eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Gradu Amaierako Lana

Informatika Ingeniaritzako Gradua

Konputazioa

Knot problemarako instantzia sortzaile eta boolean satisfiability solver bidezko ebazlea

Sua de la Cruz Odriozola

Zuzendariak
Juan Miguel López

2023.eko irailaren 17

Laburpena

Gradu Amaierako Lan honetan, “Dot Knot” problemaren instantziak sortzeko eta ebazteko erronkari heltzen zaio, irudiak abiapuntu gisa hartuz. Problema hauek bi ebazle desberdin erabiliz ebazten dira: pysat paketea eta ortools paketea. Bi ikuspegi horiek problema konplexuak ebazteko duten eraginkortasuna alderatzea da helburu nagusia. Beste aldetik, froga kasu ugariak eta esanguratsuak lortzeko arazo-sortzaile bat garatu da, arazoaren instantzien irudiak sortzen dituena. “Dot Knot” problema boolearra ez denez, beharrezkoa izango da “Dot Knot”-en gaienko SAT-erako laburketa burutzea.

Lan honetan arazo anitzen ebazpenaren paralelizaziorekin ere egingo da lan, eta konputazio-eskari handiko egoeretan ebazleen errendimendua ebaluatzen lagunduko duena.

Azken finean, ikerketa honek pysat eta ortools-en abantaila eta desabantailei buruzko informazio baliotsua emango du "Dot Knot"problemaren instantziak ebazteko garaian. Hortaz, mundu errealeko arazoetan SAT ebazleen ulermenean eta aplikazioan aurrera egiteko.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

Algoritmoen Zerrenda

KAPITULUA 1

Sarrera eta hasierako definizioak

Kapitulu honetan GrALari hasiera eman eta testuingurua azalduko da. Hortaz, oinarritzat hartu den jokoa, lanaren betekizun orokorrak eta problemaren definizioa azalduko dira.

1.1 Testuingurua

Gradu Amaierako Lan honetan egin dena ulertzeko, lehenik eta behin Dot Knot jokoa ulertu behar da. Problema hau, baita ezagutua "Flow Free" edo "Connect the Dots" izenekin besteak beste, logikako buru-hausgarri bat da. Lauki-sare bat izanda, hainbat laukietan kolore ezberdinetako puntuak daude, jokoaren helburua kolore guztiak elkartzea izanik. Jokoaren muina honek dituen baldintzetan dago: puntuak elkartzeko lerro horizontalak eta bertikalak baino ezin dira marraztu, taularen espazio osoa betez eta bata bestearekin gainjarri gabe. Mailetan aurrera egin ahala, zailtasuna areagotu egiten da taula handiagoekin, kolore desberdinetako puntu ugarirekin eta murrizketa berriekin, hala nola zubi-laukiekin edo hainbat koloretako puntuekin.

1.1.1 Betekizun orokorrak

Proiektuan egindakoak honako hauek dira:

1. Ebazlea: atal honen helburua jokoaren irudi bat jasota, irudi bera ebatzita itzultzea da.

    a) Irudi prozesaketa: jokoaren irudi bat hartuz, irudiko lauki-sarea, nodoak eta koloreak identifikatzea eta datu-egitura batean zehaztean datza. Hau instantzia sortu eta ebatzi ahal izateko funtsezkoa da.

    b) Instantziak ebazteko, SAT (Boolean Satisfiability) bidezko ebazle bat inplementatu da.

2. Instantzia sortzailea: "Dot Knot" problema ereduko instantzia handiak sortzea da helburu.

1.2 Problemaren definizioa

Dot Knot arazoa NP-osoa da, hau da, ez da aurkitu algoritmo eraginkorrik denbora polinomikoan soluzioa lortzen duena. Problemaren ordena jakiteko, lehenik algoritmoaren konplexutasunean eragina duten sarrerako aldagaiak ezagutu behar dira:

1. **m** eta **n**: Taularen dimentsioa adierazten dute, hau da, errenkada eta zutabe kopurua. Taularen tamainak eragin zuzena du koloreztatu beharreko laukiengan: $m \times n$ dimentsiotako lauki-sare batean, $m \cdot n$ laukien kolorea aztertu behar da.

2. **k**: Taulako puntuen kolore kopurua adierazten du. Problemaren murrizketak kontuan hartuta, horrek esan nahi du taulan $2k$ puntu daudela. Kasurik txarrenean, kolore-konbinazio posible guztiak kontuan hartu beharko lirateke baliozko soluzio bat aurkitzeko (jada koloreztatuta dauden laukiak baztertuz). Beraz, lehen laukirako k aukera daude, bigarren laukirako $k$aukera, eta horrela hurrenez hurren. Guztira, $k^{m \cdot n} - 2k$ konbinazio posible daude lauki-sarea koloreztatzeko.

Hortaz, Dot Knot problemaren konplexutasuna $O(k^{(m \cdot n)})$ da. Adierazpen honek arazoaren izaera esponentziala islatzen duela sumatu dezakegu.

1.3 Laburketa

Dot Knot problema ebazteko SAT ebazleak erabiliko ditugunez, lehenengo pausoa problemen arteko laburketa egitea izango litzateke, hau da, aztertu behar da nola adieraziko diren Dot Knot jokuaren aldagaiak eta murrizketak SAT problemakoak izango balira. Laburketa SAT-erako problemara izanik, murrizketak forma normal konjuntiboan (CNF) adierazi behar dira. Adierazpen hau OR klausulen konjuntzioan datza, non klausula batean aldagai bat edo gehiago ager daitezken. Lehenik, problema orokorraren laburketa azalduko da, ondoren, kasu eta murrizketa berezien zatia, eta azkenik, laburketaren ulerpena laguntzeko adibide bat.

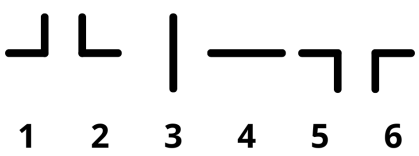
1.3.1 Oinarrizko laburketa

Murrizketen eraldaketa azaldu aurretik, laburketan erabiliko diren aldagaiak azaltzea beharrezkoa da.

Aldagai proposizionalak:

1. k kolore izanda, eta $m \times n$ dimentsiotako lauki-sarea izanda, $k \cdot m \cdot n$ aldagai behar ditugu. Hau da, lauki bakoitzeko kolore bakoitzeko literal bat. Beraz, honen adierazpena $x_{i,c}$ da, non i laukia eta c kolorea adierazten duten. Hortaz, $i \in 1..n \cdot m$ eta $c \in 1..k$ dira. $x_{i,c}$ aldagai bat True-ra ebaluatuko da baldin eta soilik baldin i laukiak c kolorekoa bada.

2. Puntua ez duten laukiak beste bi laukiekin kolore baten bitartez daude lotuta. Lotura hau adierazteko beste aldagai bat behar da. 1.1 irudian agertzen den bezala, 6 lotura ezberdin daude: goi-ezker, goi-eskuin, goi-behe, ezker-eskuin, ezker-behe eta eskuin-behe. Punturik ez duen i lauki bat izanda, s norabidea izanda: $y_{i,s}$ aldagaia dugu, non $i \in 1..n \cdot m$ eta $s \in 1..6$ diren. $y_{i,s}$ aldagai bat True-ra ebaluatuko da baldin eta soilik baldin i laukiko konexioa s norabidekoa bada.



1.1 Irudia: Loturen adierazpen grafikoa

Murrizketa bakoitzeko hainbat klausula sortu behar dira, murrizketa CNF forman duen adierazpenaren arabera.

Klausulak:

1. Nodo bakoitzak kolore bakarra izan dezake. Murrizketa hau CNF forman adierazteko, nahikoa da lauki bakoitzeko kolore guztiak onartzen dituen klausula sortzea, eta ondoren kolore pare bakoitzeko gehienez kolore bat agertzea bermatzen duen klausula sortzea. Hau da:

   a) $(x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,c})$
   b) $(-x_{i,1} \vee -x_{i,2}) \wedge (-x_{i,1} \vee -x_{i,3}) \wedge \ldots \wedge (-x_{i,c-1} \vee -x_{i,c})$

2. Puntua duen nodo bakoitzak kolore bera duen inguruko lauki bakarra izan dezake. Hau da, aurreko kasuan bezala, nahikoa da lauki eta kolore bakoitzeko inguruko laukietan kolore bera onartzen dituen klausula sortzea, eta ondoren aurreko klausulako aldagai pare bakoitzeko gehienez bat betetzen dela bermatzen duen klausula sortzea. Hau da:

   a) $(x_{i-m,c} \vee x_{i-1,c} \vee x_{i+1,c} \vee x_{i+m,c})$
   b) $(-x_{i-m,c} \vee -x_{i-1,c}) \wedge (-x_{i-m,c} \vee -x_{i+1,c}) \wedge \ldots \wedge (-x_{i+1,c} \vee -x_{i+m,c})$

3. Puntu gabeko lauki bakoitzak kolore bakoitzeko aurretik azaldutako lotura bat izan behar du (alboko bi laukiekin). Klausula honekin bermatzen da puntu batetik besterako bidea egongo dela. Honek esan nahi du i laukian s norabidea izateak, s norabideko j eta k laukiek i laukiak duen c kolore bera izan behar dutela eta alderantziz. Hau da:

   a) Logikako adierazpena: $y_{i,s} \rightarrow ((x_{i,c} \iff x_{j,c}) \wedge (x_{i,c} \iff x_{k,c}))$
   b) CNF forman: $(-y_{i,s} \vee -x_{i,c} \vee x_{j,c}) \wedge (-y_{i,s} \vee x_{i,c} \vee -x_{j,c}) \wedge (-y_{i,s} \vee -x_{i,c} \vee x_{k,c}) \wedge (-y_{i,s} \vee x_{i,c} \vee -x_{k,c})$

4. Puntu gabeko lauki bakoitzeko, lotura bakoitzeko eta kolore bakoitzeko, loturaren parte ez diren laukiak ezin dezakete laukiaren kolore bera izan:

   a) Logikako adierazpnea: $y_{i,s} \rightarrow -(x_{i,c} \wedge x_{l,c})$
   b) CNF forman: $(-y_{i,s} \vee -x_{i,c} \vee -x_{l,c})$

1.3.2 Gehigarriak

Hemen azalduko dira problema orokorraz gain jokoetako maila batzuek dituzten murrizketa gehigarriak.

1.3.2.1 Hainbat koloretako puntuak

Murrizketa honek bermatzen du lauki batean puntu berak batetik laurako kolore izateak, puntu horrekin lau konexio arte baimentzen, beti ere kolore ezberdinetakoak izanik.

Klausulak:

1. Nodo bakoitzak kolore bakarra izateko antzeko klausulak sortu behar dira, baina ezeztatzerako kasuan puntu berean dauden koloreak ez dira ezeztatzen. Adibidez, i. laukiak 1 eta 3 koloreak baditu:

   a) $(x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,c})$

   b) $(-x_{i,1} \vee -x_{i,2}) \wedge (-x_{i,1} \vee -x_{i,4}) \wedge \ldots \wedge (-x_{i,c-1} \vee -x_{i,c})$

1.3.2.2 Zubia

Badaude lauki batzuk gurutze moduko loturak onartzen dituztenak, hau da, bi kolore pasa daitezke lauki beretik, haien bidea moztu gabe, zubi baten gainetik pasako balira bezala.

Klausulak:

1. Zubia duen laukiak bi norabide (goi-behe eta ezker-eskuin) izan behar ditu, beste norabide guztiak ezeztatuz.

   a) $(-y_{i,1}) \wedge \ldots \wedge (y_{i,3}) \wedge (y_{i,4}) \wedge \ldots \wedge (-y_{i,6})$

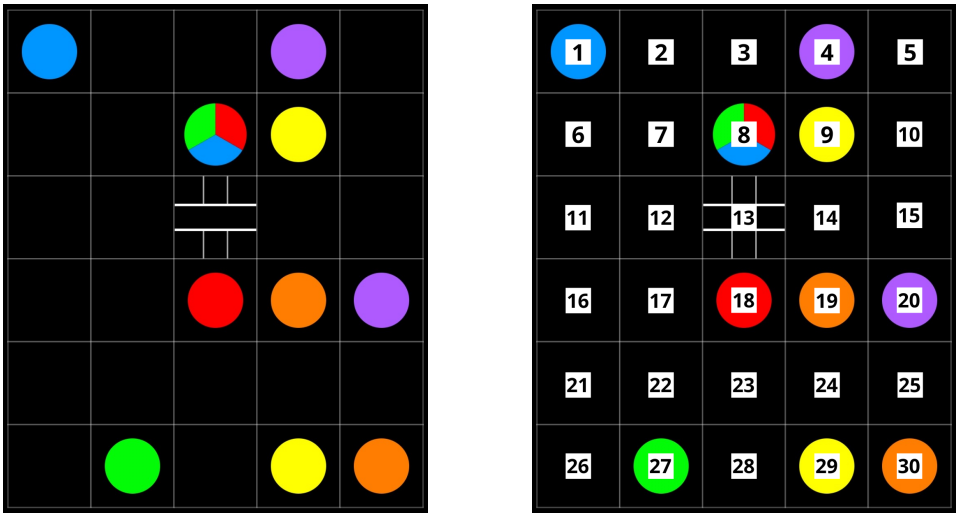
2. Lauki honen alboko zubirik gabeko laukiek, zubiaren norabidean dagoen hurrengo laukia hartuko dute kontuan bizilagun gisa.

3. Zubi laukia izanik, bi kolore izan behar ditu laukiak. Beraz, gutxienez kolore bat egon beharreko klausula sortu behar da bakarrik, kolore maximoa mugatu gabe. Bi norabideak derrigortuz laukiak bakarrik bi kolore izango dituela bermatzen da.

1.3.3 Adibidea

Aurretik azaldutako laburketa hobeto ulertzekoaren helburuarekin, "Dot Knot" jokoko problema erreal baten gainean egindako laburketa azalduko da. Adibide bezala hartuko dugun problema 1.2a irudikoa da. Irudian ikus dezakegun moduan, 6×5 dimentsiotako lauki-sarea da, 6 kolore dituena {1,2,3,4,5,6} (urdina, morea, berdea, gorria, horia, laranja).

Lehenik, aldagai proposizionalak erazagutzen hasiko gara: lauki bakoitzean zer kolore doan jakiteko, lauki bakoitzeko eta kolore bakoitzeko aldagaiak sortuko ditugu. Hau da, $6 \cdot 5 \cdot 6 = 180$ aldagai behar ditugu. Hortaz, goitik eta ezkerretik hasita, 1.2b irudian agertzen den bezala, lehenengo laukiaren kolorea adierazteko aldagaiak hurrengoak dira:



(a) "Dot Knot" instantzia **(b)** Laukiak zerrendatuta

1.2 Irudia: "Dot Knot" jokoaren adibidea

1. $x_{1,1}$: lehenengo laukia urdin kolorekoa dela adierazten du.
2. $x_{1,2}$: lehenengo laukia more kolorekoa dela adierazten du.
3. $x_{1,3}$: lehenengo laukia berde kolorekoa dela adierazten du.
4. $x_{1,4}$: lehenengo laukia gorri kolorekoa dela adierazten du.
5. $x_{1,5}$: lehenengo laukia hori kolorekoa dela adierazten du.
6. $x_{1,6}$: lehenengo laukia laranja kolorekoa dela adierazten du.

Kasu honetan, lehenengo laukiak puntua duenez, jada badakigu $x_{1,1}$ aldagaia egiazkoa den bakarra dela. 8. laukiaren kasuan, hiru koloretako puntua duenez, hiru aldagai izango lirateke egia. Beste aldetik, puntu gabeko edozein beste lauki normal batean ez dakigu zein aldagai den egia, hori ebazlearen lana baita klausulen bidez ateratzea.

Azaldutako bigarren aldagai motak loturen norabidea adierazteko balio dutela esan dugu. Beraz, 6×5 dimentsiotako lauki-sarea eta sei norabide izanda, beste $6 \cdot 5 \cdot 6 = 180$ aldagai erazagutu behar ditugu. Hortaz, bigarren laukiaren konexioen norabideak adierazteko hurrengo aldagaiak ditugu:

1. $y_{2,1}$: bigarren laukiak goi-ezker lotura duela adierazten du.
2. $y_{2,2}$: bigarren laukiak goi-eskuin lotura duela adierazten du.
3. $y_{2,3}$: bigarren laukiak goi-behe lotura duela adierazten du.
4. $y_{2,4}$: bigarren laukiak ezker-eskuin lotura duela adierazten du.
5. $y_{2,5}$: bigarren laukiak ezker-behe lotura duela adierazten du.
6. $y_{2,6}$: bigarren laukiak eskuin-behe lotura duela adierazten du.

13. laukiaren kasuan, zubi-lauki bat denez, badakigu jada bi norabide egiazkoak direla: $y_{13,3}$ eta $y_{13,4}$.

Behin aldagaiak erazagututak ditugula, lehen murrizketaren klausula multzoarekin hasiko gara. Lehenengo murrizketak dio lauki batek kolore bakarra izan dezakeela, hortaz lehenengo laukiarentzan hurrengo klausulak sortzen dira:

1. $(x_{1,1} \vee x_{1,2} \vee x_{1,3} \vee x_{1,4} \vee x_{1,5} \vee x_{1,6})$
2. $(-x_{i,1} \vee -x_{i,2}) \wedge (-x_{i,1} \vee -x_{i,3}) \wedge \ldots \wedge (-x_{i,5} \vee -x_{i,6})$

Bigarren murrizketak dio puntu batek izandako laukiak bakarrik inguruko lauki batek izan dezakeela kolore bera. Hau da, lehenengo laukiaren kasuan urdin kolorekoa dela, inguruko beste laukietatik bakarra izan daiteke urdina. Kasu honetan, lauki-sarearen iskin batean dagoenez, bakarrik bi lauki-bizilagun ditu:

1. $(x_{2,1} \vee x_{6,2})$
2. $(-x_{2,1} \vee -x_{6,1})$

Hirugarren murrizketak dioen bezala, puntu gabeko laukiek alboko bi laukiekin lotuta egon behar dute. Lotura hau gertatzeko, alboko bi lauki horiek eta uneko laukiak kolore bera izan behar dute. Adibideko 12. laukia hartuko dugu adibide moduan, gorri kolorearekin eta goi-behe norabidearekin. Hortaz, 12. laukia goi-behe norabideko lotura badu, 7., 12. eta 17. laukiek kolore bera izango dute. Esan dugunez, adibidea gorri kolorearekin egingo dugu:

1. Logikako adierazpena: $y_{12,3} \rightarrow ((x_{12,4} \iff x_{7,4}) \wedge (x_{12,4} \iff x_{17,4}))$
2. CNF forman: $(-y_{12,3} \vee -x_{12,4} \vee x_{7,4}) \wedge (-y_{12,3} \vee x_{12,4} \vee -x_{7,4}) \wedge (-y_{12,3} \vee -x_{12,4} \vee x_{17,4}) \wedge (-y_{12,3} \vee x_{12,4} \vee -x_{17,4})$

Kasu honetan problema ebazten badugu, ikusi dezakegu 12. laukia goi-behe norabidea izatea ezinezkoa dela. Hortaz, klausulak egiazko balioa izango lukete, nahiz eta gorri kolorekoa ez izan. Beste aldetik, egiazkoa den norabidea hartzen badugu, aztertutako hiru laukiak gorri ez direnez, klausulak egia izaten jarraituko lukete. Horrela ulertu dezakegu klausulak beti egiazkoak izango direla, eta hauei esker benetako norabide eta koloreen aldagaiak egiazkoak izaten bakarrak izango direla.

Laugarren murrizketarekin jarraituz, badakigu norabide batetik kanpo dagoen alboko laukiak ezin duela uneko laukiaren kolorea izan. Beraz, aurreko adibide bera jarraituz, bai 11. laukia bai 13. laukia gorri ez den beste kolorekoa izan behar dira:

1. Logikako adierazpnea: $(y_{12,3} \rightarrow -(x_{12,4} \wedge x_{11,4})) \wedge (y_{12,3} \rightarrow -(x_{12,4} \wedge x_{13,4}))$
2. CNF forman: $(-y_{12,3} \vee -x_{12,4} \vee -x_{11,4}) \wedge (-y_{12,3} \vee -x_{12,4} \vee -x_{13,4})$

Klausulak aztertuz, ikus dezakegu norabidearen aldagaia egiazkoa izatekotan, koloreen aldagaietako bat faltsua izatera derrigortzen duela, uneko laukiak eta bizilagun laukiak kolore ezberdina dutela bermatuz.

Behin oinarrizko murrizketak azalduta, murrizketa gehigarriak ikusiko ditugu. Lehenengo murrizketa berezia puntu batek hainbat kolore izatearena da. Adibidearen kasuan, 8. laukian dugu hiru koloreko puntua. Beraz, badakigu puntu honetan kolore urdina (1), berdea (3) eta gorria (4) egon daitezkeela. Hortaz, kolore hauek batera ez agertzeko klausulak ez dira sortu behar:

1. $(x_{8,1} \vee x_{8,2} \vee x_{8,3} \vee x_{8,4} \vee x_{8,5} \vee x_{8,6})$
2. $(-x_{8,1} \vee -x_{8,2}) \wedge (-x_{8,1} \vee -x_{8,5}) \wedge \ldots \wedge (-x_{8,2} \vee -x_{8,6}) \wedge (-x_{8,3} \vee -x_{8,5}) \wedge \ldots \wedge (-x_{8,5} \vee -x_{8,6})$

Azkenik, zubiaren murrizketa dugu. Kasu honetan, 13. laukian zubi bat dugu, beraz badakigu lauki horretan bi kolore pasako direla goi-behe (3) eta ezker-eskuin (4) norabideekin. Hortaz:

1. $(-y_{13,1}) \wedge (-y_{13,2}) \wedge (y_{13,3}) \wedge (y_{13,4}) \wedge (-y_{13,5}) \wedge (-y_{13,6})$

Nahiz eta adibide honetan kasu zehatzekin ibili garen, gogoratu behar da azaldutako murrizketa guztiak lauki, kolore eta norabide guztiekin egin behar direla.

KAPITULUA 2

Proiektuaren kudeaketa-plana

2.1 Irismena

Gradu Amaierako Lan honen helburua web aplikazio bat sortzea da, non "Dot Knot" problema SAT-era laburketa eginez era eraginkorrean ebazten den, liburutegi berriekin lan eginez.

2.1.1 Proiektuaren helburu zehatzen deskribapena

Jarraian, proiektu honek biltzen dituen helburu zehatzak deskribatzen dira:

1. **Irudi Prozesaketarako Programa baten Garapena:** Proiektuaren helburuetako bat "Dot Knot" jokoaren aplikazio desberdinetatik datozen irudiak prozesatzeko gai izango den programa bat sortzea da. Programa hau aldakorra izan beharko du eta "Dot Knot" arazoaren edozein irudi instantzia bihurtzeko gai izan beharko du.
2. **"Dot Knot" Problemaren SAT-erako Laburketa:** "Dot Knot" problema SAT (Murrizketa Boolearren Gogobetetzea) problema batera modu eraginkorrean murriztea bilatzen da. Murrizketa horri esker, SAT ebazleak erabili ahal izango dira "Dot Knot" jokoaren instantziak eraginkortasunez ebazteko.
3. **SAT problemetarako Python-eko Liburutegien Ikerkuntza:** Proiektuaren helburua da SAT arazoak konpontzeko erabiliko diren PySAT eta OR-Tools-en CP-SAT liburutegiak ezagutzea eta horietan sakontzea. Bere gaitasunak, ezaugarriak eta bere aplikazioa "Dot Knot" jokoaren testuinguruan ikerketa barne.
4. **Instantzia Sortzailearen Garapena:** Garatutako programaren bidez, "Dot Knot" arazoaren instantzien irudiak automatikoki sortzea gaitu nahi da. Instantzia horien dimentsioak parametro gisa zehaztu ahal izango dira, eta horrek tamaina handiako adibideak sortzea erraztuko du.
5. **Paralelizazioaren bitarteko Errendimenduaren Optimizazioa:** Azken helburua "Dot Knot" problemaren ebazpenaren errendimendua hobetzea da. Hori lortzeko, paralelizazio-teknikak ezarriko dira, instantzia handiagoak modu eraginkorragoan ebazteko.

2.1.2 Suposaketak

Egin den suposaketa bakarra "Dot Knot" problematik SAT-erako laburketa egitea posible izango dela da. Antzekoak diren laburketetan oinarrituz suposatu da laburketa hau posible izango dela. Hots, grafo baten 3-koloreztatzetik SAT-erako laburketa hartu da abiapuntu gisa.

2.1.3 Emangarriak

Enpresa kanpotik egindako proiektu bat denez, sortutako emangarri guztiak proiektuko zuzendariari eta EHUri entregatu beharreko emangarriak dira bakarrik. Emangarriak hurrengoak dira: sortutako web-aplikazioa (EDOTKNOT), lanaren garapena azaltzen duen memoria hau (EMEM), proiektuaren posterra (EPOS), eta lanaren defentsarako garatu beharreko aurkezpena (EAURK). Hurrengo 2.1 taulan azaltzen dira emangarrien identifikadoreak, azalpenak eta entrega datak.

Identifikadorea	Deskribapena	Entrega-data
EDOTKNOT	Web-aplikazioa	2023/09/05
EMEM	GrALaren memoria	2023/09/17
EPOS	GrALaren posterra	2023/09/17
EAURK	Defentsaren aurkezpena	2023/09/25

2.1 Taula: Emangarrien taula

2.1.4 LDE

Proiektua modu antolatuenean egiteko Lanaren Deskonposaketa Egitura (LDE) diagrama osatu da. Hots, hainbat lan-pakete definitu dira antolaketa egituratua izateko: Ezagutza, Produktua, Kudeaketa eta Dokumentazioa. Hona hemen 2.1 irudian agertzen diren lan-paketeen azalpenak:

Ezagutza

1. **Teknologikoa (T)** lan-paketeak proiektua garatzeko beharrezkoa diren teknologien ezagutza eskuratzeko atazak hartzen ditu barne.

Produktua

1. **Laburketa (L)** lan-paketeak "Dot Knot"problematik SAT-erako laburketa egiteko prozesua biltzen du.
2. **Aplikazioaren garapena**
   a) **Irudi Prozesaketa (IP)** lan-paketeak irudien prozesaketarekin zerikusia duten funtzioen garapena hartzen du barne.
   b) **Laburketa Programatu (LP)** lan-paketeak laburketa programatzeko beharrezkoak diren prozesuak hartzen ditu barne.

c) **Instantzia Sortzailea (IS)** lan-paketeak instantzia sortzailea programatzeko beharrezkoak diren prozesuak hartzen ditu barne.

d) **Web Aplikazioa (WA)** lan-paketeak web aplikazioa sortzeko beharrezkoak diren prozesuak hartzen ditu barne.

3. **Frogak (F)** paketeak proiektuaren funtzionamendua egokia dela ziurtatzeko prozesuak hartzen ditu barne.

Kudeaketa

1. **Plangintza (P)** lan-paketeak hasierako plangintza egiteko atazak biltzen ditu.

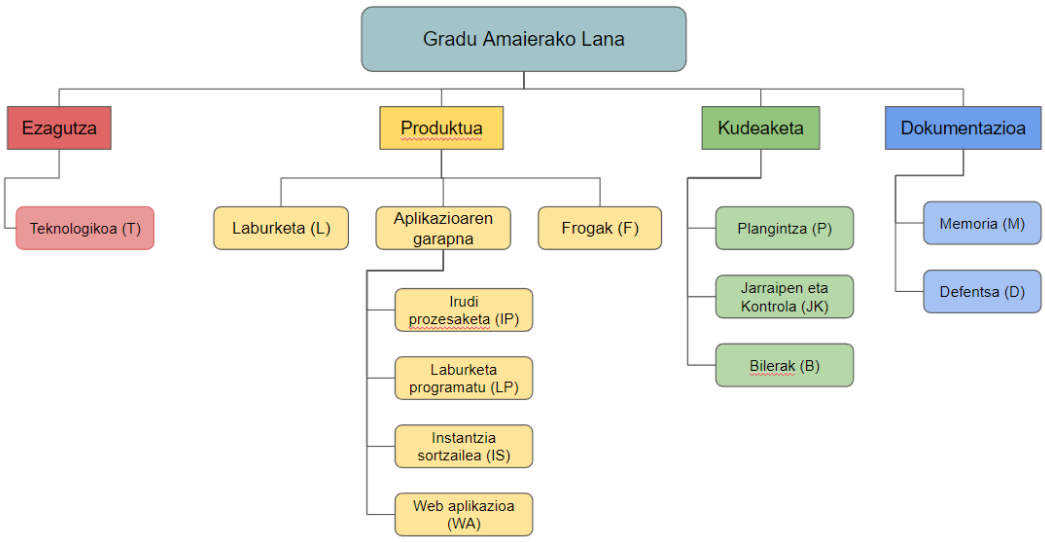
2. **Jarraipen eta Kontrola (JK)** lan-paketeak proiektuaren garapen egokia bermatzeko atazak biltzen ditu.

3. **Bilerak (B)** lan-paketeak proiektua aurrera ateratzeko zuzendariarekin izan beharreko bilerak biltzen ditu.

Dokumentazioa

1. **Memoria (M)** lan-paketeak GrALaren memoria garatzeko beharrezkoak diren atazak biltzen ditu.

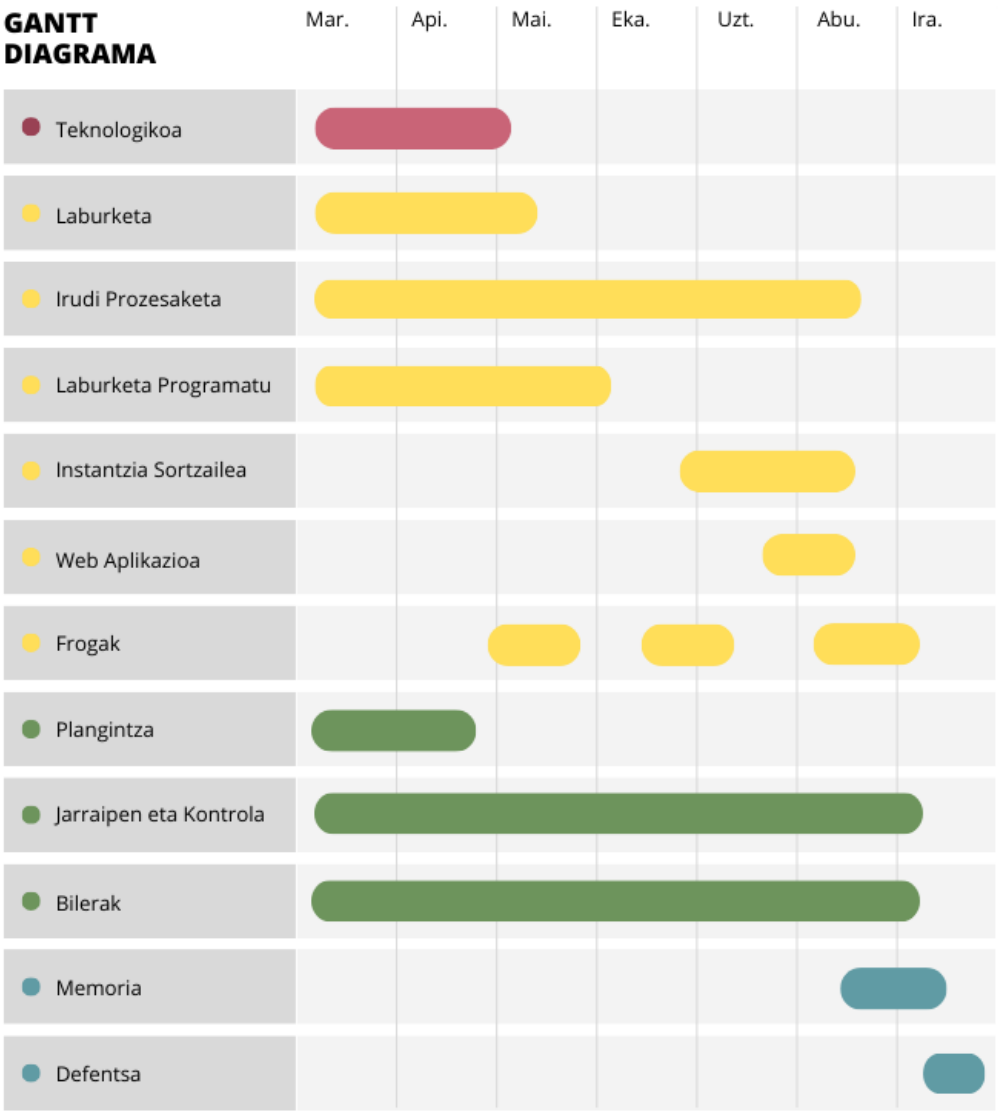
2. **Defentsa (D)** lan-paketeak proiektuaren defentsa burutzeko egin beharreko atazak hartzen ditu barne.



2.1 Irudia: LDE diagrama

2.2 Denbora kudeaketa

Atal honetan proiektua amaitzeko eta entregatzeko mugarriak azaltzen dira. Alde batetik, 2.2 irudian gantt diagrama dugu, non aurretik definitutako lan-paketeen hasiera- eta bukatze-daten estimazioa agertzen den, eta bestetik, proiektuaren garapenaren mugarriak erakusten dituen 2.2 taula dugu.



2.2 Irudia: Gantt diagrama

	2023			
	martxoa	iraila		
	1	7	17	25-29
Proiektuaren hasiera	✓			
Lanaren matrikula		✓		
Defentsaren eskaera		✓		
Lanaren igoera ADDI-ra			✓	
Posterraren bidalketa			✓	
GrALaren defentsa				✓

2.2 Taula: Garapenaren mugarriak

2.3 Informazio-sistemak

Atal honetan azaltzen da nola gordeko diren proiektuan zehar garatutako dokumentuak:

1. **Ordenagailu pertsonala**: Proiektuan aurreratu ahala, sortutako fitxategien segurtasun kopiak ordenagailuan lokalean gordeko dira.

2. **Google Drive**: Lokalean gordetako fitxategien bertsioak Google Drive-en gordeko da. Era honetan segurtasuna bermatzen da ordenagailuan arazoren bat izatekoaren kasuan.

3. **Overleaf**: Memoria garatzeko Overleaf web orria erabiliko da. Tresna honek hodeian lan egiten duenez, sortutako dokumentua segurtasunez gordeko da.

2.4 Komunikazio-sistemak

Proiektuaren garapen egokia bermatzeko zuzendariarekiko komunikazio txukuna eta jarraitua izatea ezinbestekoa da. Hau lortzeko hurrengoa bete da:

1. **Bilerak**: Proiektuaren jarraipen egokia egiteko bi astez behin proiektuko zuzendariarekin bilerak egin dira, bai bere bulegoan bai telematikoki, bileran landuko denaren arabera.

2. **Posta elektronikoa**: Zalantzak argitzeko, intereseko materiala partekatzeko eta bilerak adosteko erabili da.

2.5 Arriskuak

A1. Laburketa egin ez ahal izatea. Ez dakigu posible izango den “Dot Knot”-etik “SAT”-erako laburketa egitea, ezta laburketa egitea posible bada, hau eraginkorra izango denik. Honek arrisku handia suposa dezake proiektu osoa laburketa honen ingurukoa baita. Hala ere, arrisku hau gertatzearen aukera ez da oso altua, “Dot Knot” problema eta antzeko problemak ezagunak baitira eta antzeko problemen laburketak posible direla dakigu aurretik.

A2. Irudietatik informazio erauzketa. "Dot Knot" jokoko irudietatik informazioa lortzeko ezintasunak proiektuaren irismena eta helburuak arriskuan jar ditzake, prozesu hau proiektuaren funtsezko zatia baita. Arrisku hau gertatu izanez gero, "Dot Knot" problemaren informazio erauzketaren planteamendua aldatu beharko litzateke, proiektuan ahalik eta eragin txikiena izatea saiatuz. Honek atzerapenak, irismenean aldaketak edo helburu batzuen ezintasuna suposa dezake. Hala ere, gaur egun irudi prozesaketa inguruko teknologiak oso aurreratuak dira, arrisku hau gertatzearen probabilitatea jaitsiz.

A3. Denbora estimazio eskasa. Proiektu honetan teknologia asko helburu ezberdinekin erabiltzen dira. Gainera, helburu hauek haien arteko menpekotasuna dute, beraz atal guztien funtzionamendu zuzena ezinbestekoa da proiektua aurrera ateratzeko. Hortaz, jakinik horrelako garapen proiektuetan ezustekoak gertatzen direla, garapeneko ataza bakoitzari eskaini beharreko denboraren estimazioa ondo kalkulatu behar da, motz ez geratzeko. Ezustekoak eta denbora larria ekiditzeko, Produktua lan-paketeko atazei denbora estimazioa tartearekin esleituko zaie.

2.6 Jarraipen eta Kontrola

Atal honekin amaitzeko, proiektuaren garapenaren eta arazoen jarraipen eta kontrola aztertuko da. 2.3 taula dugu hasieran zehaztutako dedikazio denborak eta errealitatean igarotako denbora alderatzen duena.

Taulan ikus dezakegunez, nahiz eta Produktua lan-paketeko atalen estimazioak eskuzabaltasunez egin, atal batzuetan denbora gehiago eskaini zaio:

1. **Irudi Prozesaketa**: Lan pakete honetan "Dot Knot" jokoaren irudi-espektro zabal baterako irudien prozesamendua orokortzea zen helburu nagusia. Hortaz, aplikazio ezberdineko irudiekin lan egiterako garaian espero ez ziren zailtasunekin izan dira. Izandako zailtasun hauek ondo azaltzen dira 4.2 atalean.

2. **Instantzia Sortzailea**: Instantzia sortzailearen garatzeko prozesua hasieratik definituta ez zegoenez, estimatutako baino denbora gehiago erabili da instantzia sortzailea egiteko prozesu egokiena ikertzen.

Nahiz eta estimatutako ordu baino gehiago dedikatu behar izan, lanari eskaini behar izandako denboraren igoera hau ez da esanguratsua izan proiektua normaltasunez garatzeko orduan, hasieratik proposatutako helburuak guztiak betetzea lortuz.

Atazak	Estimatutako orduak	Ordu errealak	Desbiderapena
Ezagutza	15	15	0
Teknologikoa	15	15	0
Produktua	200	225	+25
Laburketa	10	10	0
Irudi Prozesaketa	65	85	+20
Laburketa Programatu	60	60	0
Instantzia Sortzailea	20	25	+5
Web Aplikazioa	20	20	0
Frogak	15	15	0
Kudeaketa	13	13	0
Plangintza	4	4	0
Jarraipen eta Kontrola	6	6	0
Bilerak	3	3	0
Dokumentazioa	80	80	0
Memoria	70	70	0
Defentsa	10	10	0
Guztira	308	325	25

2.3 Taula: Jarraipen eta Kontrola

KAPITULUA 3

Teknologiak

Atal honetan proiektua garatzeko erabili diren teknologia ezberdinak azaltzen dira. Proiektuan erabilitako teknologia batzuk dagoeneko ezagunak izan dira, hauen erabilpena erraztuz. Bestalde, teknologia berriak ezagutu dira proiektuan zehar, hauen erabilpena erronka bat bilakatuz.

3.1 Proiektuan erabilitako teknologiak

3.1.1 Python

Python oso ezaguna den programazio-lengoaia interpretatua da, objektuetara bideratua, maila handikoa eta semantika dinamikoa duena [1]. Gradu Amaierako Lan honen osotasuna Python programazio lengoaian burutzea erabaki da, irudiak prozesatzeko eta SAT arazoen inguruko pakete ugariek ematen duten erraztasunarengatik. Hurrengo ataletan lan honetan erabili diren pakete nagusiak azaltzen dira.

3.1.1.1 Flask

Web-aplikazio bat garatu nahi izan denez, Flask paketea erabiltzea erabaki da honek ematen dituen erraztasunengatik. Flask-ek web-aplikazioen sorrera sinplifikatzen du, HTTP eskaerak, URL ibilbideak eta aplikazio-ikuspegiak maneiatzeko funtsezko tresna-multzoa eta antolamendu-egitura eskaintzen baititu [2].

Teknologia hontaz baliatuz, erabiltzailearen eta aplikazioaren bitarteko elkarrekintza samurtu du. Hortaz, web-aplikazioa sortu da non alde batetik irudiak pasata jokuaren problemak ebatz daitezkeen, eta bestetik, jokoaren instantziak sortu. Bi prozesu horiek erraz gauza daitezke flask-en erraztasunari esker.

3.1.1.2 OpenCV

OpenCV (Open Source Computer Vision Library) ordenagailu bidezko ikusmenerako eta ikaskuntza automatikorako kode irekiko software-liburutegia da [3]. Irudiekin lan egiteko funtzio sorta zabala eskaintzen du, hala nola ingeradak aurkitzeko eta ezaugarriak ateratzeko. Irudien prozesaketa garrantzi handiko arloa da proiektu honetan, problema

ebazteko lehenik irudi batetik instantzia sortu behar baita. Hurrengo atal guztientzat izan da beharrezkoa pakete hau:

1. **Irudien Iragazketa**: OpenCV-k iragazteko teknika ugari eskaintzen ditu, zarata murrizteko erabil daitezkeenak. Gainera, OpenCV-k irudien ezaugarri bereizgarriak lortzeko aukera ematen du, hala nola ingeradak. Horrela, taularen dimentsioak lortu dira, gainerako irudia alde batera utzita (hala nola mailaren zenbakia edo laguntza-menua).

2. **Objektuen Detekzioa**: OpenCV-k aukera ematen du irudietan objektu espezifikoak detektatzeko, eta hori erabili dugu jolasaren funtsezko elementuak identifikatzeko eta bereizteko, hala nola kolore-puntuak, laukiak eta zubi-laukiak.

3. **Irudien Segmentazioa**: Liburutegiak irudiak elementu indibidualetan segmentatzeko tresnak eskaintzen ditu, laukiak aztertzeko erabili ditugunak.

4. **Marrazketa eta Irudien Manipulazioa**: Liburutegiak marrazteko gaitasunak eskaintzen ditu, eta horrek aukera ematen du soluzio-ibilbideak marrazteko eta jokoaren emaitzen bistaratze argiak sortzeko. Baita ere "Dot Knot"problemaren instantzia berriak sortzeko.

3.1.1.3 PySAT

PySAT Gradu Amaierako Lan honetan murrizketa boolearrak asetzeko problemak ebazteko erabiltzen diren Python-eko bi liburutegietako bat da. PySATen helburua SAT problemak ebazteko interfaze inkrementala sinple eta bateratu bat eskaintzea da [4].

Pakete hau erabiltzeko erraza da: literal bakoitza zenbaki oso batekin irudikatzen da (zero ezik), eta klausula bakoitza zenbaki naturalen zerrenda batekin. Zenbakia negatiboa bada, literala ezeztatuta dagoela esan nahi du. Azkenik, klausulen zerrenda CNF formatura pasatzeko metodoa erabili ondoren, paketeko ebazle bat erabiliz problemaren soluzioa itzultzen du, hau exisitzen bada. Gainera, klausulak sortzeaz gain, posible da hasierako balioak premisa bezala pasatzea.

3.1.1.4 OR-Tools: CP-SAT

OR-Tools Google-k garatutako liburutegien eta optimizazio-tresnen multzo bat da. Optimizazio arazo ugariri aurre egiteko diseinatuta dago: programazio lineala, ibilbideak eta sare-fluxua, besteak beste [5].

Proiektu honetan erabilitako paketea CP-SAT izan da, murrizketa boolearrak asetzeko problemak ebazteko balio duena. PySAT-ekiko desberdintasun nabariena CNF formatua erabiltzen ez duela da. Hau da, hainbat funtzio ditu murrizketak aplikatzeko, baina murrizketa hauek ez dute ezinbestekoan OR klausulak izan behar. Era honetan, Laburketa atalean azaldutako adierazpen batzuk zuzenean idatz daitezke, OR klausuletara transformatu gabe.

3.1.2 HTML eta CSS

Aurretik azaldu bezala, proiektu honen amaierako emaitza web-aplikazio baten bitartez erakutsiko da. Horretarako, HTML eta CSS fitxategiak sortzea beharrezkoa izan da.

HTML (HyperText Markup Language) webguneak sortzeko markatzeko lengoaia estandarra da. Atalak, paragrafoak eta estekak sortzeko ematen du, HTML elementuen bidez, hala nola etiketak eta atributuak [6]. Proiektu honetan web aplikazioaren orrien egitura zehazteko erabili da.

CSS (Cascading Style Sheets) HTML markatzeko lengoaian idatzitako dokumentu baten aurkezpena deskribatzeko erabiltzen den estilo-orriko lengoaia bat da [7]. Hortaz, proiektu honetan web aplikazioaren orrien egitura zehazteko erabili da.

3.2 Garapenerako teknologiak

3.2.1 Spyder

Spyder kode irekiko Python-entzako ingurune zientifiko bat da. Garapenerako tresna erabilgarriak eskaintzen ditu: fitxategien edizioa, programak arazteko aukera, eta datu esploraziorako menua, besteak beste [8]. Funtzionalitate hauen erabilgarritasunarengatik eta jada ingurune ezaguna denez, proiektuko python fitxategi guztiak Spyder erabiliz garatzea erabaki da.

3.2.2 Visual Studio Code

Visual Studio Code Microsoft-en iturburu-kode editorea da. Honen ezaugarriak arazketarako euskarria, sintaxia nabarmentzea eta kode-osaketa adimenduna izatea dira, besteak beste [9]. Web orriak garatzeko honek eskaintzen dituen erraztasunengatik eta ingurune ezaguna denez, proiektuko HTML eta CSS fitxategiak sortzeko erabili da.

3.2.3 Google Drive

Google Drive hodeian fitxategiak biltegiratzeko eta partekatzeko plataforma da. Hortaz, aurretik ezaguna den ingurunea denez, proiektuaren segurtausn kopiak gordetzeko eta proiektuan zehar idatzitako oharrak gordetzeko erabili da.

KAPITULUA 4

Inplementazioa

4.1 Arkitektura

Proiektuaren arkitektura orokorra arkitektura monolitiko bat da, non kontrol-fluxua app.py fitxategiak zuzentzen duen, HTTP eskaerak maneiatuz eta beharren arabera beste modulua deituz. Lan honetan betetzen den prozesu bakoitza modulutan banandu da. Hurrengo puntuan azaltzen da fitxategi eta karpeta bakoitzaren betebeharra.

4.1.1 Karpeta Nagusia

Karpeta honetan proiektua aurrera ateratzeko modulu nagusiez gain, bigarren mailako karpetak ere daude, karpeta bakoitza beharrezko fitxategiekin. Sortutako moduluak hurrengoak dira:

1. **`app.py`**: Fitxategi hau web aplikazioaren sarrera-puntua da. Web aplikazio bat sortu eta erabiltzailearen ibilbideak eta eskaerak maneiatzen ditu.

2. **`DotKnot_reduction_SAT.py`**: "Dot Knot" problemaren ebazpenarekin lotutako funtzionaltasuna du, PySAT erabiliz. Zehazki, bi metodo ditu, lehenengoa "Dot Knot" problematik SAT-erako murrizketa egiteko eta bigarrena SAT problemaren soluziotik abiatuta alderantzizko murrizketa egiteko.

3. **`DotKnot_reduction_SAT_ortools.py`**: Aurreko kasuan bezala, "Dot Knot" problemaren ebazpenarekin lotutako funtzionaltasuna du, baina kasu honetan OR-Tools erabiliz. Aurrekoak bezala, bi metodo ditu, "Dot Knot" problematik SAT-erako murrizketa egiteko eta SAT problemaren soluziotik abiatuta alderantzizko murrizketa egiteko.

4. **`generator.py`**: "Dot Knot" problemaren instantzien sorkuntza kudeatzen du. Aurrerago azalduko den `gen/` direktorioetako moduluetaz laguntzen da. Alde batetik, problemaren instantzia sortzen du, eta bestetik, instantziaren irudia sortu eta `images/created/` direktorioan gordetzen du.

5. **`img_processing.py`**: Irudien prozesamenduarekin lotutako funtzioak ematen ditu. Zehazki, bi funtzio ditu: lehenengoak problemaren instantzia sortzeko behar den informazio guztia iruditik erauzten du, eta bigarrenak soluzioa marrazten du jatorrizko irudian.

6. **`main.py`**: "Dot Knot" problemaren ebazpena koordinatzen du, beste moduluak deituz. Bi metodo ditu, bata PySAT erabiltzen duena eta bestea OR-Tools.

7. **`run.py`**: Probak exekutatzeko script-a da, bi ebazleek problemak ebazteko behar duten denbora neurtzeko.

4.1.2 Bigarren Mailako Karpetak

1. **`gen/`**: Problemaren instantziak sorkuntzarekin erlazionatutako python fitxategiak ditu. Fitxategi hauek "Numberlinks" [10] proiektuko instantzia sortzailearen fitxategiak dira, `generator.py` fitxategiak erabiltzen dituenak instantziaren irudia sortzeko.

2. **`images/`**: Hiru karpeta ditu: `already solved/` karpeta, non mugikorreko jokoetan zuzenean ebatzitako argazkiak gordetzen diren; `problem`, ebatzi gabeko irudiak gordetzen dituenak; eta `result`, aplikazioak ebatzitako irudiak dituenak.Azkeneko bi direktorioek 7 karpeta dituzte, hurrengo erlazioarekin:

   a) **`big samples/`**: Instantzia sortzailearekin sortu diren instantzia handien irudiak gordetzen ditu.

   b) **`Connect Dots 1/`**: Connect Dots [11] izeneko lehenengo aplikazioko problemen irudiak gordetzen ditu.

   c) **`Connect Dots 2/`**: Connect Dots [12] izeneko bigarren aplikazioko problemen irudiak gordetzen ditu.

   d) **`created/`**: Web aplikazioa exekutatzerakoan sortzen diren instantzien irudiak gordetzen ditu.

   e) **`Dot Knot/`**: Dot Knot [13] izeneko aplikazioko problemen irudiak gordetzen ditu.

   f) **`Flow Free/`**: Flow Free [14, 15] izeneko aplikazioko problemen irudiak gordetzen ditu.

   g) **`uploads/`**: Web aplikazioa exekutatzerakoan ebazteko igo diren instantzien irudiak gordetzen ditu.

3. **`static/`**: Karpeta honek fitxategi estatikoak ditu. `css/` karpeta du, non web aplikazioan erabiltzen den CSS fitxategia gordetzen duen.

4. **`templates/`**: Web aplikazioan erabiltzen diren HTML fitxategian gordetzen ditu.

4.2 Garapena

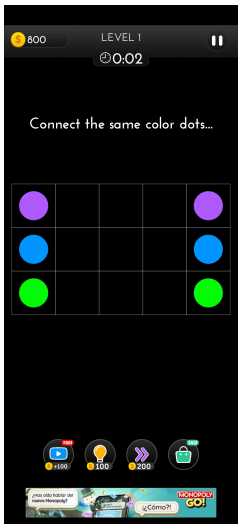
Proiektuaren garapenak fase ezberdinak izan ditu. Alde batetik, instantzia ebazlea garatzeko beharrezkoak izan diren prozesuak datoz. Lehendabizi, irudi batetik lauki-sarea eta bestelako informazio esanguratsua erauzteko prozesua azalduko da. Ondoren, laburketa egiteko sortu diren bi bertsio ezberdinak azalduko dira.

Beste aldetik, instantzia sortzailea egiteko beharrezkoa izan diren metodoak azalduko dira. Azkenik, web aplikazioaren garapena azalduko da, web orrian sortutako funtzionalitateak aztertuz.

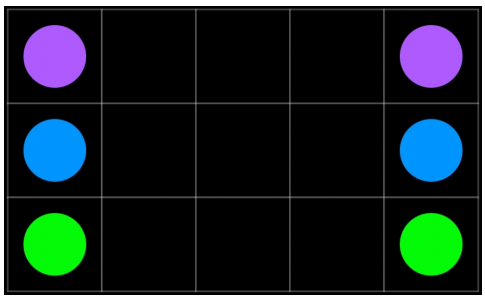
4.2.1 Irudi Prozesaketa

"Dot Knot" problema irudi batetik ebatzi ahal izateko lehenengo pausoa irudiak eskaintzen duen informazio esanguratsua lortzea da. Horretarako, prozesu hau hiru zatitan banatu da: lehenengo helburua sare-laukia identifikatzea da, hurrengoa sare-laukiaren laukiak, eta azkenik, lauki bakoitzaren barnean dagoen informazio esanguratsua (kolore puntuak, zenbat kolore puntu batean eta zubi-laukiak).

Irudi prozesaketa osoan zehar ezinbestekoa izan den funtzioa OpenCV moduluko `findContours` funtzioa da. Funtzio honek irudi bitarra parametro bezala jaso behar du besteak beste, irudian agertzen diren ingerada itzultzen ditu. Beraz, hasieran jasotako irudia beharrezko trasnformazio batetik pasa ondoren `findContours` funtzioari parametro bezala ematen zaio, eta honek itzulitako ingeradaetatik handiena sare-laukiaren ingerada da.



(a) Hasierako irudia

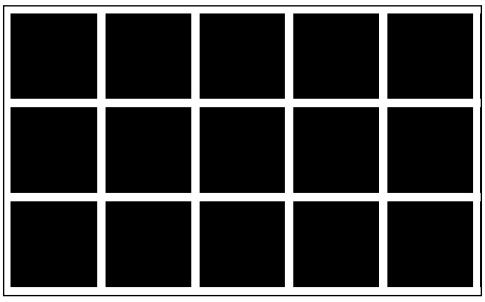


(b) Sare-laukiaren eremua

Hasiera batean garatutako prozesuak funtzio honen inguruan lan egitea nahikoa zen, aukeratutako irudiak erraz prozesatzen baziren. `findContours` funtzioaren bidez laukiak eta puntuak identifikatzea erraza zen, eta hauen ingeradatan oinarrituz problemaren instantzia sor zitekeen. Hala ere, beste aplikazioetako irudiak prozesatzen hasterakoan, kolore eta forma aldaketak direla eta, laukien ingeradak ez ziren ondo errekonozitzen kasu batzuetan. Gainera, zubi-laukiak zituzten irudiak erabiltzerakoan laukien identifikazioa zailtzen zen. Honek eragin zuen irudiaren hasierako eraldaketa sakonagoa egitea eta laukien identifikazio prozesua erabat aldatzea. Beraz, lehendabizi sare-laukia identifikatu behar zela eta gero

irudi segmentu honen gainean beste eraldaketak eginez laukien identifikazioa egiten behar zela ondorioztatu zen.

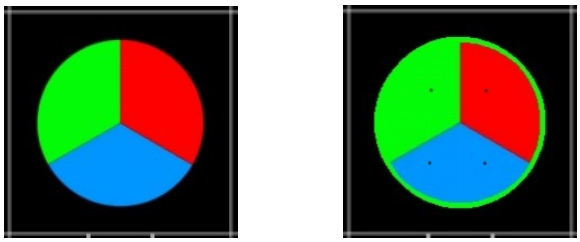
Behin irudiaren eremua murriztu dugula, hurrengo pausoa sare-laukiaren laukiak lortzea da. Horretarako, sare-laukiaren irudia dilatatu egiten dugu, ondoren `HoughLinesP` funtzioa erabiliz sare-laukiaren lerro bakoitza errekonozitzen da ondoren lerro hauekin irudi berri batean margotzeko, 4.2 irudian ikusten den bezala. Honen helburua sare-laukiko barruko zarata kentzea da, lauki bakoitzaren identifikazioa zuzen egiteko.



4.2 Irudia: Sare-laukia hutsik

Behin laukien ingeradak lortuta, lauki bakoitza aztertzen da, barnean duen informazio esanguratsuarekin problemaren instantzia sortuz. Orain bai lauki bakoitzaren barnean `findContours` funtzioa aplikatzea nahiko da puntuen ingeradak lortzeko.

Hala ere, 1.3.2 atalean agertzen diren murrizketak identifikatzeko, ez da nahikoa `findContours` funtzioarekin. Kolorezko puntuen kasuan, aurretik ingeradaren erdiko puntuaren kolorea aztertuz lortzen zen honen kolorea, baina orain hainbat kolore izan ditzakeenez, gehiago aztertu behar da puntua. Kolorea jakiteko, 4.3b irudian beltzez ikusten diren lau pixel-ak aztertu behar dira.

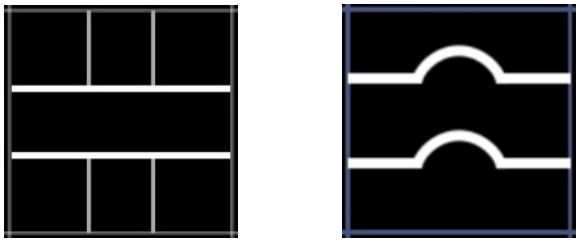


(a) Hiru kolore-puntua **(b)** Pixel-ak aztertzen

4.3 Irudia: Hiru koloreko puntuaren prozesaketa

Ondoren, pixelen kolorea konparatzeko garatutako funtzioa erabiliz, puntuak dituen koloreak ezagutu ditzakegu. Gerta daiteke irudiaren kalitate baxuarengatik, begi-bistan kolore bakarra dena, pixelen kolore balio ezberdinak izatea. Horretarako garatutako `compare_colors` funtzioa erabiltzen da, koloreak konparatzerako garaian pixelaren balioaren desberdintasun txiki bat onartzen duena. Era honetan, ekiditzen da kolore ezberdin bezala tratatzea kolore berdineko lau pixel-ak.

Zubi-laukien kasuan, jokoaren arabera zubiak forma nahiko ezberdina du. Azkenean, lauki batean zubia dagoela jakiteko, nahikoa izan da uneko laukian zirkunferentzia ez den ingerada multzoa dagoela jakitearekin.



(a) Dot Knot zubia **(b)** Flow Free zubia

4.4 Irudia: Aplikazio ezberdinetako zubiak

Behin irudi prozesaketa bukatuta, 4.1a irudiko problematik hurrengo instantzia lortuko litzateke:

```
[ [ [2], [-1], [-1], [-1], [2] ],
  [ [1], [-1], [-1], [-1], [1] ],
  [ [0], [-1], [-1], [-1], [0] ] ]
```

Non kolore bakoitza zenbaki oso ez negatiboarekin adierazten den, eta lauki hutsa -1 zenbakiarekin. Lauki bakoitza lista batekin adierazten da lauki berean hainbat kolore egon daitezkeen kasuetarako. Zubia egotearen kasuan, -2 zenbakiarekin adieraziko litzateke.

4.2.2 Laburketa

Laburketa egiterako garaian, murrizketa nagusiak definitzeko nahi erraz izan zen, 3-koloreztatze problemaren antzekoak baitira. Zailtasun handien ematen zuen murrizketa puntuak elkartzearena izan zen. Hasiera batean uste izan zen nahikoa zela lauki bati alboko laukietako biren kolore bera esleitzea, baina definizio hau ez zen guztiz zuzena eta soluzio okerrak sorrarazten zituen. Azkenean, problemari buruzko informazio eta eztabaidak bilatuz, 1.3.1 atalean agertzen den norabideen murrizketa azaltzen zuen artikulua [16, 17] aurkitu nuen.

Laburketa kodetzeko bai PySAT bai CP-SAT erabili izan dira. Bi pakete hauek erabiliz laburketa egiteko sortutako algoritmoa oso antzekoa da, pakete bakoitzak eskaintzen dituen tresnak baliatuz. Grafoa eta kolore zerrenda izanik, 1.3 atalean azaltzen diren klausulak sortzen dira.

PYSAT-en kasuan, nahikoa izan da OR klausulak lista moduan sortzea. Hau da, OR klausula bakoitzeko literalak lista batean gordetzea. Ondoren, lista hauek guztiak dituen beste lista bat PySAT paketeko `CNF` funtzioari parametro bezala pasata, sortutako klausulak CNF formatuan itzuliko dizkigu funtzioak. Era honetan, SAT ebazleak SAT instantzia ebatzi ahal izango du. Horrez gain, aurretik dakigu hainbat laukien kolorea. Hortaz, `assumptions` lista sortzen da egiazko balioa izan behar duten literalekin, ebazleari parametro bezala pasatzeko. 4.1 algoritmoan ikusten da problema sortzeko eta ebazteko prozesua.

```
cnf = CNF(from_clauses=clauses)
with Mergesat3(bootstrap_with=cnf) as sat:
    a = sat.solve(assumptions=assum)
    if a:
        b = sat.get_model()
```

4.1 algoritmoa: PYSAT-en Mergesat3 ebazlea

CP-SAT-en kasuan, problemaren modeloa sortzen da hasieratik. Honek duen abantaila zuzenean modeloa eraikitzen hasten dela da, hau da, murrizketak paraleloan sortzen dira. PySAT-ekiko desberdintasun nabariena CNF formatuaren ez beharra da. Honen ondorioz, badaude murrizketa batzuk zuzenean sartu daitezkeenak modeloan, CNF formatura pasatzeko beharrezko klausulak sortu gabe. Honen adibidea `OnlyEnforceIf` metodoa da. Metodo hau murrizketa baten gainean egiten da dei. Hortaz, bakarrik sortuko da beste murrizketa metodoari pasatzen zaion adierazpen boolearra egiazkoa bada. Hau da, inplikazioen kasuetarako erabil daiteke. Hemen dugu metodo honen erabileraren adibide bat:

```
model.addBoolOr(x,y).OnlyEnfornceIf(z)
```

Beraz, modeloaren murrizketa guztiak sortu ondoren, ebazleari modeloa pasata honek problema ebatziko du, eta problema ebazgarria bada, literalen balioa lor daiteke.

```
status = solver.Solve(model)
if status == cp_model.OPTIMAL or status == cp_model.FEASIBLE:
    solution = [solver.Value(clauses)]
```

4.2 algoritmoa: CP-SAT ebazlea

Azkenik, behin SAT instantziaren soluzioa dugula, "Dot Knot" problemarako transformazioa egin behar da. Kasu honetan ere bi paketeen algoritmoa oso antzekoa da, egin behar den lan bakarra egiazko literal bakoitzak adierazten duen kolore edo norabidea problema adierazten duten bi grafoetan gordetzea baita. 4.1a irudiko problemaren emaitza eta koloreen norabideak adierazten dituzten grafoak hurrengoak dira:

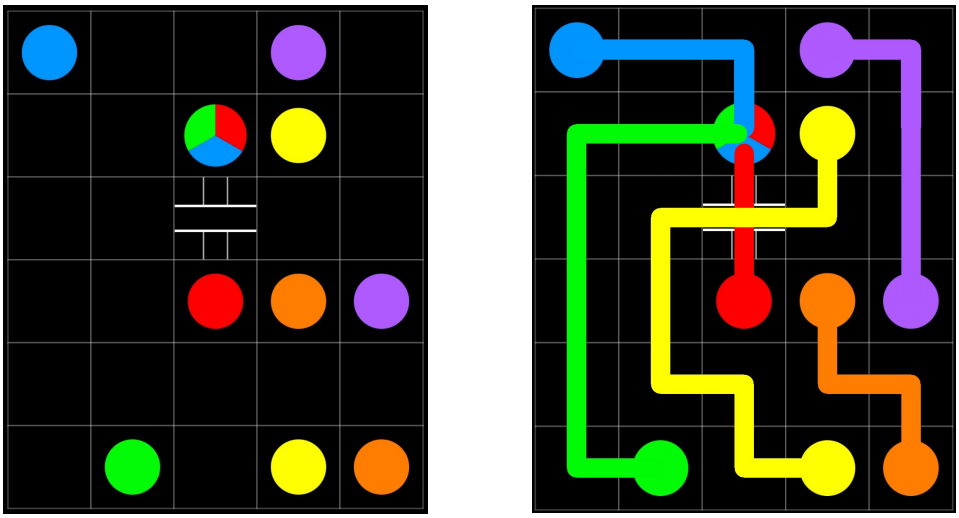
- Emaitza:

```
[ [ [2], [2], [2], [2], [2] ],
  [ [1], [1], [1], [1], [1] ],
  [ [0], [0], [0], [0], [0] ] ]
```

- Norabideak:

```
[ [ [-1], [3], [3], [3], [-1] ],
  [ [-1], [3], [3], [3], [-1] ],
  [ [-1], [3], [3], [3], [-1] ] ]
```

Norabideen grafoan 1.3.1 atalean azaldutako norabideak orden berean agertzen dira, 0tik 5rako adierazpenarekin. Ikus daitekenez, puntua duten laukiek -1 zenbakia dute norabidea adierazteko, kolore bereko bi konexio ez baitituzte. Grafo hau beharrezkoa da problemaren soluzioaren irudia sortzerako garaian alboko laukien kolorea aztertu gabe kolore konexioak margotzeko. 4.5 irudian ikus dezakegu "Dot Knot" jokoko irudi bat prozesatu, laburketa egin, problema ebatzi eta soluzioa margotu ondoren lortutako emaitza, sarrerako irudiarekin alderatuta.



(a) Sarrerako irudia **(b)** Soluzioa marraztuta

4.5 Irudia: Joko baten sarrerako irudia eta lortutako emaitza

4.2.3 Instantzia Sortzailea

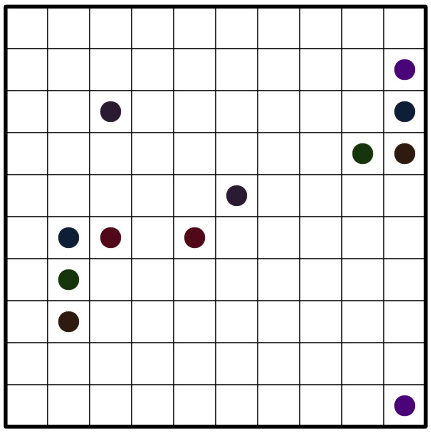
Hasiera batean instantzia sortzailea egiteko ebazlea erabiliz egin zitekeela pentsatu zen. Hots, erabakitako dimentsiotako taula bat sortu eta kolore-puntuak sare-laukian ausaz kokatuz instantzia bat sortzen zen. Ondoren, ebazleak problema hau ebazten bazuen, ebatz zitekeen instantzia lortuko genuke. Beste aldetik, instantzia ebaztezina bazen, kolore-puntuak kokapen berrietan jarriz egingo genuke proba berriro, ebazgarria den instantzia batekin topo egin arte. Ideia honekin, laburketa behin bakarrik egitearekin nahikoa zen, ebazteko kasu bakoitzean hasierako aldagaien erazagupena (`assumptions`) aldaketa bakarra izanik. Hala ere, "Dot Knot" problema ebazteko prozesuetatik, exekuzio denbora handiena hartzen duena SAT ebazlearen prozesua da. Hortaz, instantzia ertain eta handiekin frogak egiterakoan, ebazteko denbora konputazionalki garestia ateratzen da. Horren ondorioz, puntuak ausaz jartzearen ideia baztertzea erabaki zen.

Behin aurreko ideia baztertuta, Numberlinks [10] proiektuko instantzia sortzailea erabiltzea erabaki zen. Numberlinks proiektuko `gen.py` funtzioa erabiltzen da instantzia sortzeko, ondoren instantziaren irudia marrazteko.

Instantziaren irudia sortzeko lehenik `gen.py` funtzioari dei egiten zaio dimentsioak parametro bezala pasata. Honek testu bat itzultzen du instantzia errepresentatzen duena. Hona hemen 10×10 dimentsiotako instantziaren adibidea:

```
10 10
..........
.........6
..4......1
........23
.....4....
.15.5.....
.2........
.3........
..........
.........6
```

Beraz, testu honetatik grafo moduko instantzia bat sortzen da, eta ondoren OpenCV modulua erabilita instantziaren lauki-sarea eta puntuak margotzen dira, irudi bat sortuz (ikusi 4.6 irudia). Irudi hau `images/problem/created/` direktorioan gordetzen da.



4.6 Irudia: Sortutako instantziaren irudia

4.2.4 Web Aplikazioa

Proiektuaren beste helburuetako bat problemak ebatzi eta instantziak sortzen dituen web aplikazioa sortzea da. Aplikazio hau sortzeko HTML eta CSS erabiltzeaz gain, Flask erabiltzea erabaki da.

Proiektu honetan eraginkortasuna helburu denez, mugikorreko aplikazioetako problemak ebazteko CP-SAT erabiltzen duen ebazlea erabiliko da, hau eraginkorrena baita 5 atalean ikusiko den moduan. Pasatako argazki kopuruaren arabera, paraleloan edo seriean ebatziko dira problemak, ahalik eta azkarren ebazteko errekurtsoak xahutu gabe. Beste funtzionalitatearen kasuan, instantziak sortu eta ebaztekoarena, CP-SAT erabilita ebatziko da ere.

Web aplikazioaren egitura simplea da: hasierako orrian bi funtzionaltasunen artean zein bete nahi den aukeratzen da. Eskainitako lehenengo aukera problema ebaztekoarena da, eta bigarrena, dimentsioetatik abiatuta problema sortzearena.

4.7 Irudia: Hasierako orria

Lehenengo aukera sakatzen bada, fitxategiak igotzeko orrialde batera eramango gaitu (ikusi 4.8 irudia). Hemen, irudi bat edo hainbat irudi igo ditzakegu, ebatzi nahi ditugun problemaren kopuruaren arabera. Fitxategiak `JPG` edo `PNG` izan behar dira problema ebatzi ahal izateko. Argazkiak igo ondoren, Flask-en bidez CP-SAT-eko ebazlea erabiliz egindako ebazteko prozesuari egingo zaio dei, ebatzi nahi den irudiaren izena pasata. Behin problema ebatzita, metodo honek emaitzaren irudiaren izena itzuliko du, web aplikazioan bistaratu ahal izateko (ikusi 4.9 irudia).

```
r_img = main_ortools('uploads'+img_name))
```

Hainbat irudi badira, problemak paraleloan ebatziko dira. Problema hauek kudeatu ahal izateko, igotako irudiak `images/problem/uploads/` direktorioan gordetzen dira, eta ebatzitakoak, berriz, `images/result/uploads/` direktorioan.

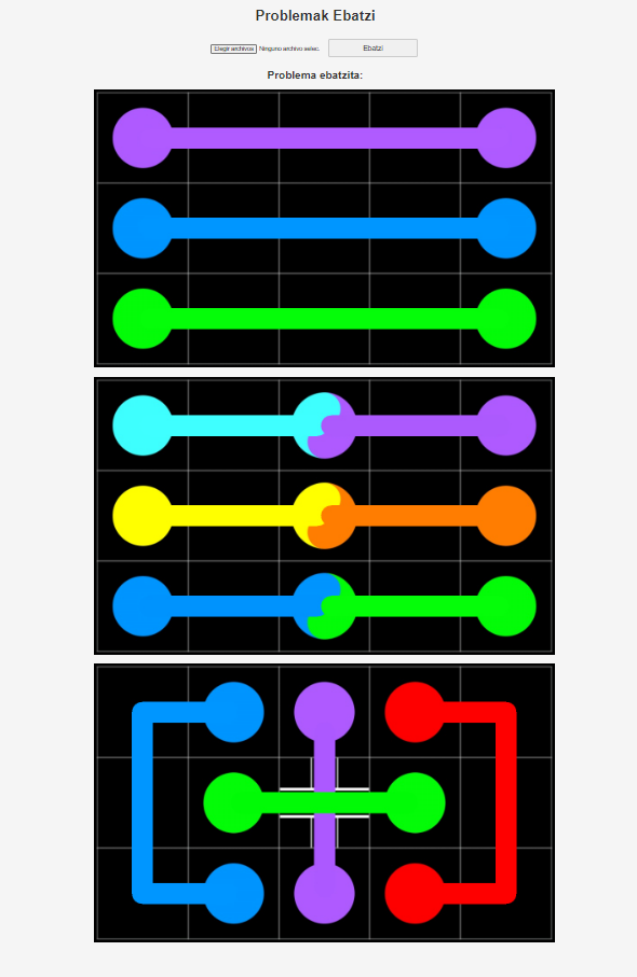
4.8 Irudia: Irudiak igo eta ebazteko orria

Bigarren aukera sakatzen bada, berriz, problema sortzeko aukerarekin egingo dugu topo. Instantzia sortzeko bi datu sartu beharko dira; alde batetik errenkada kopurua, eta bestetik, zutabe kopurua (ikusi 4.10 irudia). Instantzia handiak nahi ditugunez, gutxienez 7×7 dimentsiotako instantzia sortzea eskatzen da. Hots, errenkada edo zutabe txikiagoak adierazten badira, errore mezua agertuko da (ikusi 4.11 irudia). Errore hauek agertzeko `flash` mezuetaz baliatzen da.

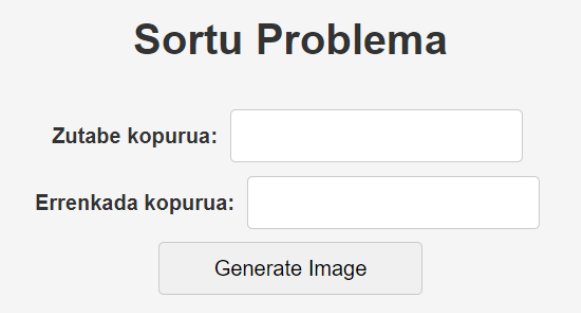
Instantzia sortzeko dimentsio zuzenak sartzen badira, Flask-en bidez `generator` funtzioari egiten zaio dei, zutabe eta errenkada kopurua parametro moduan pasata.

```
generated_image = generator.generator(num_columns, num_rows)
```

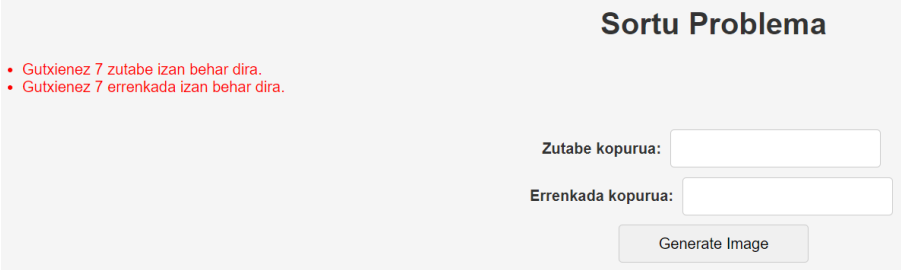
Azkenik, behin instantzia sortuta, instantzia ebatzita ere azalduko da (ikusi 4.12 irudia). Instantzia hauek kudeatzeko, sortutako instantziaren irudia `images/problem/created/` direktorioan gordetzen da, eta ebatzitakoarena, berriz, `images/result/created/` direktorioan.



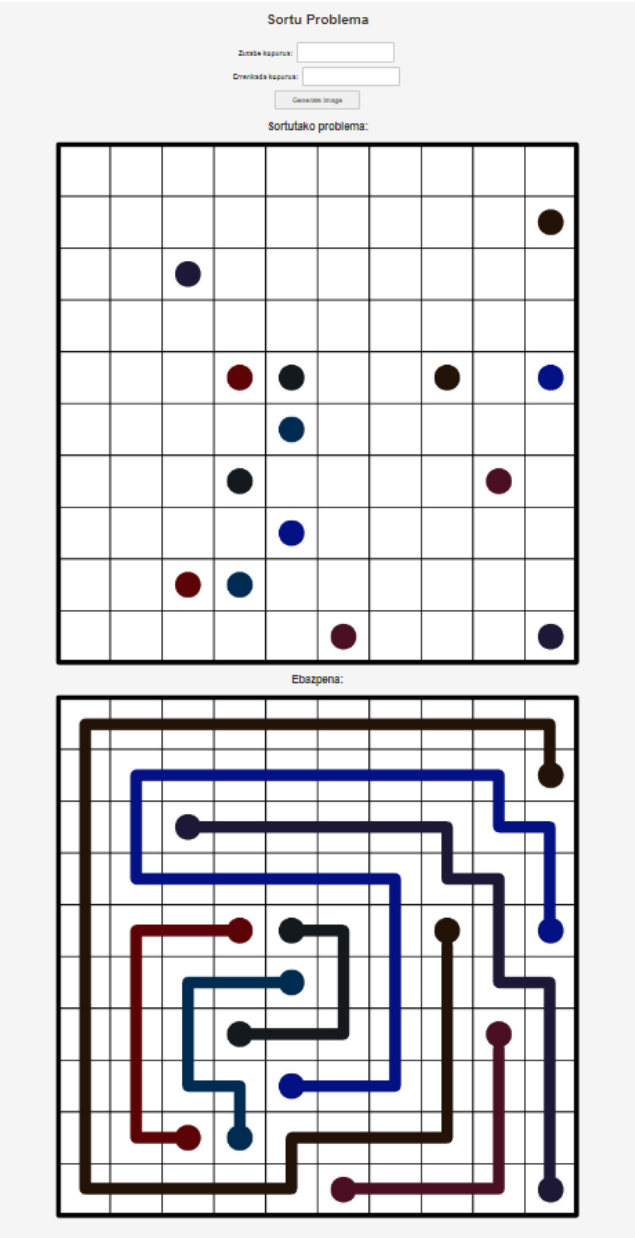
4.9 Irudia: Problemak ebatzita



4.10 Irudia: Instantzia sortzeko eta ebazteko orria



4.11 Irudia: Instantzia sortzen dimentsioen errorea



4.12 Irudia: Instantzia sortuta eta ebatzita

KAPITULUA 5

Froga kasuak

Atal honetan laburketaren bi bertsioen exekuzio denborak azalduko dira. Bi zatitan banatu da atala; lehenengo zatian tamaina txikiko instantzia askoren exekuzio denborak aztertuko dira, eta bigarrenean, berriz, tamaina handiko instantzia gutxiagoren exekuzio denborak. Bi paketeen eraginkortasuna alderatzeaz gain, errendimendua hobetzeko problemen ebazpena paralezatzea erabaki da. Beraz, exekuzio denboren tauletan lau exekuzio denbora adieraziko dira:

1. **PySAT serie**: kasu honetan problemak seriean ebatzi dira PySAT paketea erabilita.
2. **PySAT paralelo**: kasu honetan problemak paraleloan ebatzi dira PySAT paketea erabiliz. Hau da, hainbat problema aldi berean ebatzi izan dira.
3. **CP-SAT serie**: kasu honetan problemak seriean ebatzi dira OR-Tools liburutegiko CP-SAT paketea erabilita.
4. **CP-SAT paralelo**: kasu honetan bai problemak paraleloan ebatzi dira OR-Tools liburutegiko CP-SAT paketea erabilita.

OR-Tools-en kasuan azaldu beharra dago CP-SAT paketeak SAT instantziak sortzeko eta ebazteko jada paralelizazioa erabiltzen duela. Beraz, CP-SAT "seriean" exekutatzen dela esaterakoan, problema multzoari buruz ari gara, eta ez problema bakoitzaren ebazpenari buruz. Froga guztiak egiteko Acer Aspire A515-54G ordenagailu bat erabili izan da. Ordenagailu honen kontuan izan beharreko ezaugarri teknikoak hurrengoak dira:

1. **Prozesadorea (CPU)**: Intel Core i5-10210U (4 nukleo, 8 hari, 1.60-2.11GHz).
2. **RAM Memoria**: 8GB (7.72GB erabilgarri, DDR4 SDRAM teknologia).
3. **Memoria**: SSD 512GB.
4. **Sistema Eragilearen bertsioa**: Windows 10 Home.
5. **Txartel Grafikoa**: NVIDIA GeForce MX250.

5.1 Instantzia Txikiak

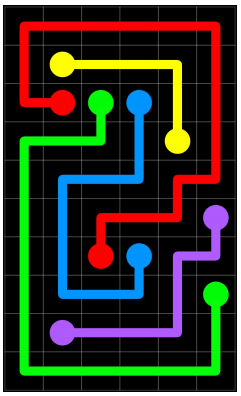
Instantzia txikiak bezala hartzen ditugu 15×15 dimentsioa baino txikiagoa duten instantziak. Hau da, mugikorreko aplikazioetan agertzen diren problema guztiak instantzia txikiak dira. Beraz, instantzia txikien data set-a betetzeko, lau aplikazio ezberdineko problemen argazkiak atera dira, 469 murrizketa ezberdinetako problemak lortuz. Hots, "zubia" (1.3.2.2) eta "hainbat koloretako puntua" (1.3.2.1) murrizketak dituzten irudiak badira ere. Problema hauen dimentsioak 5×5-tik 10×10-erakoak dira, eta problemaren kolore kopurua tamainarekiko proportzionala da. Ebatzitako bi problemen adibideak ikus ditzakegu 5.1a eta 5.1b irudietan. Erabilitako aplikazio bakoitzetik lortutako problema guztien exekuzio denborak 5.1 taulan ikus ditzakegu.

		PySAT		CP-SAT	
Aplikazioa	Argazki kopurua	serie	paralelo	serie	paralelo
Dot Knot	100	1m34s	___	2m38s	___
Flow Free	150	3m45s	___	3m26s	___
Connect Dots 1	119	2m47s	___	2m23s	___
Connect Dots 2	100	1m57s	___	2m30s	___
Guztira	469	10m3s	7m9s	10m28s	3m47s

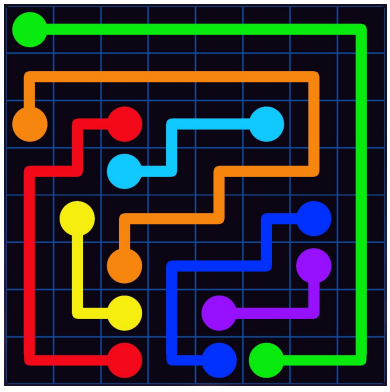
5.1 Taula: Instantzia txikien exekuzio denborak

Alde batetik, serie exekuzioak alderatzen baditugu ikusiko dugu PySAT CP-SAT baino pixkat azkarragoa izan dela. Tamaina txikiko instantzia asko izanik, eraginkorragoa da zuzenean problemak seriean ebaztea hauen ebazpena paralelizatu gabe, PySAT-ek egiten duen moduan. CP-SAT-en kasuan, instantzia txiki bakoitzaren modeloaren sorkuntza eta ebazpena paralelizatzeak ez du errendimenduan eragin positiborik.

Beste aldetik, bai PySAT-en bai CP-SAT-en paralelo bertsioa serie exekuzioa baino azkarragoa izan da. Taulan ikusten den moduan, errendimendu altuen ematen digun aukerak CP-SAT-en bidezko exekuzio paraleloa da. Kasu honetan, instantzia txikiak izan arren, prozesuen kudeaketa eraginkorra denez, paralelizazioaren kostu gehigarria izatea ez du eraginik exekuzioaren eraginkortasun positiboan.



(a) Dot Knot ebatzita



(b) Connect Dots 2 ebatzita

5.1 Irudia: Ebatzitako irudien adibideak

5.2 Instantzia Handiak

Instantzia handiak bezala hartzen ditugu 15×15 dimentsioa eta gorako dimentsioak dituzten instantziak. Hau da, mugikorreko aplikazioetan agertzen diren problemak motz geratzen dira. Horretarako, sortutako instantzia sortzaileaz baliatuz, 15×15-tik 40×40-erako instantziak sortu ditugu. Kasu honetan, murrizketa berezirik gabeko instantziak izango dira bakarrik.

Instantzia hauentzat bi taula sortu dira: lehenengoa 5.2 taula, 15×15 dimentsiotik 26×26 dimentsiorako instantziak biltzen dituena, eta 5.3 taula, 27×27 dimentsiotik 40×40 dimentsiorako instantziak biltzen dituena. Banaketa honen bi arrazoi daude:

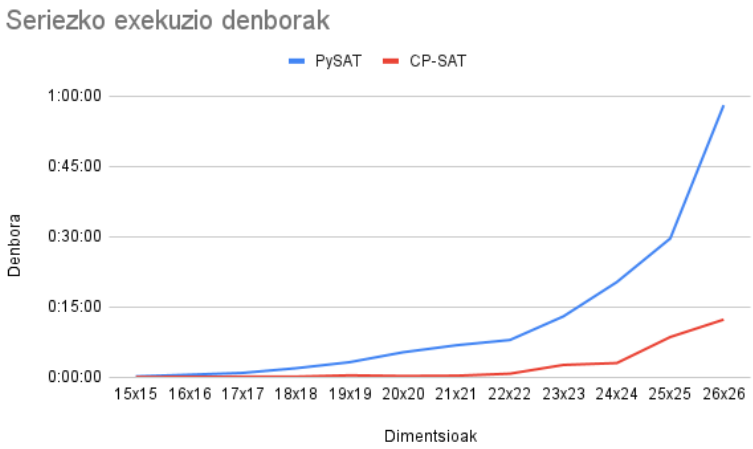
1. **Exekuzio denbora irregularrak**: Alde batetik, 5.3 taulan ikusten denez, instantzia hauen exekuzio denbora oso irregularra da. Tamaina bereko instantzia ezberdinen artean desberdintasun nabaria dago. Horrez gain, tamaina desberdineko instantzien artean ez da proportzionaltasun bat nabarmentzen, 5.2 taulako kasuetan ez bezala.

2. **Paralelizazioa**: Exekuzio denborak irregularrak direla eta, bakarrik 15×15 dimentsiotik 26×26 dimentsiorako instantziak paralelizatzea erabaki da. Honen helburua exekuzio hauetatik ondorio errealagoak eta garbiagoak ateratzea da.

		PySAT		CP-SAT	
Dimentsioak	Instantzia sortzen	serie	paralelo	serie	paralelo
15×15	1s	13.87s	____	5.27s	____
16×16	3s	35.64s	____	7.95s	____
17×17	2s	58.55s	____	9.16s	____
18×18	4.6s	1m58s	____	10.75s	____
19×19	7.1s	3m15s	____	27.53s	____
20×20	13.5s	5m22s	____	18.41s	____
21×21	11.3s	6m52s	____	22.94s	____
22×22	11.15s	7m59s	____	48.12s	____
23×23	22.25s	13m1s	____	2m40s	____
24×24	17s	20m20s	____	3m4s	____
25×25	13.4s	29m38s	____	8m37s	____
26×26	30.9s	58m3s	____	12m20s	____
Guztira		2h28m14s	2h8m24s	29m8s	18m49s

5.2 Taula: Instantzia handien exekuzio denboren lehenengo taula

Instantzia txikien kasuan ez bezala, serie exekuzioak alderatzen baditugu ikusten dugu CP-SAT PySAT baino askoz azkarrago bukatu duela exekuzioa. Kasu honetan tamaina handiako instantziak direnez, askoz nabariagoa da modeloaren sorkuntza eta ebazpena paralelizatzeak ematen duen abantaila. 5.2 irudian garbi ikus dezakegu PySAT eta CP-SAT erabiltzearen arteko aldea.



5.2 Irudia: Exekuzio denboren grafikoa

Beste aldetik, bai PySAT-en bai CP-SAT-en paralelo bertsioa serie exekuzioa baino azkarragoa izan da. Hala ere, kasu honetako desberdintasuna ez da instantzia txikiena bezain nabaria izan. Taulan ikusten den moduan, errendimendu altuen ematen digun aukerak CP-SAT-en bidezko exekuzio paraleloa da.

Lehen esan bezala, instantzia handien bigarren tauletako instantzien exekuzio denborak ez dute 5.2 grafikoko joera jarraitzen. Honen arrazoia mugako kasuan datza. Nahiz eta bi instantzia tamaina berekoak izan eta kolore kopuru berdina izan, algoritmo ebazlearen prozesuaren arabera, instantzia bat ebaztea bestea baino zailagoa izan daiteke. Hau da, puntuen kokapenaren arabera, instantzia bat algoritmoarekiko mugako kasua izan daiteke. Hona hemen instantzia handien bigarren taula:

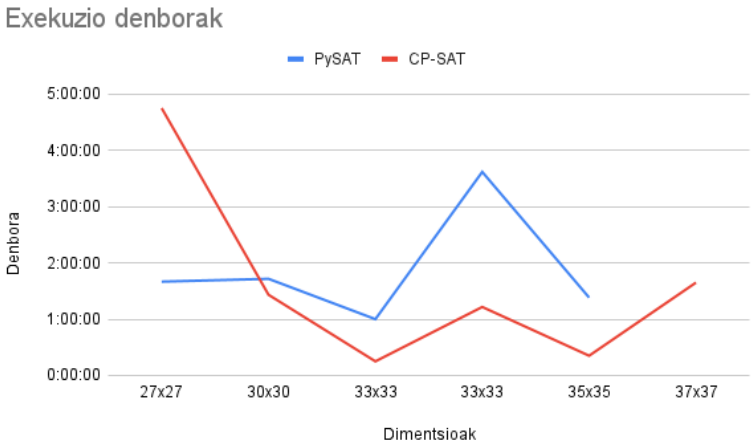
Dimentsioak	Instantzia sortzen	Laburketa egiten	PySAT serie	CP-SAT serie
27×27	31.6s	52s	1h40m	4h45m
30×30	44.1s	1m34s	1h43m	1h26m
33×33		55s	1h	15m
33×33	1m16s	1m44s	3h37m	1h13m
35×35		35s	1h23m	21m
36×36	2m23s	2m	>14h	>5h
37×37		1m10s	_____	1h39m
37×37		1m23s	_____	
40×40	1m22s	1m9s	_____	>14h

5.3 Taula: Instantzia handien exekuzio denboren bigarren taula

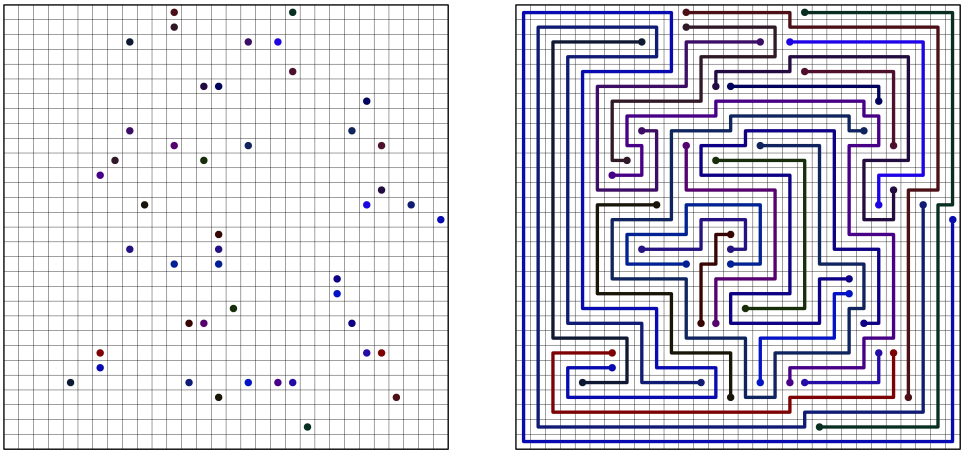
Taulan ikusten den moduan, 27×27 tamainako instantzia azkarrago ebazten da PySAT ebazlearekin CP-SAT-enarekin baino. Izan ere, instantzia hau arretik azaldutako muga kasua da CP-SAT ebazlearentzat.

Beste aldetik, 33×33 tamainako bi instantzien arteko exekuzio denboraren aldea nabarmena da. Aurreko kasuan bezala, lehenengo instantzia mugako kasua izan da, baina kontrako norabidean. Hau da, instantzia hau CP-SAT ebazlearentzat ebazteko erraza da.

Orokorrean ikus daiteke exekuzio denbora tamainaren menpekoa izateaz gain, kolore puntuen kokapenarekiko menpekoa izaten hasi dela. 5.3 grafikoan garbi ikus dezakegu 5.2 irudikoko joera albo batera utzi dela. 5.4 irudian ebatzitako 30×30 problemaren adibideak ikus dezakegu.



5.3 Irudia: Exekuzio denboren grafikoa



(a) 30×30 dimentsioko problema **(b)** 30×30 dimentsioko problema ebatzia

5.4 Irudia: 30×30 dimentsioko problema ebatzi aurretik eta ondoren

KAPITULUA 6

Ondorioak

Amaitzeko, kapitulu honetan proiektutik ateratako ondorioei buruz hitz egingo da, proiektuan erabilitako teknologien eta garatutako ideien eraginkortasuna eta erabilgarritasuna hausnartuz.

6.1 Ondorioak

Atal honetan, proiektu honen inguruko hainbat ondorio azalduko dira.

Proiektuak hasieran bi helburu nagusi zituen: alde batetik "Dot Knot" problema SAT ebazle baten bidez ebazteko laburketa egin, instantzia sortzailearekin batera, eta bestetik, funtzionalitate hauek eskaintzen dituen web aplikazioa garatzea.

Laburketa egiteko helburua espero bezala lortu da. "Dot Knot" problemako instantzia batetik SAT instantzia sortzea lortu da, laburketa zuzena eta eraginkorra izanik. Gainera, 5.3 taulan ikusten den bezala, laburketaren exekuzio denbora ebazlearekin alderatuta oso txikia da.

Beste aldetik, instantzia sortzaile batetik abiatuta irudiak sortzea lortu dugu. Hala ere, hasierako idea ebazlearen bidez instantzia sortzea izan arren, hau burutzea ez da posible izan. Azkenean "Dot Knot" problemen irudiak sortzeko "Numberlinks"proiektuko instantzia sortzaileaz baliatuz egin da.

Azkenik, web aplikazioaren garapenaren inguruan, esperotako bi funtzionalitateak era eraginkorrean eskaintzea lortu da. Interfazea ulertzeko erraza da, bai hainbat irudi ebazteko bai instantziak sortzeko.

Hurrengo ataletan froga kasuetatik ondorioztatutako ideiak partekatuko dira. Alde batetik paralelizazioaren inguruan ateratako ondorioak ditugu, eta bestetik, "Dot Knot" problemaren ingurukoak.

6.1.1 Paralelizazioa

Egindako froga kasuetatik paralelizazioaren arloko hainbat ondorio atera ditzakegu. Lehenengo instantzia txikien exekuzioetatik ateratako ondorioak azalduko dira, eta ondoren instantzia handien exekuzioetatik.

Instantzia txikiak seriean exekutatzen badira, hobe da problemen modelo sorkuntza eta ebazpena paralelizatu gabe egitea. Honen arrazoi nagusiak hurrengoak dira:

1. **Paralelizazioaren kostua**: Programa bat paralelizatzerakoan hariak edo prozesuak sortu eta kudeatu, sinkronizatu eta haien artean komunikatu egiten dira. Honek seriean exekutatzearekiko kostu gehigarria suposatzen du. Instantzia txikientzat, kostu hori esanguratsua izan daiteke paralelizazioaren benetako onurarekin alderatuta, eraginkortasun murriztua ekar dezakeena.

2. **Baliabideen erabilera**: Paralelizazioak sistemaren baliabide gehiagoren erabilpena eskatzen du, hala nola memoria eta PUZ. Instantzia txikiak erabiliz, baliabideen gainkarga neurriz kanpokoa izan daiteke problemaren tamainari dagokionez, eta horrek eragin negatiboa izan dezake makina berean gauzatzen diren beste prozesuetan.

Beste aldetik, instantzia txikiak paraleloan exekutatzea seriean exekutatzea baino eraginkorragoa da. Kasu honetan, problemeak paraleloan ebazteaz gain, problema bakoitzaren ebazpena paraleloan izatea eraginkortasuna nabarmen hobetzen du.

Instantzia handien kasuan, zuzenean hobe da bai problemen arteko ebazpena paralelizatzea bai problemaren modeloaren sorkuntza eta ebazpena paralelizatzea. Hortaz, garbi geratzen da CP-SAT-en ebazlea paralelizatzen badugu oso eraginkorra dela.

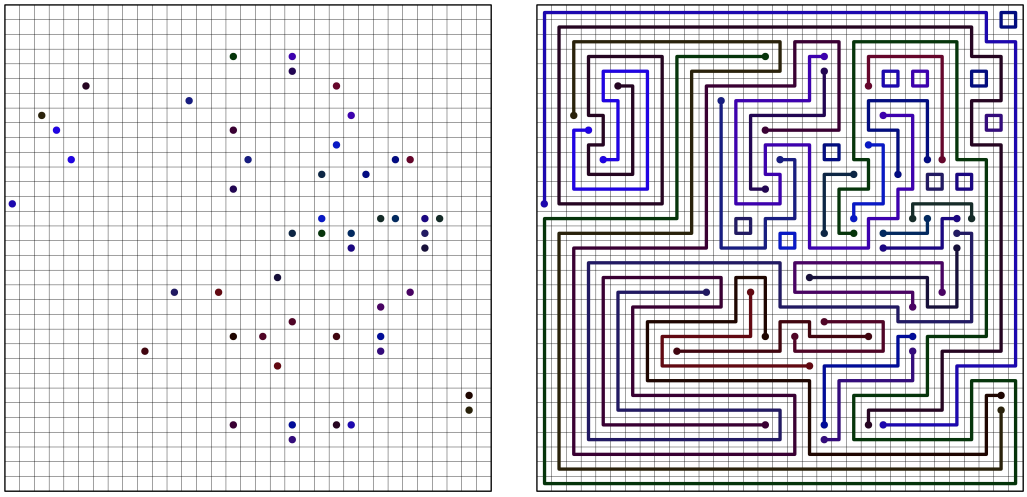
6.1.2 "Dot Knot" problema

"Dot Knot" problema hau ebazteko prozesu eraginkorrena aukeratuz, problema denbora eraginkorrean ebazteko muga 37×37 dimentsiotako problemen inguruan dago. Beraz, gizakiok joku moduan jolasten ditugun problemak ebazteko nahikoa da, joko hauek tamaina txikikoak baitira. Hala ere, bizitza errealeko problemetara egokitu nahi bada, muga hau kontuan izan beharreko alderdi bat da.

Beste aldetik, tamaina txikiko eta ertaineko problemak ebazteko puntuen kokapena ez du eragin handirik, baina 26×26 dimentsioa baino handiagoko problemengan eragina handituz doa. Faktore hau kontuan izan beharreko beste bat da, mundu errealeko problemak ebatzi nahi badira.

6.2 Etorkizunerako Lana

Proiektu honetan egindako laburketak ez du bermatzen emaitzetan zikloak egongo ez direnik (ikusi 6.1 irudia). Normalean, "Dot Knot" eta antzeko jokoetako problemak ziklo gabeko ebazpenak dituzte, beraz CP-SAT erabiliz emaitza egokia aukeratu daiteke, zikloak baztertuz. Hala ere, laburketan zikloak ekiditzeko murrizketa sortzea posible den ikertzea interesgarria izango litzateke.



6.1 Irudia: Puntu guztiak konektatuz zikloak onartzen dituen problema

Beste aldetik, proiektu honetan garatutako ebazlea bizitza errealeko arazoetara egokitzea interesgarria izango litzateke. "Dot Knot" problemaren premisa puntuak konektatzearena da, beste konexioekin talka egin gabe. Gaur egungo munduan, premisa antzeko arazo ugari ditugu; hala nola, hodi sistemen diseinua, hornidura-kateen plangintza, eta eraikinetako kableen plangintza, besteak beste.

Bibliografia

[1] Python. https://www.python.org/doc/essays/blurb/, 2022. Ikusi 17 orrialdea.

[2] Flask. https://pythonbasics.org/what-is-flask-python/, 2012. Ikusi 17 orrialdea.

[3] OpenCV. https://opencv.org/about/, 2012. Ikusi 17 orrialdea.

[4] PySAT. https://pysathq.github.io/, 2012. Ikusi 18 orrialdea.

[5] OR-Tools. https://developers.google.com/optimization?hl=es-419, 2012. Ikusi 18 orrialdea.

[6] HTML (HyperText Markup Language). https://www.hostinger.com/tutorials/what-is-html, 2012. Ikusi 19 orrialdea.

[7] CSS. https://en.wikipedia.org/wiki/CSS, 2012. Ikusi 19 orrialdea.

[8] Spyder. https://www.spyder-ide.org/, 2012. Ikusi 19 orrialdea.

[9] Visual Studio Code. https://en.wikipedia.org/wiki/Visual_Studio_Code, 2012. Ikusi 19 orrialdea.

[10] Thomas Dybdahl Ahle. Numberlinks. https://github.com/thomasahle/numberlink, 2012. Ikusi 22, 27 orrialdeak.

[11] Connect the Dots - Match. play.google.com/store/apps/details?id=com.playvalve.connect.dots, 2023. Ikusi 22 orrialdea.

[12] Connect the Dots. play.google.com/store/apps/details?id=com.bigman.connectdots, 2012. Ikusi 22 orrialdea.

[13] Dot Knot - Line & Color Puzzle. play.google.com/store/apps/details?id=net.ibexsolutions.flow8, 2015. Ikusi 22 orrialdea.

[14] Flow Free. play.google.com/store/apps/details?id=com.bigduckgames.flow, 2012. Ikusi 22 orrialdea.

[15] Flow Free: Bridges. play.google.com/store/apps/details?id=com.bigduckgames.flowbridges, 2012. Ikusi 22 orrialdea.

[16] Flow Free redux: eating SAT-flavored crow. https://stackoverflow.com/questions/23622068/algorithm-for-solving-flow-free-game/23626076#23626076, 2014. Ikusi 25 orrialdea.

[17] Algorithm for solving Flow Free Game. https://mzucker.github.io/2016/09/02/eating-sat-flavored-crow.html, 2016. Ikusi 25 orrialdea.