Grafoei buruzko problemak	Grafoen koloreztatze-problema, Minimum Dominating Set, Maximum Independent Set	Garraiobide- eta esleipen-problemez gain, sareko informazio-trukaketa edota grafoen teoriako problema ugari ebazteko erabili ohi dira.

3.1 Taula: Paketeko problema garrantzitsuenen taula.

3.1.2 Problemen inplementazioaren ezaugarriak

Jakina den bezala, problema hauek denak, R lengoaiaz inplementaturik daude, beraz, R lengoaiaren arauak jarraitzen dituzte.

Ikertu ditugun heinean, hainbat ezaugarri dituztela ohartu ginen, eta dakigunez, ezaugarri horiek beharrezkoak izan dira guk inplementatu ditugun problema berrien diseinuan.

Pakete honetako problemek diseinu estrategia bat jarraitzen dute: problemak funtzio baten bidez inplementatuta daude, non, funtzio horrek, funtzio-zerrenda bat itzultzen duelarik.

Problema guztiek, *evaluate* funtzioa dute inplementaturik, hau da, soluzioa ebaluatzen duen funtzioa. Adibidez, grafoa koloreztatzearen problemak, kolore kopurua itzuliko itzuliko liguke; TSP motakoek ibilbidearen distantzia, eta motxilaren problemak soluzioan aukeratu ditugun elementuen balioa.

Horrez gain, *valid* eta *correct* funtzioak ere badituzte problema batzuk. Lehenengoak, soluzioa bideragarria denetz itzuliko du eta emaitza negatiboa izanez gero, zuzentzeko funtzioa da aipatutako bigarren funtzio hori. Badira beste azpifuntzio osagarri batzuk dituzen problemak ere, hala nola *plot* funtzioa, emaitzak bistaratzeko. Esan bezala beraz, problemen funtzio orokorrek, azpifuntzio hauen zerrenda itzultzen dute.

3.1.3 Algoritmoen ezaugarriak

Paketean, hainbat algoritmo metaheuristiko daude inplementaturik eta , problemen *evaluate*, *valid* eta *correct* funtzioak izanez gero, edozein problema ebatz daitezke algoritmo hauek erabiliz.

Ondorengo hauek dira, beraz, paketean ditugun metaheuristikak:

Algoritmoak
Ant Colony Optimization
Estimation of Distribution Algorithms
Algoritmo Genetikoa
Bilaketa Lokala
Particle Swarm optimization

3.2 Taula: Paketeko algoritmo mehaheuristikoen taula.

Problema eta algoritmo hauek denak azterturik, probaturik eta inplementazioa ulerturik, geure problema berriak diseinatu eta inplementatu ditugu.

3.2 Problema berriak eta algoritmo erakitzaileak

Proiektu honetan inplementatu ditugun problema berriak, sekuentzia edo *String* motako problemak dira. Hau da, instantziak zein soluzioaksekuentzia multzokoak dira. Bi proble-

ma izan dira inplementatu ditugunak eta bata bestearen osagarri dira. Ondoren bakoitzaren eraikitzaileak inplementatu ditugu.

3.2.1 Problemak

Bi problema inplementatu ditugu, Closest String Problem (CSP) eta Farthest String Problem (FSP). Bi problema hauetan sekuentzia multzo bat izango dugu. Lehenengoan helburua sekuentzia horiei ahalik eta gertuen dagoen sekuentzia bat topatzea da. Bigarrenean berriz, sekuentzia horiei ahalik eta urrunen dagoen sekuentzia bat topatzea da helburua.

Jarraian bi problema hauek formalizatuko ditugu

Definizioa 3.1. ***Closest String Problem (CSP)*** *Izan bitez A alfabetoa eta $\mathcal{S}_n$ alfabeto horrekin lor daitezken n tamainako sekuentzien multzoa. Izan bedi $d : s_n \times s_n \to \mathbb{R}$ bi sekuentzien arteko distantzia neurtzeko funtzioa. CSPan $S = \{s_1, s_2, ..., s_m\}, s_i \in \mathcal{S}_n$ sekuentzia multzoa dugu eta $s^* \in \mathcal{S}_n$ sekuentzia bilatu nahi dugu, non s^* $max\{d(s^*, s_i) \forall s_i \in S\}$ minimizatzen duen sekuentzia den.*

Guk distantzia hori kalkulatzeko, *Hamming* distantzia erabili dugu.

FSP problemak berriz, distantziarik txikiena itzuliko digu.

Definizioa 3.2. ***Farthest String Problem (FSP)*** *Izan bitez A alfabetoa eta $\mathcal{S}_n$ alfabeto horrekin lor daitezken n tamainako sekuentzien multzoa. Izan bedi $d : s_n \times s_n \to \mathbb{R}$ bi sekuentzien arteko distantzia neurtzeko funtzioa. FSPan $S = \{s_1, s_2, ..., s_m\}, s_i \in \mathcal{S}_n$ sekuentzia multzoa dugu eta $s^* \in \mathcal{S}_n$ sekuentzia bilatu nahi dugu, non $s^* min\{d(s^*, s_i) \forall s_i \in S\}$ maximizatzen duen sekuentzia den.*

Inplementazioari dagokionez, *valid* funtzioa bi problema hauek berdina daukate. Hemen, alfabetoa da kontutan hartuko dena batez ere. Soluzioan, alfabetoan jarri ez dugun karaktereen bat egonez gero, soluzioa ez da bideragarria izango, ez CSPn ez eta FSPn.

evaluate ebaluazio funtzioak ere, berdinak ez, baina antzekoak dituzte, aldatzen dena zera da, lehen esan bezala, CSPk distantzia maximoa itzultzen du eta FSPk berriz minimoa. FSPren helburua maximizatzea da, baina *metaheuR* paketean algoritmoak minimizatzeko inplementatuta daude eta, hortaz, helburu funtzioari zeinua aldatu diogu. Horrexegatik, FSPk distantzia minimoa itzuliko digu baina negatiboan.

3.2.2 Algoritmo eraikitzaileak

Problema hauek inplementatu ondoren, problemak ebazteko eraikitzaile bana diseinatu genuen, ondoren, paketeko *BasicLocalSearch* eta *BasicGeneticSearch*ekin alderatuko ditugularik.

CSPrako erabili dugun heuristikoa zera da: Geuk sortu nahi dugun *String* berria ahalik eta gertuen egotea nahi dugu gainerako sekuentzietatik. Horrexegatik beraz, posizio bakoitzean agertzen diren karaktereetan fijaturik, posizioan gehien errepikatzen diren karaktereak ezartzea iruditu zaigu egokiena. Horrela beraz, sortuko dugun String berriaren posizio bakoitzean, beste Stringetan, posizio horretan gehien errepikatzen den alfabetoko karakterea ezarriko dugu. Gerta daiteke ordea, kopuru berdinean errepikatzea karaktere hori, beraz, horrelakoetan, zoriz aukeratutako karakterea ezarriko dugu posizio konkretu horretan. Jarriak algoritmoaren sasikodea ikus daiteke.

Procedure 1 CSP Greedy

1: **function** CSPGREEDY **Input:** s_1, ..., s_m ; **Output:** s ▷ s_1 eta s_m azpindizeak dira, definizioan agertzen diren moduan
2: $n = luzera(s1)$
3: **for** $i = 1$ to n **do**
4: $s[i] = gehien\_errepikatzen\_dena(s_1[i], ..., s_n[i])$
5: **end for**
6: **end function**

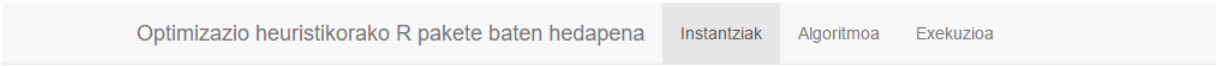
Bigarren problemarako, FSPrako eraiki dugun eraikitzailearentzako heuristikoa ere antzekoa da. Baina kasu honetan, gutxien errepikatutako karakterea ezarriko da. Hemen, gerta daiteke, alfabetoan aurretik sekuentzian agertu ez zaigun karaktereen bat egotea eta beraz, gutxien errepikatutakoa bera izango denez, bera jarriko da posizio honetan. Horregatik, garrantzitsua da alfabetoa zehaztea mota honetako problemetan. Aurrekoan bezala, gutxien errepikatutako kantitateak berdinak badira, karakterea zoriz ezarriko da.

3.3 Interfazea

Problema berriak inplementatu ondoren, proiektuaren bigarren zatiari ekin diogu, hau da, web-asistentearen diseinua eta garapena egin dugu. Horretarako, [Chang et al., 2016]*Shiny* aukeratu dugu, bera baita *R* ingurunean web aplikazioak garatzeko sortu den frameworka. Horrela beraz, *Shiny framework*ari buruz ikertu ondoren, tutorearekin egindako bilera

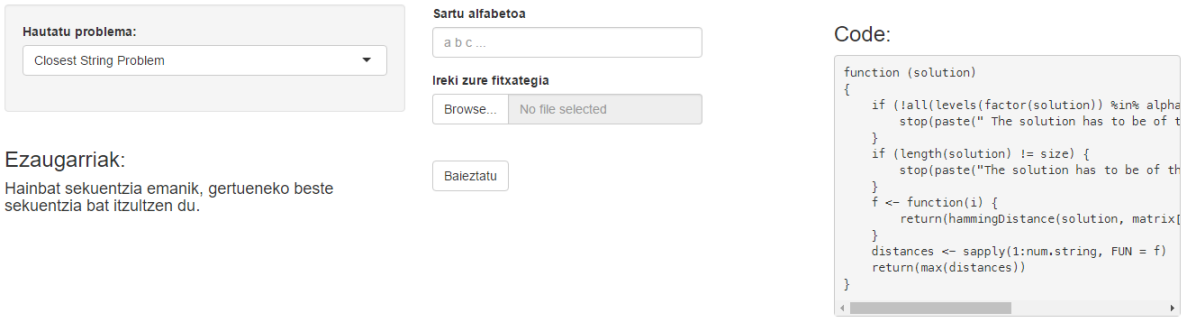
batean, interfazearen diseinua nolakoa izango zen adostu zen. Hasieratik esan zen aplikazioa sinplea izango zela, problema gutxi izango zituela aukeratzeko eta ebazteko, baina, problema gehiago gehitu ahal izateko modukoa izan behar zela.

Interfazeari begiratuz, hiru atal izango dituela erabaki zen. Lehenengo atalean problema sortuko da. Bigarren atalean, ebazteko algoritmoa aukeratuko da, eta horrekin batera algoritmoak behar dituen parametroak zehaztuko dira. Azkenik, hirugarren atalean, exekutatu egingo da algoritmoa; exekuzioari, mugak (denbora, ebaluazio kopurua eta iterazio kopurua) ezarriko zaizkio.



3.1 Irudia: Gure web-asistentean ditugun erlaitzak.

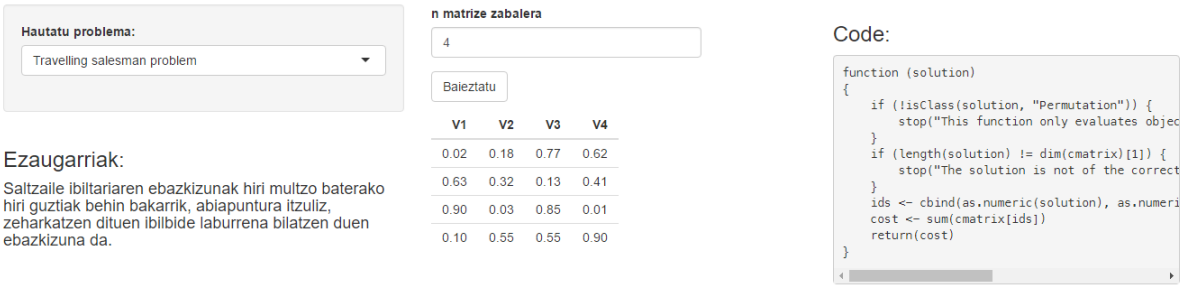
Hiru erlaitz ditu gure aplikazioak 3.1, bat lehen aipatutako atal bakoitzeko. Lehenengo, problema aukeratzen da. Aplikazioaren lehenengo bertsio honetan, geuk inplementatutako String problemak eta Travelling Salesman Problem (TSP) problema klasikoak eskuragarri daude.



3.2 Irudia: String problemak.

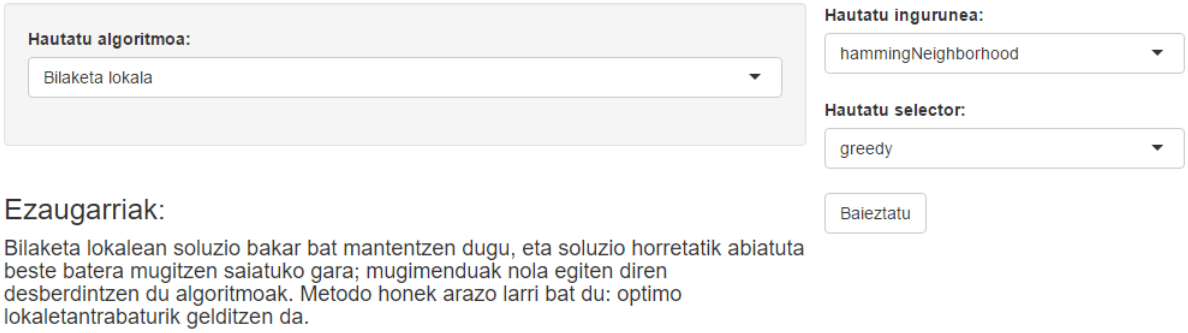
Ebatzi behar den instantziak zehazteko, CSP eta FSP kasuetan 3.2, sekuentziak fitxategietatik kargatzeko moduan inplementatu da asistentea. Hau da, Stringak fitxategi batean idatzi eta aplikazio honetatik kargatu ahalko da. TSPan berriz 3.3, matrizea zoriz sortzea erabaki da, erabiltzaileari lana erraztearren, honek, hiri kopurua izango den matrizearen luzera soilik adierazi beharko du.

Aplikazioa konplexuago bihurtuz bezala, problema denek, aukera denak izango lituzkete instantziak sortzeko, bai zoriz sortzea bai fitxategitik kargatzea.



3.3 Irudia: TSP problemak.

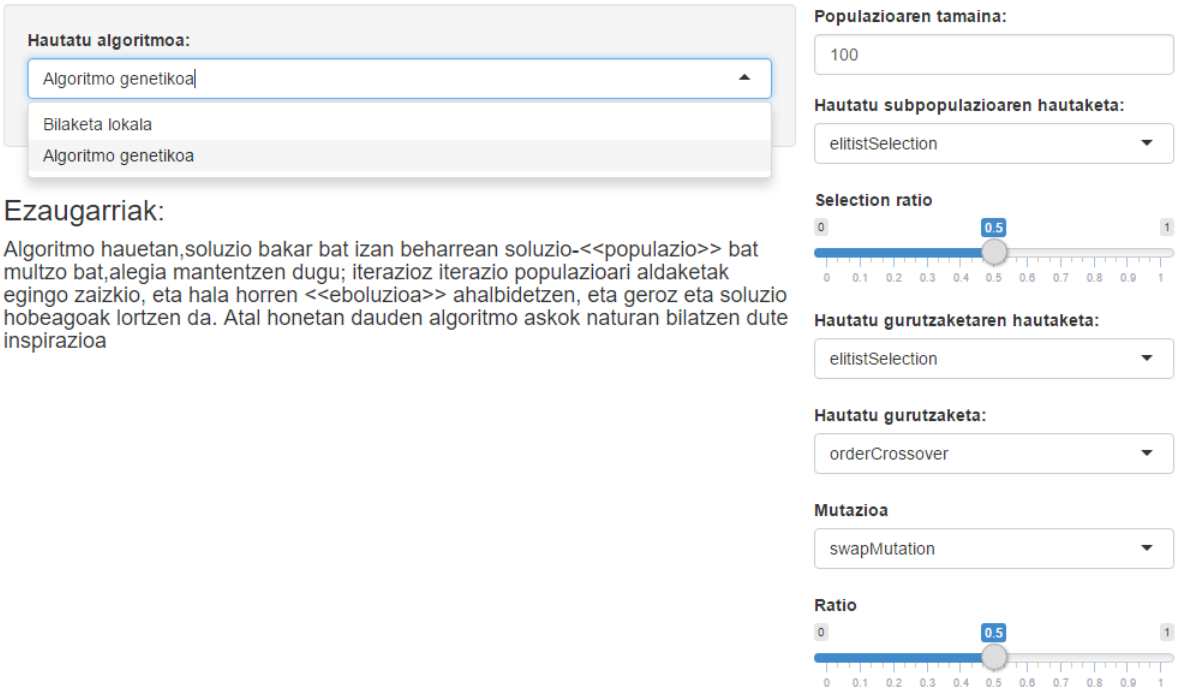
Bigarren erlaitzean, ebazpenerako algoritmoa aukeratzen da. Gure aplikazioak bi algoritmo ditu hautagai: Bilaketa Lokala eta Algoritmo Genetikoa. Hauen parametroak hurrengo kapituluan azalduko diren arren, esan beharra dago, algoritmoaren arabera finkatu behar diren parametro ezberdinak direla. Bi algoritmo horien interfazea 3.4 eta 3.5 irudietan ikus dezakegu.



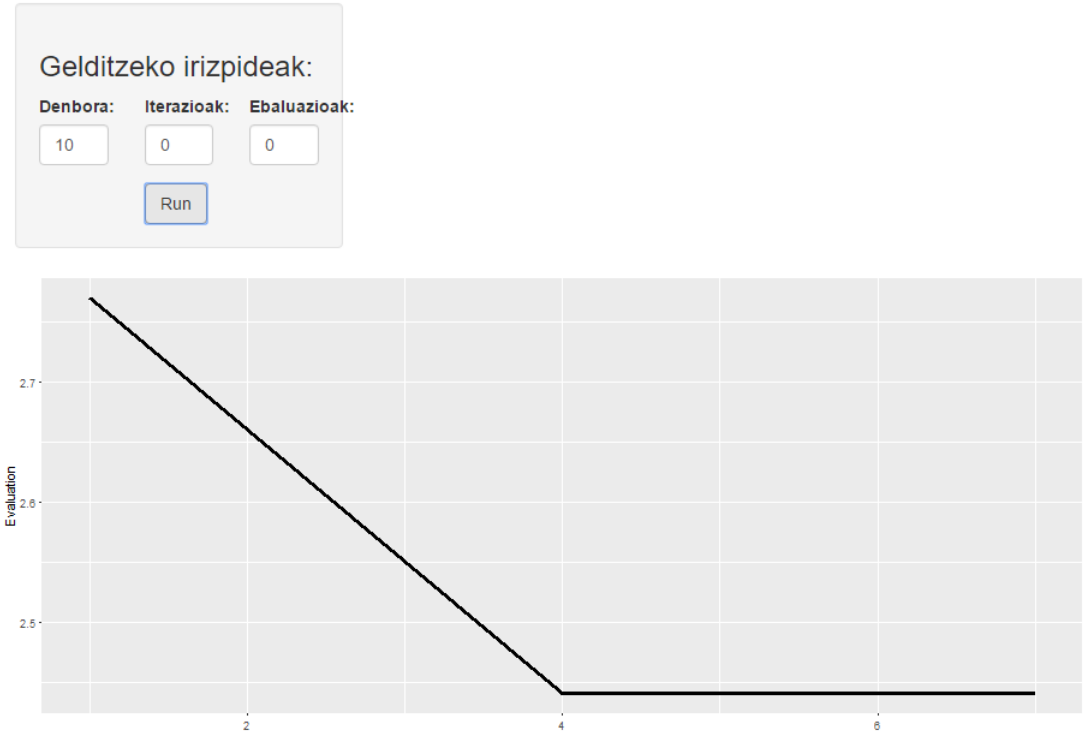
3.4 Irudia: Bilaketa lokala.

Azkeneko erlaitzean, exekuzio leihatilan alegia 3.6, hiru baldintzak sartzeko gelaxkez gain (denbora, iterazio eta ebaluazio kopurua), exekutatu edo *run* botoia gehitu da. Behin horrela, botoi horri eman ondoren, algoritmoa martxan jarriko da eta esandako baldintzak bete ostean, emaitzen grafikoa agertuko da pantailaren beheko zatian.

Aplikazioa ikasleei zuzenduta dagoenez, ikasleek berarekin ikasi ahal izango dute. Hori horrela izan dadin, hainbat ezaugarri gehitu zaizkio aplikazioari. Alde batetik, problema aukeratzerakoan, aukeratutako problemaren ezaugarriak agertuko dira pantailan. Eta



3.5 Irudia: Algoritmo genetikoa.



3.6 Irudia: Exekuzioa.

problema sortu ostean, eskuineko atalean, problemaren kodea agertuko da. Horrela, aukeratutako problema, nola inplementatuta dagoen ikusi eta ikasi ahalko du ikasleak.

Amaitzeko, gauza bera gertatuko da algoritmoak leihatilan. Aukeratzen duen algoritmoaren arabera, bakoitzak dituen ezaugarriak azaltzen dira, eta parametroak aukeratzeko dituenez, algoritmoak nola sortzen diren ikasi ahalko du ikasleak.

4. KAPITULUA

Esperimentazioa eta erabilera kasua

Kapitulu honetan, aurreko kapituluan deskribatutakoa martxan jarriko dugu. Aurretik esan bezala, problema berriak erabiliz esperimentazio bat egingo dugu eta hori dela hizpide, erabilera kasua aztertuko dugu. Esperimentazio honen helburua kodea probatzea izan da, hau da, ez da izan algoritmorik hoberena zein den erabakitzea edo algoritmoen portaera aztertzea.

Lehenik eta behin, esperimentazioan erabilitako problemen instantziak azalduko ditugu, eta ondoren egindako esperimentazioa deskribatu eta lortutako emaitzak aztertuko ditugu.

4.1 Instantziak

Instantziak problemaren instantziazioak dira, hau da, CSP eta FSP kasuetan sekuentzia multzo bat.

4.1.1 Erabili ditugun instantziak

Aurreko kapituluan aipatu bezala, bi motatako problemekin egin dugu lan aplikazioa garatzerakoan, baina, String-problemak izango dira kapitulu honetan erabiliko ditugunak, beraz, sekuentzia motako problemen instantziak erabiliko ditugu.

CSP eta FSP instantziak:

Problema hauek sekuentzia-problemak dira. Nahi hainbeste sekuentzia sartu ahal izango dizkiogu problemari, hori bai, sekuentzia denak luzera berekoak izan beharko dute. Sekuentziez gain, alfabeto bat sartuko dugu; alfabetoa sekuentzietako karaktereek osatu beharko dute gutxienez, baina karaktere gehiago ere sar daitezke alfabetoan.

Gure aplikazioan, fitxategi batetik kargatzeko aukera ezarri dugu. Fitxategi honetan, lerroka sartuko ditugu stringak; lerro bakoitza sekuentzia bat izanik. Esan bezala, lerro guztiek tamaina bera izan beharko dute. Alfabetoa ordea, eskuz sartu beharko da sisteman, karaktere batetik bestera espazio batez banaturik.

Problema hauek, geuk inplementatu ditugunez, beraien erabilerari garrantzia eman nahita, mundu errealeko datuak erabiltzea adostu genuen probak eta esperimentazioa egiteko.

CSP eta FSP problemak DNA sekuentziak aztertzeko bereziki interesgarriak dira (nahiz eta problema hauek sinplifikazioak izan). Hori dela eta instantziak sortzeko DNA sekuentziak erabiltzea erabaki dugu.

Konparagarriak diren tamaina bereko DNA sekuentziak biltzea ez da erraza DNA genomikoan eta, horregatik, DNA mitokondriala (DNAmt) erabili dugu.

Laburki, gure zelula guztietan, gure DNA genomikoaz gain, mitokondriak deritzen egitura batzuk daude. Mitokondriek DNA sekuentzia motzak dituzte, DNAmt. DNA horrek populazioen genetikarako oso interesgarria den ezaugarri bat du, amarengandik soilik heredatzen da. Hori dela eta, amaren leinua ikertzeko oso informazio erabilgarria da.

DNAmt sekuentziaren tamaina 16k ingurukoa da, baina posizio gehienak berdinak dira gizaki guztietan. Izan ere, bakarrik bariabilidade handiko zatiak erabiltzen dira, HVR1, HVR2 eta HVR3 deritzonak; zati horien tamaina 300 da gutxi gora behera.

Guretzat bereziki interesgarriak dira hiru puntu:

1. HVR zatien tamaina finkoak dira eta, hortaz, gizaki ezberdinen sekuentziak tamaina berekoak izango dira.

2. Jatorri berdineko gizakiak sekuentzia antzerakoak izango dituzte, DNAmt oso egonkorra baita.

3. Badaude DNAmt duten datu base publikoak.

Gure proba egiteko http://www.mtdb.igp.uu.se/ web-orriko datu-baseetatik Kanariar Uharteetako sei pertsonen sekuentziak lortu ditugu eta horrekin CSP eta FSP instantzia bat sortu dugu. Instantzia honetan sei sekuentzia ditugu 360 tamainakoak.

4.2 Algoritmo eraikitzailearekin esperimentatzen

Gure esperimentazioaren lehenengo proba algoritmo eraikitzaileekin egin dugu. Sortutako instantziari bi algoritmoak (CSPrako eta FSPrako) 10 aldiz aplikatu dizkiegu.

CSParen kasuan beti soluzio berdina lortzen dugu. Horrek esan nahi du posizio guztietan gehien errepikatzen den karakterea bakarra dela. Ebaluazioa 252 da, hau da, urrutien dagoen sekuentziak 108 posizio ditu ezberdin algoritmoak itzultzen duen soluzioarekiko. Soluzioa ona den edo ez jakiteko hurrengo atalean metaheuristikoekin lortutako soluzioekin alderatuko dugu.

Gauza bera egin dugu FSParen eraikitzailearekin. Kasu honetan, errepikapen guztietan soluzioa ezberdina da, baina beti 359 distantziara dagoen soluzioa da.

Honek zentzua dauka, instantziako sekuentzia guztiak Kanariatik datozenez antzerakoak izango dira eta, hortaz, guztietatik urrun dagoen sekuentzia bat topatzea erraza da. Dena dela, proba honek laguntzen digu algoritmoa ondo inplementatuta dagoenetz ikusten.

4.3 Shinyrekin esperimentatzen

Diseinatu eta inplementatu dugun web-aplikazio edo asistente honen bidez, Bilaketa Lokala eta Algoritmo Genetikoa erabiliz, Sekuentzia-problemak ebatzi ditugu. Atal honetan, horiekin esperimentatu ondoren lortu ditugun emaitzak aditzera emango ditugu. Ondoren, algoritmo horiekin lortutako emaitzak, guk inplementatutako eraikitzaileekin lortutako emaitzekin konparatuko ditugu.

Orduan beraz, aurretik erabili ditugun sei pertsona horien DNA sekuentziekin esperimentatu dugu. Emaitzak aurkeztu baino lehenago, algoritmo bakoitzerako behar ditugun argumentuak azalduko ditugu eta ondoren izandako emaitzak tauletan azalduko ditugu.

4.3.1 Probatutako algoritmoak eta beren parametroak

Bilaketa Lokala

Bilaketa lokala aplikatzeko hiru gauza izan behar ditugu kontutan: hasierako soluzioa, ingurune definizio bat eta inguruneko soluzio bat aukeratzeko prozedura. Hiru elementu hauek *initial.solution, neighborhood* eta *selector* parametroen bidez ezarri ditugu.

initial.solution edo hasierako soluzioa zoriz sortzea erabaki dugu. Hasierako soluzio hori, ingurune objetua sortzeko erabiliko da ondoren.

Objetu hori *Hamming* distantzian oinarritutako ingurune objektua izango da, Hau da, *hammingneighborhood ()* funtzioari hasierako soluzioa sartuko diogu parametrotzat eta honela *neighborhood*a sortuko dugu.

Azkenik, inguruneko soluzio bat aukeratzeko prozeduretako bat aukeratuko dugu: *firstImprovementSelector()* funtzioa edo *greedySelector()* erabili ditzakegu. Lehenengo funtzio honek, uneko soluzioaren ingurunea arakatu du eta *fitness*a hobetzen duen lehenengo soluzioa itzultzen du. Bigarren funtzio horrek berriz, beti inguruneko soluziorik onena aukeratzen du. Amaitzeko, esan beharra dago lehenengo bi parametroak finko daudela aplikazioan, baina azkena erabiltzaileak aukeratzen duela.

Algoritmo Genetikoa

Algoritmo Genetikoa aplikatzeko garaian, gauza gehiago hartu behar dira kontuan. Hauexek dira hain zuzen ere, aplikazioari sartu beharko dizkiogun datuak:

Populazioaren hasieratzea da lehenengo sartu beharreko datua, gure kasuan, populazioaren tamaina sartuko dugu soilik, eta tamaina honekin zoriz sortuko dugu hasierako populazio hori.

Bigarrenik, soluzioen hautespena egingo dugu eta segur aski urratsik garrantzitsuenetakoa izango da, honek kontrolatzen baitu populazioaren eboluzioa. Hiru motatakoa izan daiteke: hautespen elitista, populazioan dauden soluziorik onenak hautatzeko; erruleta-hautespena, non, soluzioen kalitatearekiko proportzionala den probabilitatearekin hautatzen duen eta azkenik, lehiaketa-hautespena, lehenengo indibiduo guztietatik azpi-multzo bat guztiz zoriz aukeratu eta ondoren azpi-multzo honetatik soluziorik onena hautatzen dutena.

Hautespen hauek, subpopulazioaren aukeraketa egiteko nahiz gurutzaketaren aukeraketa egiteko aukerak izango dira, bientzako berdinak alegia. Gurutzaketa aipatuta, sekuentzia-problemek funtzio bat erabiltzen dute horretarako, *kPointCrossover* alegia eta funtzio horrek k balio bat behar du zenbat gurutzaketa puntu izango dituen adierazteko.

Mutazioa da hurrengo pausua. Populazioa eboluzionatu ahal izateko soluzioak desberdinak izatea berebizikoa da, populazioak dibertsitatea mantentzea, alegia. Hori dela eta, behin gurutzaketa-operadorearen bidez soluzio berriak lortzen ditugunean, ausazko aldaketak eragin ohi dira mutazio operadorearen bidez. Gure kasuan, sekuentzia-problemek, *factorMutation* funtzioa erabiltzen dute horretarako. Honek, argumentu bat du, 0tik 1era doana, zenbat posizio aldatu nahi den adierazteko. Mutazio hau, probabilitate batekin aplikatzen zaie sortu berri diren soluzioei.

Azkenik, gelditze irizpideak adierazi beharko ditugu, guk esan ezean algoritmo genetikoa ez baita geratzen eta eboluzionatzen jarraitzen baitu amaigabe.

4.3.2 Lortutako emaitzak

Exekuzioan, algoritmoen parametro bakoitzarekin hamar proba egin dira, tauletan bildutako emaitzak, lortutako txikienak dira. 4.1 irudiko taulak bilaketa lokalarekin lortutako emaitzak jasotzen ditu eta 4.2 taulak, berriz , algoritmo genetikoarekin lortutako emaitzak, lehenengo fasean.

	CSP		FSP	
	firstImprovementSelector	greedySelector	firstImprovementSelector2	greedySelector3
Emaitza	242	248	282	278
Denbora	29,75161	40,07567	14,3374	24,57235
Ebaluazio kopurua	15845	31321	8510	18361
Iterazio kopurua	39	28	17	16

4.1 Irudia: Bilaketa lokalaren emaitzak CSPrekin.

4.2 taulan, $K = 2$ izanik, 3 minutuan lortutako emaitzak ikus daitezke, populazioa 100 delarik eta $Ratio = 0,5$; $Selection.Ratio = 0,5$ eta $Mutation.Rate = 0,5$ izanik. Honela, ikus daiteke 203 dela lorturiko emaitzarik txikiena.

Horrela beraz, baldintza horiek betetzen dituenekin, hau da, subpopulazioa hautatzeko metodoa elitista eta gurutzaketa egiteko metodoa erruletarena izanik, eta berriz ere, $K = 2$ izanik, 3 minutuan lortutako emaitzak ikus daitezke, populazioa 100 delarik proba gehiago egin ditugu balio minimoagoa lortzeko asmoz.

Subpopulazioa				Gurutzaketa			
		Roulette					
Elitist	Tournament	Rank BAI	Rank EZ	Elitist2	Tournament2	Roulette	Emaitza
x				x			232
	x			x			243
		x		x			243
			x	x			240
x					x		205
	x				x		222
		x			x		220
			x		x		236
x						x	203
	x					x	219
		x				x	219
			x			x	251

4.2 Irudia: Algoritmo genetikoaren emaitzak CSPrekin, lehenengo fasean.

Ratio	Selection ratio	Mutation Rate	Emaitza
0,01	0,5	0,5	158
0,99	0,5	0,5	237
0,01	0,01	0,5	167
0,01	0,99	0,5	162
0,01	0,25	0,5	154
0,01	0,75	0,5	157
0,01	0,25	0,01	212
0,01	0,25	0,99	144

4.3 Irudia: Algoritmo genetikoaren emaitzak CSPrekin, bigarren fasean.

Ratio, Selection.Ratio eta Mutation.Rate parametroak banan-banan limiteetara emanez probatu ditugu. Ratio parametroari adibidez, balio txikiena emanik, emaitza hobea lortu dugu; beraz, Ratioren balio txiki hori mantenduz, Selection.Ratio parametroarekin egin ditugu probak, eta berdin Mutation.Raterekin. Emaitzak 4.3 taulan ikus daitezke.

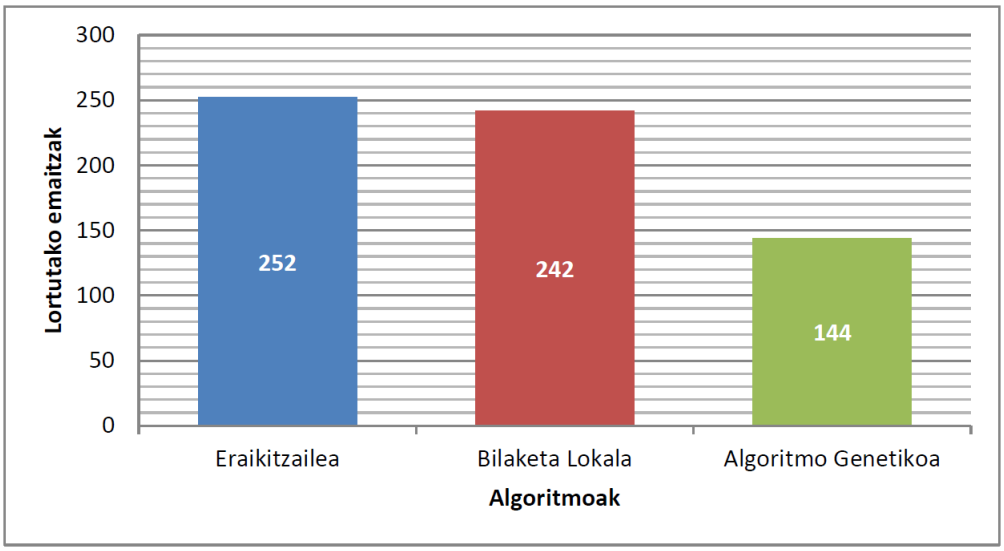
Soluziorik onena beraz, subpopulazioa hautatzeko metodo elitistarekin, gurutzaketa metodoa erruletarekin, $Ratio = 0,01$, $Selection.Ratio = 0,25$ eta $Mutation.Rate = 0,99$ izanik lortu dugu. Emaitza 144 izan da.

Baldintza horiekin, K balioa igoz, populazioa aldatuz eta denbora gehiago jarriz, ez dugu emaitza hobetzerik lortu, beraz, bukaerako emaitza 144 dela finkatu dugu.

4.3.3 Ondorioak

Proba horiek denak egin ondoren, 4.4 grafikan emaitzarik onenak bildu ditugu. Ikus daitekenez, emaitzarik onena, Algoritmo Genetikoa erabiliz lortu dugu. Beste bi algoritmoekin, bai gure eraikitzailearekin eta bai bilaketa lokalarekin, emaitza antzekoak lortu ditugula ikus daiteke.

Baina lehenago esan bezala, esperimentazio honen helburua, kodea probatzea izan da; guk inplementatutako eraikitzailearena eta Shiny erabiliz String problemena.



4.4 Irudia: Esperimentazioko ondorioen taula.

5. KAPITULUA

Ondorioak eta jarraipena

5.1 Jarraipena eta Kontrola

5.1.1 Proiektuaren garapena

Ez da proiektuaren osatzea arriskuan jarri duen atzerapen edo arazo larririk egon. Osotasunean, proiektua bukatzeko 300 ordu aurreikusita daude plangintzan. Hala ere, plangintzan aurreikusitako orduekin konparatuz, desbideraketak gertatu dira eta gutxi gora behera hamabost bat ordu gehiago sartu dira. Desbideraketa horiek 5.1 irudian ikus daitezke.

5.1.2 Kalitatea

Kalitateari dagokionez, irismena ongi bete dela esan dezakegu. Helburuak lortu dira eta emaitza onak izan ditugu.

Arriskuak

Plangintzaren 2.7.1 atalean, arrisku hauek aipatu ditugu:

- Hardware arazoak.
- Datuak galtzea.

Kodea	Ataza/Zeregina	Denbora	Erreala	
-	-	300	315	15
1	KUDEATU	15	20	5
1.1	Plangintza	4	5	1
1.2	Jarraipena	7	10	3
1.3	Kontrola	4	5	1
2	IKERTU	80	75	-5
2.1	Irakasgaian kokatu	10	5	-5
2.2	Paketea aztertu	50	45	-5
2.3	Frameworka aztertu	20	25	5
3	GARATU	90	105	15
3.1	Pakete hedatu	30	35	5
3.2	Aplikazioa garatu	60	70	10
4	ESPERIMENTATU	65	60	-5
4.1	Probatu	58	50	-8
4.2	Ondorioak atera	7	10	3
5	ITXI	50	55	5
5.1	Memoria idatzi	35	40	5
5.2	Defentsa prestatu	15	15	0

5.1 Irudia: Aurreikusitako-denborak eta denbora-erreala adierazten dira, desbiderapenekin batera

- Irismen aldaketak.
- Teknologia berriekin arazoak.
- Denbora falta.

Hardwarearekin arazo handirik izan ez dugun arren, batzuetan trabatuta geratu zaigu ordenagailua eta horrelako kasuetan berrabiarazi beharra izan dugu. Denbora asko galdu ez den arren, proiektuarekiko arreta galdu da horrelako kasuetan. Hala ere datuak ez dira inongo momentuan galdu, aurretik aipatu bezala, GitHuben eta Google Driven gordetzeaz gain memoria USB batean ere gorde baitira fitxategiak. Horretaz gain, irismenean ere aldaketarik ez da egon proiektuan zehar.

Bestalde, programazio arazoak ohikoak dira. Hasierako formalizazioa puntu garrantzitsua zen proiektu honetan, eta hori definitzeaz gain, inplementatzen hastea ataza konplexua izan da. Hala eta guztiz ere, hasiera batean kasu konkretuekin eta datu txikiekin probak egin dira eta ondoren benetako datuekin probatu ahal izan dugu.

Azkenik, denbora arazo nagusia izan dela uste izan da. Atazak ongi kudeatu arren, inplementazio atalak aurreikusitako denbora baino gehiago eskaintzea eskatu du. Beraz, horrek atazen datak piska bat atzeratzera eragin du. Hala ere lortu behar izan ditugun helburuak ongi bete dira.

5.1.3 Ebaluazio pertsonala

Atal honetan, proiektuaren egikaritzapenari eta egindako lanari buruzko ebaluazio pertsonala egin dugu.

- Egindako lanarekin gustora gaude. Proiektu luze bat aurrera eramateko gai izan gara, arazo handiegirik gabe eta laguntzarik gabe, eta lortutako emaitzak positiboak dira. Gainera, emaitza horiek onuragarriak direla uste dugu. Hala ere, hainbat hobekuntza egin daitezke paketean, nahiz eta gure proiektutik kanpo gelditzen den.

- Lanaren banaketa ondo egin da. Hasieran esandako atazak bete dira, hau da, proiektuaren inplementazioaren zati bakoitzerako, bere ikerketa atala egin da, eta honen ostean inplementazioa burutu da.

- Aplikazioaren lehenengo bertsio hau, luzagarria da, beste edozein informatikarik, osa dezakeena edo hobetu dezakeena.

- Aplikazio hau informatikariez gain, algoritmoak ikasten dituzten beste graduetako ikasleentzat ere erabilgarria dela uste da, adibidez, gainerako ingenieritzetako ikasleentzako , matematika ikasketak egiten ari direnentzako edo enpresen administrazioa ikasten ari direnentzako.

- Dokumentazioari dagokionez, bertako kapituluak ongi azaldu direla uste da eta horri dagokion lanarekin ere gustora gaude.

5.2 Ondorioak

Kapitulu honetan, proiektuan lortutako emaitzen berrikuspena egiten dugu. Gainera, etorkizunean egin litezkeen hobekuntzak edo jarraipen-lana aipatzen ditugu. Bestalde, ikasitako lezioak ere bertan aurki daitezke.

5.2.1 Lortutako emaitzak

Proiektuan zehar lortutako emaitzak ikusita, ondorio batzuk atera ditugu. Lortutako emaitzetatik honakoak ondorioztatu ditugu:

metaheuR paketeko problema berriak

Proiektu honetan, ezezaguna zen pakete bat aztertu, ezaugarriak bildu eta ezaugarri horiek jarraituz, problema berriak inplementatzea lortu dugu.

Web-aplikazioa

Web-aplikazioari dagokionez, teknologia berri bat erabiliz, Shiny frameworka erabiliz alegia, aplikazioaren lehen bertsio sinple bat inplementatzea lortu dugu. Martxan jarri dugu eta problema berriak ebazteko erabiltzea lortu dugu.

5.2.2 Etorkizunerako lana

Proiektuaren iraupenean zehar ahal beste egin badugu ere, sistema hobetzeko aukera ugari daude. Beraz, atal honetan etorkizun batean egin daitekeen lana azaltzen dugu, sistemaren ahalmena eta malgutasuna areagotuz:

metaheuR paketea

Dakigunez, *metaheuR* paketea nahi bezainbeste luza daiteke, problema berriak sar baitaitezke. Hortaz, esan daiteke pakete hau inoiz ez dela bukatutzat emango. Hori horrela, proiektu honek aurrera jarrai dezake alde horretatik.

Shiny eta web-asistentea

Beste aldetik, aplikazioa era sinple batean eginda dago; zerotik abiatu da eta prototipo bat izan daiteke, hau da aplikazioaren lehen bertsioa deitu genezaioke. Esan bezala, hiru problema bakarrik ditu hautagai, eta horiek ebazteko, algoritmo genetikoa eta bilaketa lokala soilik ditu.

Ideala izango litzakeena zera da, paketean dauden problema denak aplikazioan bertan erabilgarri egon ahal izatea, eta metodo heuristiko gehiago izatea horiek ebatzi ahal izateko. Horrela, Bilaketa Heuristikoak ikasten ari diren ikasleek irakasgaian ikasten diren problema denen exekuzioak ikus eta ikas ahal izango dituzte berau erabiliz.

Azkenik, diseinu aldetik ere, hobekuntzak jasatea ondo legoke, itxura aldetik estiloak erabiliz, koloreak ezarriz eta erakargarriagoa bilakatuz.

5.2.3 Ikasitako lezioak

Orokorrak

- Memoria idazteko garaian, testu guztia egituratzea pisutsua da. Horregatik, idazten hasi aurretik, eskema bat egitea komeni da memoriaren egiturarekin. Modu honetan, atal bakoitzean zer sartu behar dugun aurretik jakinda, askoz errazagoa izan zaigu modu errazago batean idaztea.
- Gutxi ezagutzen dugun edo ezagutzen ez dugun pakete batekin lanean hasi behar izan dugunean, beharrezkoa izan zaigu lehenik paketearen ideia orokor bat egitea, hau da nola dagoen inplementatuta begiratzea eta zehaztasunak argitzea. Ezaugarri horietan oinarrituta beraz, errazagoa izan zaigu problema berriak diseinatu eta inplementatzea.

Zehatzak

- Proiektu osoa kontuan hartuta, konputazio alorrean ikasten diren optimizazio-problemak diseinatzen eta inplementatzen ikasi dugu eta horiei erabilpen bat ematen.
- Problemaren definizioa argi eta garbi ulertu eta burutu behar dela ikasi dugu, garrantzitsua bait da ondoren ongi inplementatu ahal izateko.

Bibliografia

[Calvo et al., 2015] Calvo, B., Ceberio, J., and Mori, U. (2015). *BILAKETA HEURISTIKOAK*. Euskal Herriko Unibertsitatea (UPV/EHU).

[Calvo et al., 2014] Calvo, B., Mori, U., and Carreño, A. (2014). *metaheuR: Package including classical metaheuristics and combinatorial optimization problems*. R package version 0.2.

[Chang et al., 2016] Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2016). *shiny: Web Application Framework for R*. R package version 0.14.2.