Háskóli Íslands

Aðgerðargreining

IÐN401G

Lokaverkefni - Próftöflubestun

Höfundar:
Egill Ian Guðmundsson
Hildur Ösp Sigurjónsdóttir
Stefán Carl Peiser
Unnur Sigurjóns

Umsjónarkennari:
Tómas Philip Rúnarsson

4. apríl 2016

Efnisyfirlit

1 Ágrip

Bestun próftöflu hefur verið vandmeðfarið verkefni víða um heim síðastliðna áratugi. Verkefnið snýst um að skipuleggja fjöldan allan af prófum yfir takmarkað tímabil. Þessu tímabili er skipt upp í ákveðið marga prófstokka, oftast tvo stokka á dag og þarf verkefnið auk þess að uppfylla fyrirfram ákveðin skilyrði sem kallast skorður. Skorðurnar eru mismunandi eftir skólum. Þær skorður sem nauðsynlegt er að uppfylla í öllum tilfellum kallast harðar skorður. Sem dæmi um harðar skorður má nefna að sami nemandi getur ekki þreytt tvö próf á sama tíma (í sama prófstokki) og að takmarkaður fjöldi nemenda getur þreytt próf á sama tíma, enda takmarkaður sætafjöldi í boði fyrir hvern stokk. Ef allar harðar skorður eru uppfylltar við bestun próftöflu er litið svo á að lausnin sé lögleg.

Aðrar skorður sem hægt er að setja, en teljast ekki nauðsynlegar til þess að leysa verkefnið, eru kallaðar lausar skorður. Dæmi um lausar skorður eru t.d. ákjósanlegur hvíldartími nemenda á milli prófa eða að fjölmenn námskeið séu með próf framarlega á próftímabili. Gefa má þessum lausu skorðum meira eða minna vægi og þar með stilla líkanið til þess að gefa sérsniðnar lausnir. Hægt er að hafa sem mestan hvíldartíma milli prófa , sem fæsta prófstokka , fjölmenn eða erfið námskeið sem fyrst eða taka tillit til allra skorða eins og gert er í lokin.

Eins og sést frekar í eftirfarandi köflum er hægt að breyta röðuninni á prófunum með ýmsu móti til að fá niðurstöður frábrugðnar röðun Háskóla Íslands. Hægt er að fækka prófstokkum niður í 13, auka meðalhvíldartíma nemenda um hálfan dag eða færa fjölmennari námskeið framar í töflu. Allt þetta má skoða nánar í niðurstöðum skýrslunnar sem sýnir mismunandi útfærslur á röðun próftöflunnar fyrir vormisseri 2016.

2 Inngangur/bakgrunnur

Verkefnið sem leyst var af hendi snerist um að hanna líkan sem bestar próftöflu nemenda við Verkfræði- og náttúruvísindasvið Háskóla Íslands. Hugsanlega má rekja þörf verkefnisins til þess að í dag er einn aðili innan háskólans sem sinnir því verkefni að útbúa próftöflur fyrir alla nemendur skólans, þ.e. prófstjóri. Auðvelda mætti ferlið með því að útbúa líkan sem uppfyllir kröfurnar í stað þess að raða prófunum handvirkt. Að sögn kennara ætti líkanið okkar að virka fyrir háskólann í heild sinni, en ekki verður þó farið ítarlegra í það hér.

Upp voru gefnar þrjár harðar skorður:

1. Enginn nemandi getur setið próf samtímis þ.e.a.s. í sama prófstokki.
2. Próf samkenndra námskeiða skulu vera á sama tíma.
3. Fjöldi úthlutaðra sæta getur ekki verið meiri en þau sem eru til staðar, í þessu tilfelli 450 sæti.

Auk þess gætu nemendur kosið að hafa smá hvíld á milli prófa og erfiðari próf fyrr á tímabilinu til að einbeita sér frekar að þeim. Einnig verður skoðað hvort hægt sé að koma í veg fyrir að einhverjir hópar innan deildarinnar taki mörg próf á stuttum tíma og þá hvernig væri hægt að leysa það. Óskir kennara varðandi lausnina var að fjölmenn námskeið fengju að hafa prófin snemma á próftímabilinu en stjórnendur skólans vildu sjá lausn sem gæti lágmarkað kostnað vegna húsnæðis auk yfirsetu- og aðstoðarfólks yfir prófatímabilið sem þýðir stytting próftímabilsins eins og hægt er.

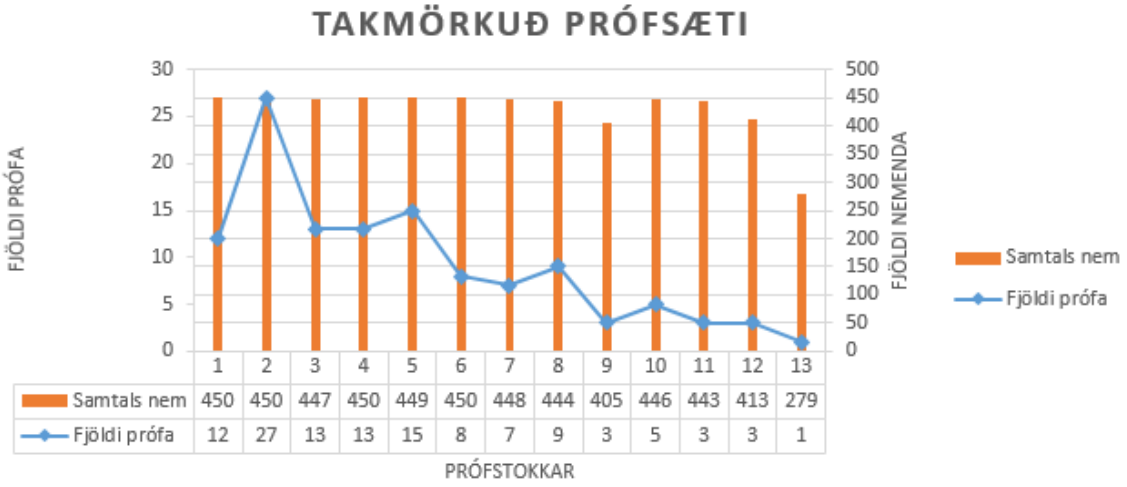
Í þessu tiltekna líkani er notast við gögn um fallprósentu frá prófstjóra til þess að ákvarða erfiðleika prófs þar sem hærra fall þýðir að öllum líkindum erfiðara próf.

Reynt var að hafa sem mesta hvíld milli prófa án þess að lengja próftímabilið um of. Erfiðlega gekk að hafa þriggja stokka hvíldartíma eftir öll próf og áætla má að einhverjir nemendur séu í prófum eitthverja daga í röð eða samdægurs.

Stilla má líkanið til að lágmarka fjölda stokka, hámarka hvíldartíma, hafa fjölmenn námskeið sem fyrst eða taka mið af þessu öllu. Hvíldartíminn er hafður sem nokkrar skorður sem taka mið af því hversu margir þurfa að vera í árekstri á prófum til þess að líkanið hliðri prófum og auka hvíldartíma. Hvernig skal stilla þessar skorður er fjallað um nánar í seinni köflum. Leggja má enn meiri áherslu á hvíldartíma með því að sleppa markfalli en hægt er að leggja áherslu á lágmörkun prófstokka eða að færa fjölmenn námskeið framar með því að nota tilsett markföll.

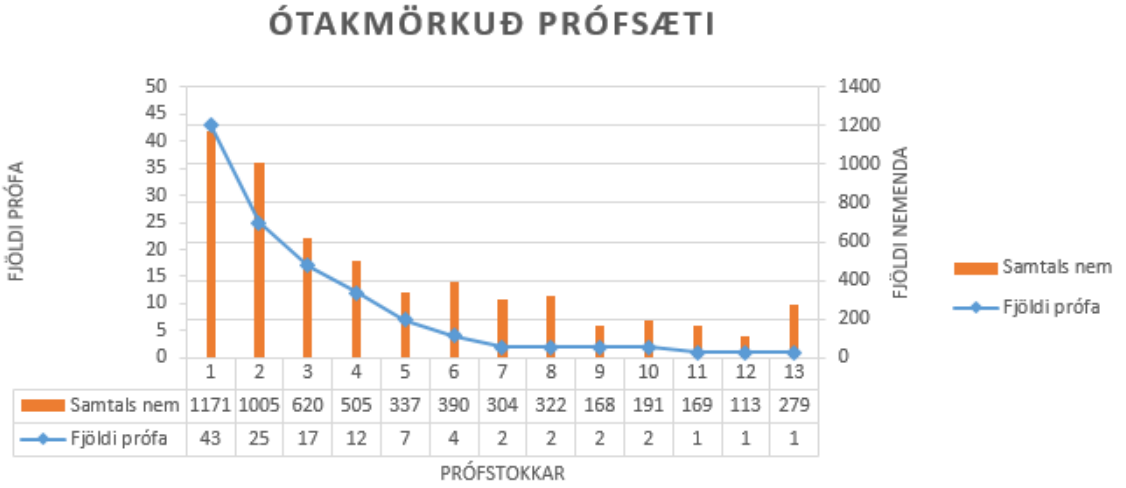
3 Niðurstöður, niðurlag og tillögur

Líkanið var þróað í nokkrum skrefum. Í öllum tilfellum, nema þess sé sérstaklega getið, var gert ráð fyrir því að nemandi gæti ekki setið tvö próf á sama tíma og að próf samkenndra áfanga séu á sama tíma. Einnig er gert ráð fyrir því að hámarks úthlutunarsæti fyrir hvern prófstokk séu 450 sæti. Byrjað var á því að kanna hver lágmarksfjöldi prófstokka er sem þarf til að búa til próftöflu með öllum hörðum skorðum sem settar voru hér að undan [graf 1] og einnig hve margir prófstokkarnir yrðu ef fjöldi sæta væri ótakmarkaður [graf 2].



	1	2	3	4	5	6	7	8	9	10	11	12	13
Samtals nem	450	450	447	450	449	450	448	444	405	446	443	413	279
Fjöldi prófa	12	27	13	13	15	8	7	9	3	5	3	3	1

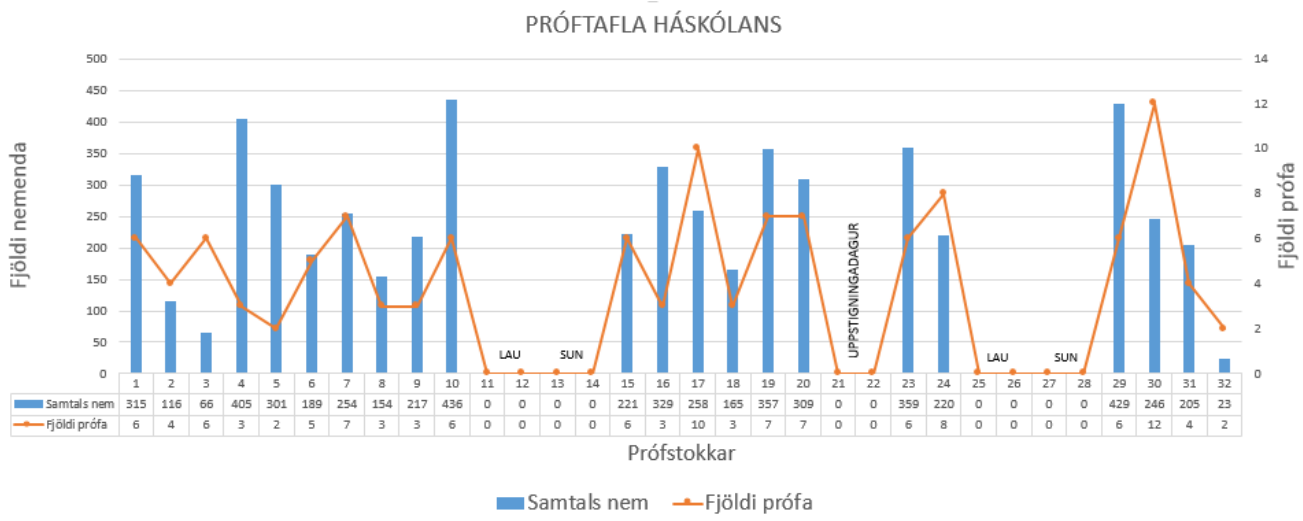
Mynd 1: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki. Fjöldi tiltækra sæta í hvern prófstokk var takmarkaður við 450.



	1	2	3	4	5	6	7	8	9	10	11	12	13
Samtals nem	1171	1005	620	505	337	390	304	322	168	191	169	113	279
Fjöldi prófa	43	25	17	12	7	4	2	2	2	2	1	1	1

Mynd 2: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki. Fjöldi tiltækra sæta í hverjum prófstokki var ótakmarkaður.

Í báðum tilvikum reyndist besta lausnin nota 13 prófstokka, þ.e. engu skipti hvort sætin sem voru tiltæk voru 450 eða að fjöldi sæta væri ótakmarkaður og því styttist próftímabilið ekkert þrátt fyrir að sætaskorðan væri fjarlægð. Hámarksfjöldi sæta sem þyrfti í seinna tilvikinu væri þó 1171 sæti. Dreifing nemenda er jafnari ef fjöldi prófsæta er takmarkaður heldur en þegar fjöldi prófsæta er ótakmarkaður. Því næst var fundinn heppilegur mælikvarði á hvíld milli prófa fyrir hópanna 61. Það var gert með því að finna meðalhvíldartíma milli prófa fyrir alla hópa. Fyrir auglýsta próftöflu háskólans fyrir vorið 2016 fengust neðangreind gögn [graf 3][tafla 1].

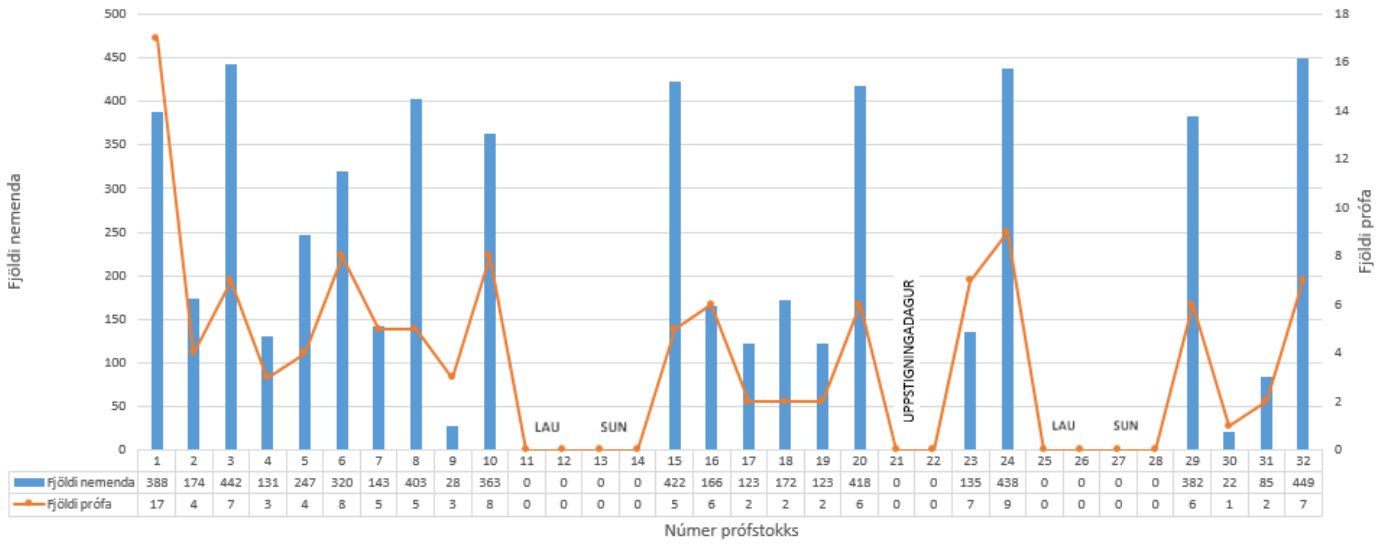


Mynd 3: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki fyrir auglýsta próftöflu Verkfræði- og náttúruvísindasviðs Háskóla Íslands fyrir vormisseri 2016.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	6,63	8,48
Fjöldi daga	2,90	3,72

Tafla 1: Taflan sýnir meðalhvíldartíma hópa miðað við auglýsta próftöflu háskólans

Fundin var betri lausn með því að byrja á að tryggja að sem fæstir nemendur væru í tveimur prófum sama dag og því næst var lögð áhersla á að sem fæstir nemendur væru í prófum tvo daga í röð. Þegar það líkan var keyrt fengust niðurstöður sem sýndar eru á grafi 4.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Fjöldi nemenda	388	174	442	131	247	320	143	403	28	363	0	0	0	0	422	166	123	172	123	418	0	0	135	438	0	0	0	0	382	22	85	449
Fjöldi prófa	17	4	7	3	4	8	5	5	3	8	0	0	0	0	5	6	2	2	2	6	0	0	7	9	0	0	0	0	6	1	2	7

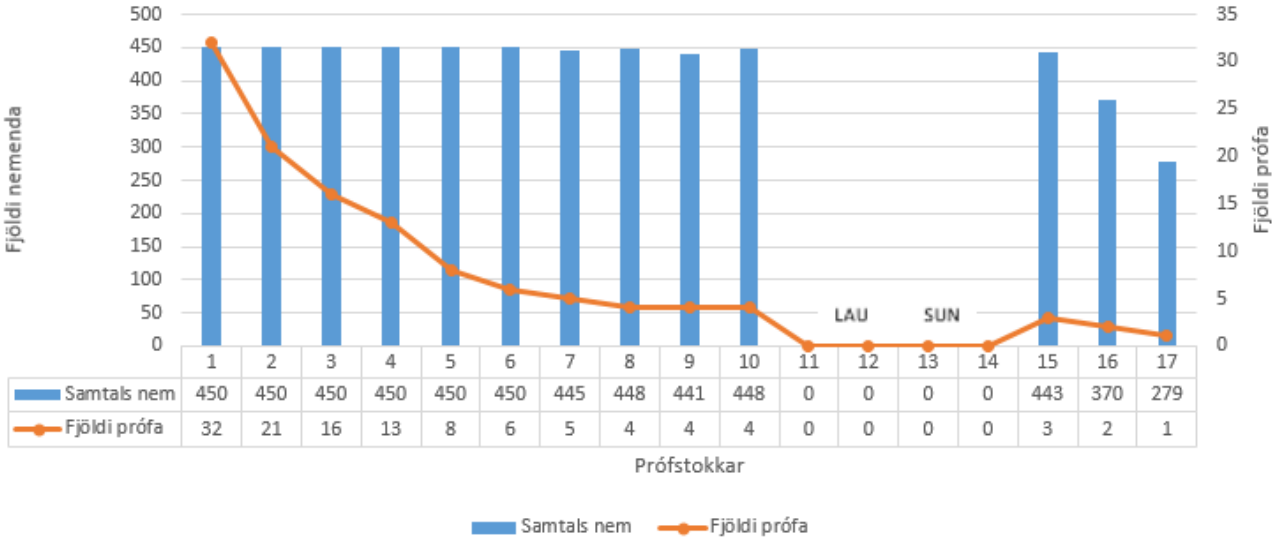
Mynd 4: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki þegar leitað var að lausn betri en auglýst próftafla frá háskólanum gaf.

Þessi lausn gefur að meðalhvíldartími frá byrjun próftímabils til síðasta prófs er lengri og jafnframt er meðalhvíldartími nemenda frá fyrsta og síðasta prófi fyrir hvern hóp lengri [tafla 2]. Fjöldi nemenda er rokkandi, allt frá 22 nemendum og upp í 449 nemendur. Þessi lausn væri hentug nemendum sem kjósa góðan hvíldartíma á milli prófa en væri síður kostur fyrir stjórnendur skólans.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	7,82	9,25
Fjöldi daga	3,48	4,08

Tafla 2: Taflan sýnir meðalhvíldartíma hópa við bættan hvíldartíma milli prófa.

Til þess að lágmarka fjölda prófstokka var líkanið hannað þannig að prófunum var raðað í fremstu prófstokkana og þar með tryggði það einnig að nemendur klára prófin á sem styðstum tíma.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Samtals nem	450	450	450	450	450	450	445	448	441	448	0	0	0	0	443	370	279
Fjöldi prófa	32	21	16	13	8	6	5	4	4	4	0	0	0	0	3	2	1

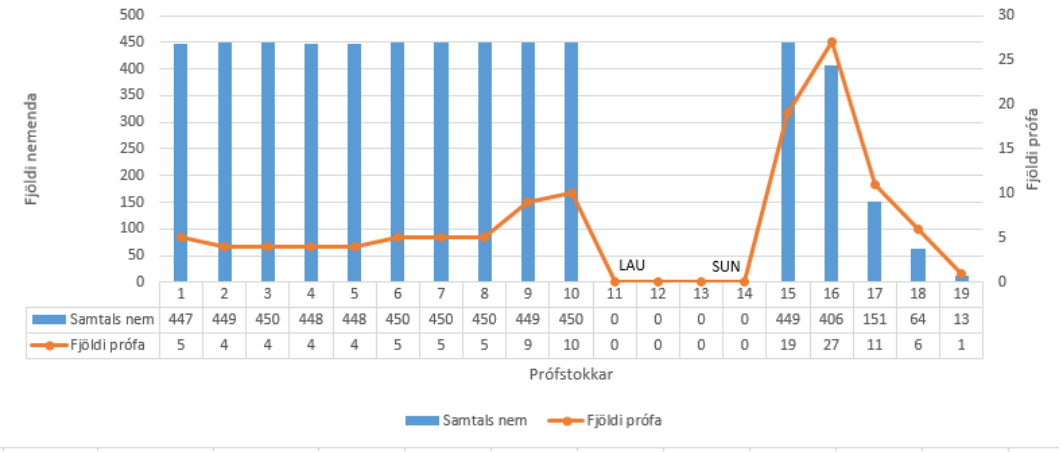
Mynd 5: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki þegar fengin var lausn við lágmörkun prófstokka.

Á grafinu [5] sést að fyrstu sex prófstokkarnir hafa flesta nemendur eða 450 talsins og eru því fullnýttir. Einungis í prófum sem þreytt eru í síðustu tveimur stokkunum er fjöldi nemenda færri en 400 en í stokki 16 eru 370 nemendur og í síðasta stokknum er fjöldi nemenda 279. Hvíldartíminn á milli prófa styttist en meðalhvíldartími frá byrjun próftímabils til síðasta prófs er 0,87 dagar og meðalhvíldartími frá fyrsta prófi til síðasta prófs fyrir hvern hóp er 0,94 dagar. Þar með sést að meðalhvíldartíminn frá byrjun til enda próftímabils styttist um 2,61 daga og meðalhvíldartími frá fyrsta prófi til síðasta prófs styttist um 3,14 daga.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	2,25	2,47
Fjöldi daga	0,87	0,94

Tafla 3: Taflan sýnir meðalhvíldartíma hópa þegar fjöldi prófstokka var lágmarkaður.

Kennarar vildu sjá próf fjölmennari námskeiða fyrr á tímabilinu og var líkaninu breytt til þess að koma á móts við óskir þeirra. Graf sex sýnir breytingu á dreifingu nemenda í prófstokka.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Samtals nem	447	449	450	448	448	450	450	450	449	450	0	0	0	0	449	406	151	64	13
Fjöldi prófa	5	4	4	4	4	5	5	5	9	10	0	0	0	0	19	27	11	6	1

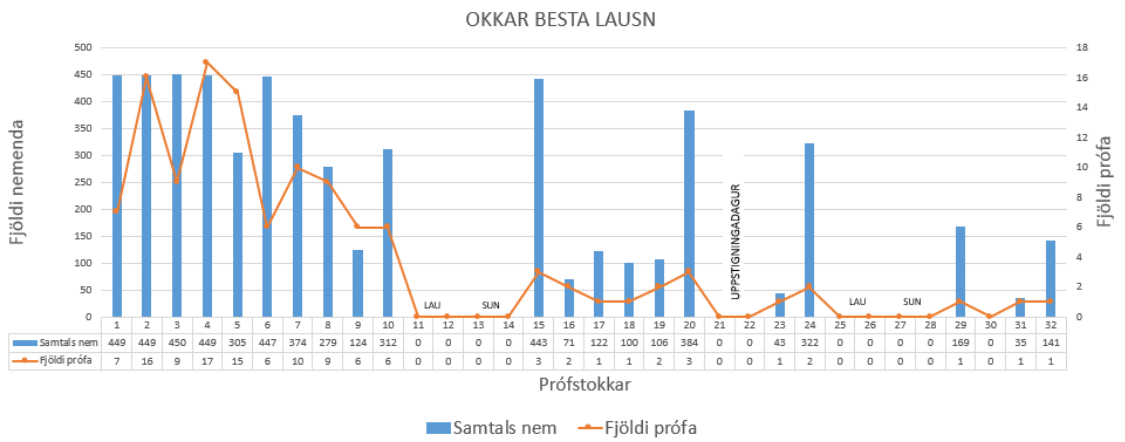
Mynd 6: Grafið sýnir fjölda nemenda og prófa í hverjum prófstokki. Hér er fjölmennustu námskeiðunum raðað fremst í prófstokkana.

Við það að raða prófum fjölmennustu námskeiða fremst á tímabilið eykst meðalhvíldartími hópa miðað við þá niðurstöðu er fékkst þegar fjölda prófstokka var lágmarkaður [tafla 3] en er þó töluvert minni en meðalhvíldartími auglýstrar próftöflu fyrir deildina vorið 2016 [tafla 1]. Niðurstöður meðalhvíldartíma hópa við þessa lausn má sjá í töflu 4. Dreifing nemenda yfir tímabilið er ekki ósvipuð og þegar fjöldi prófstokka var lágmarkaður, en heildarfjöldi nýttra prófstokka, þegar prófum fjölmennustu námskeiðunum er raðað fremst í próftöfluna, er 19. Síðustu þrír prófstokkarnir telja færri en 150 nemendur en aðrir nýttir prófstokkar fleiri en 400 nemendur.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	3,27	3,35
Fjöldi daga	1,41	1,42

Tafla 4: Taflan sýnir meðalhvíldartíma hópa þegar fjölmennustu námskeiðunum er raðað fremst í prófstokka.

Að lokum var stillt upp líkani sem talið er gefa bestu lausn fyrir nemendur, kennara og stjórnendur skólans. Við líkangerðina var í fyrsta lagi haft í huga að fjölmenn námskeið væru eins framarlega á próftímabili og unnt væri. Í öðru lagi var reynt eftir fremsta megni að tryggja að nemendur væru ekki í tveimur prófum sama dag og að þeir hefðu að minnsta kosti einn prófstokk í hvíld á milli prófa helst þó heilan dag. Í þriðja lagi voru próf með háa fallprósentu sett eins framarlega á próftímabil og unnt var. Að lokum var einnig haft í huga að stjórnendur skólans vilja spara kostnað og þar með var reynt að minnka fjölda nýttra stokka.



Mynd 7: Grafið sýnir hagkvæmnustu lausn sem fengin var samkvæmt þeim skorðum sem settar voru.

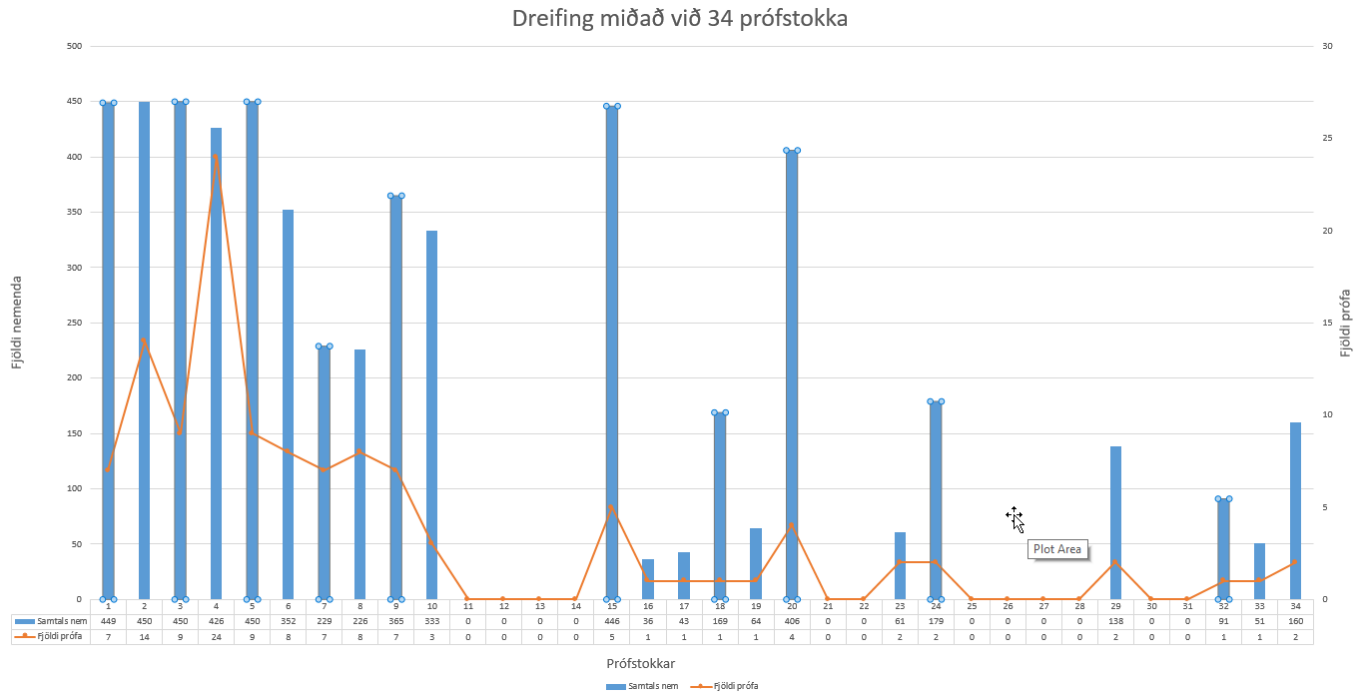
	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	5,98	7,13
Fjöldi daga	2,56	3,05

Tafla 5: Taflan sýnir meðalhvíldartíma hópa miðað við hagkvæmustu lausn.

Líkanið er betra en lausn Háskóla Íslands að því leyti að það notar einum færri prófstokk og þar með minnkar kostnaður við prófhald, sem um nemur yfirsetufólki og kennslustofum. Hvíldartími er ögn minni, u.þ.b. hálfur dagur, heldur en auglýsta lausnin en er þó ásættanlegur fyrir nemendur. Líkanið var hannað þannig að ef fjöldi nemenda sem taka próf í tveimur mismunandi fögum var nægur átti lausnin að hafa ákveðinn fjölda stokka á milli. Eftir nokkrar prófanir kom í ljós að hentugast var, til þess að tryggja sem mestan heildarhvíldartíma, að hafa a.m.k. einn stokk á milli prófa ef fjöldi nemenda í báðum fögum var meiri en eða jafn tveimur, tvo stokka ef fjöldi nemenda var meiri en eða jafn níu og þrjá stokka hið minnsta ef fjöldi nemenda var meiri en eða jafn 15. Þar með tókst að tryggja að sem flestir nemendur fái að minnsta kosti þriggja stokka frí, þ.e.a.s. einn og hálfan dag, á milli prófa. Þar að auki voru "fjölmenn" námskeið færð framar á próftímabilið en það var gert að ósk kennara því það hentar þeim betur til yfirferðar. Hægt var að hafa allra stærstu námskeiðin fremst í uppröðuninni en það reyndist óhagstætt þar sem önnur fjölmenn námskeið lentu í mun verri stöðu. Því var ákveðið að nota lausn sem tekur tillit til heildarinnar en ekki einungis stærstu námskeiðanna.

Einnig hefur verið tryggt að próf með hærri fallprósentu, þ.e.a.s. strembin próf, eru framarlega í uppröðuninni sem nemendum hefur þótt ákjósanlegt. Að endingu var líkanið fyrir bestu lausn keyrt fyrir mismunandi fjölda prófstokka. Ekki fékkst lögleg lausn þegar prófstokkum var fækkað. Hinsvegar, þegar þeim var fjölgað fengust mismunandi lausnir sem gáfu eftirfarandi hvíldartíma.

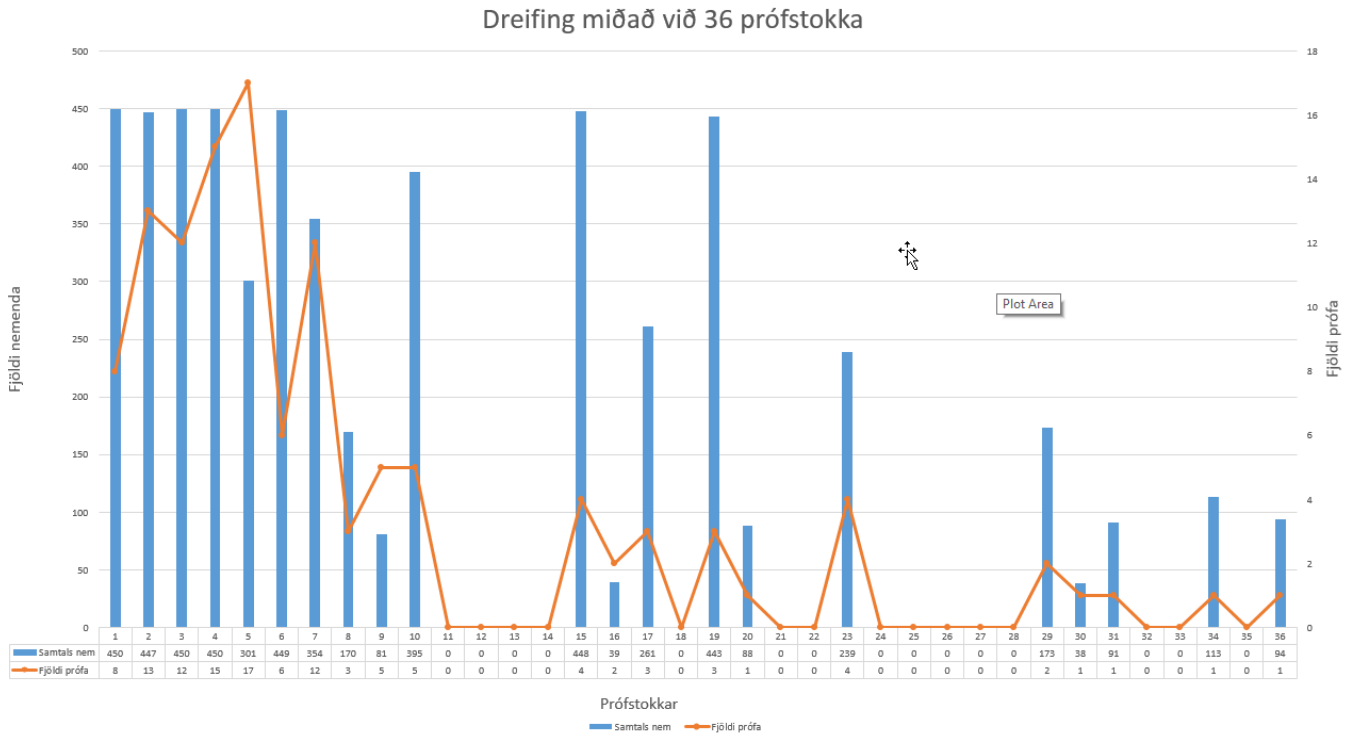
Að lokum var prófað hvernig lausn myndi breytast ef fjöldi prófstokka yrði breyttur, en það kom í ljós að 32 prófstokkar er algjört lágmark en hinsvegar væri hægt að bæta við prófstokka, það kom í ljós að 38 stokkar væri hámarksfjöldi stokka sem gæfi nýja lausn í próftöflubestuninni.



Mynd 8: Grafið sýnir hagkvæmnustu lausn fyrir 34 prófstokka.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	6,25	5,08
Fjöldi daga	2,62	2,13

Tafla 6: Taflan sýnir meðalhvíldartíma hópa miðað við 34 prófstokka.

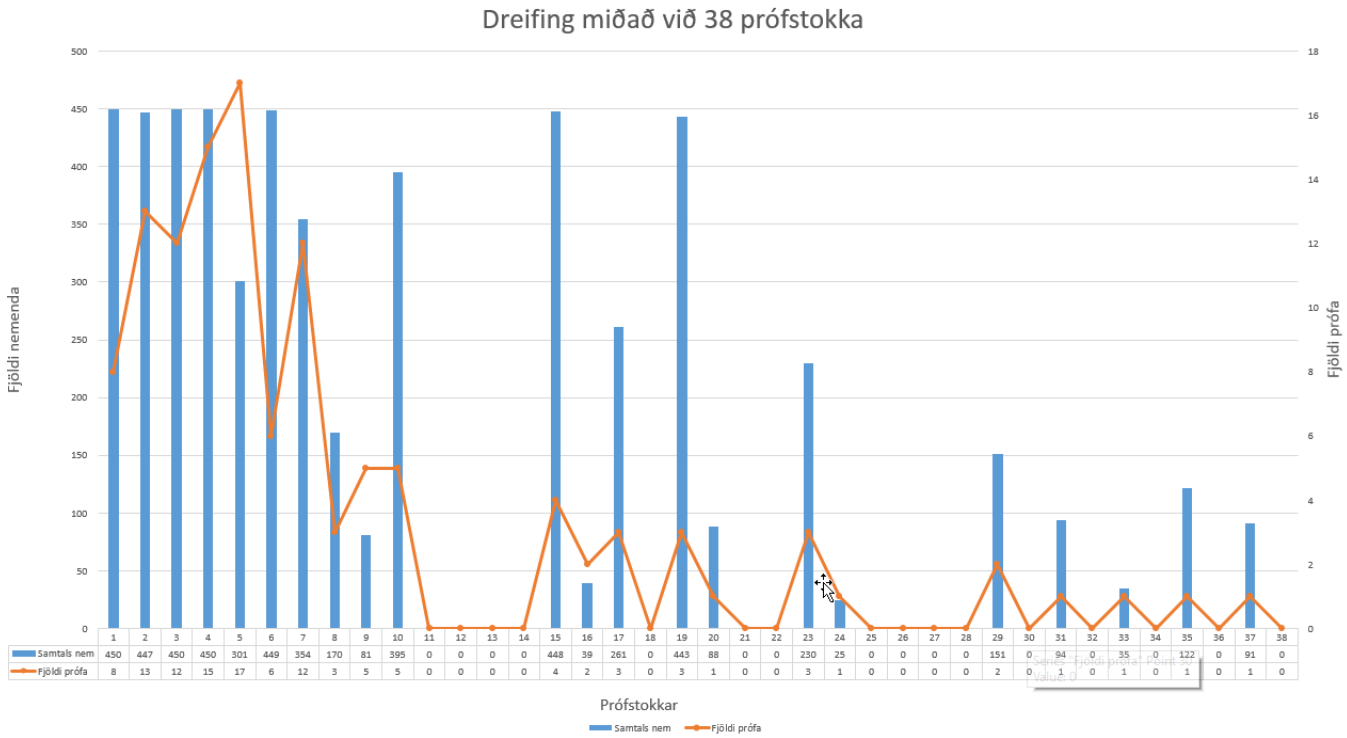


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Samtals nem	450	447	450	450	301	449	354	170	81	395	0	0	0	0	448	39	261	0	443	88	0	0	239	0	0	0	0	0	173	38	91	0	0	113	0	94
Fjöldi prófa	8	13	12	15	17	6	12	3	5	5	0	0	0	0	4	2	3	0	3	1	0	0	4	0	0	0	0	0	2	1	1	0	0	1	0	1

Mynd 9: Grafið sýnir hagkvæmnustu lausn fyrir 36 prófstokka.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	5,87	4,85
Fjöldi daga	2,49	2,06

Tafla 7: Taflan sýnir meðalhvíldartíma hópa miðað við 36 prófstokka.



Mynd 10: Grafið sýnir hagkvæmnustu lausn fyrir 38 prófstokka.

	Allt prófatímabili	**Prófatímabil hópa**
Fjöldi stokka	6,07	4,96
Fjöldi daga	2,62	2,13

Tafla 8: Taflan sýnir meðalhvíldartíma hópa miðað við 38 prófstokka.

Hér sést að ef prófstokkunum er fjölgað í annaðhvort 34 eða 38 þá eykst meðalhvíldartími yfir allt prófatímabilið (fyrir 36 stokka þá verður nýtingin það léleg að hvíldartíminn mínnkar allstaðar), en hinsvegar bitnar það verulega á ákveðnum hópum þar sem meðalhvíldartíminn fyrir próftímabil hópa mínnkar allstaðar, auk þess þarf að hafa í huga að eftir því sem prófstokkum er fjölgað þá jafnframt því eykst kostnaður, þar sem það þarf að nýta stofur yfir lengra tímabil ásamt því að greiða þarf yfirsetufólk og kennarar þurfa að fara á milli stofa í stað þess að geta nýtt tímann til prófyfirferðar.

4 Aðferðir

Notast var við bæði `AMPL` og `JavaScript` forritun til að búa til líkan og vinna úr upplýsingum sem úr því fengust. Mælt er með að nota `GLPK` forritasafnið til að þýða líkanið yfir á `.lp` skráartegund, `Gurobi` til að keyra `.lp` líkanið og fá lausn ásamt því að nota `Node.js` til að keyra `JavaScript` forritið sem fylgir.

Byrjað var á að safna saman gögnum í gagnaskrá (`profrodun2016.dat`). Þar voru skráðir þeir prófstokkar sem ekki eru nýttir til prófa þ.e.a.s. þeir stokkar sem lenda á helgum, á uppstigningardegi eða öðrum frídögum. Þessir prófstokkar voru skilgreindir í menginu `offSlots` í skránni. Því næst voru gögn fyrir fallprósentu sett inn, sem fengust hjá prófstjóra, en fallprósentan var skilgreind sem `cidDifficulty`. Fallprósentan er nýtt til þess að geta sett próf með hárri fallprósentu, erfið próf, framarlega á tímabilið. Þá var bætt inn fylki `conjoinedCourses` fyrir samkennd námskeið. Þau námskeið sem eru samkennd hafa einn þar sem lína fyrra námskeiðs sker dálk seinni námskeiðs, annars núll ef þau eru ekki samkennd. Heiti námskeiða er geymt í vigrinum `cidExam`. Mengin `group[x]`, þar sem `x` er á bilinu einn upp í 61, geyma námskeið sem tilheyra kjörsviði `x`. Fylkið `cidCommon` geymir upplýsingar um fjölda þeirra nemenda sem eru í sameiginlegum námskeiðum.

Inni í líkaninu (`profrodun2016.mod`) má stilla `n` til að fjölga eða fækka dögum sem eru á próftímabilinu. Einnig þarf að stilla `sumCount` í samræmi við nemendafjölda. Að lokum er hægt að stilla "studentsTolerance" fyrir það hversu háan fjölda nemenda, sem ekki uppfylla skorðu, þarf svo að skorða sé þó ennþá virk. Hægt er að stilla þrjú "studentsTolerance" svo að þolmörkin geti verið mismunandi fyrir hverja skorðu. Til þess að keyra líkanið er byrjað á að velja þær skorður sem eiga að vera inni, þ.e. valið hvað lausnin þarf að uppfylla og einnig er valið það markfall sem við á. Hinar skorðurnar og markföllin eru kommentuð út með # tákni.

`JavaScript` reiknirit (`dataProcessing.js`) var notað til þess að reikna meðalhvíldartíma milli prófa, í stokkum og dögum, annars vegar frá upphafi prófatímabils og hins vegar frá fyrsta prófi hóps til síðasta prófs. Gögnin sem reikna á úr þarf að setja upp í gagnaskrá (`dataFile.js`). Undir hverja próftöfluröðun þarf eitt fylki með tveimur línum. Fyrri línan er strengjafylki með heiti námskeiða og seinni línan er heiltöluvigur með viðeigandi prófstokksnúmer fyrir hvert fag. Einnig þarf að setja inn strengjafylki fyrir hvern hóp með viðeigandi námskeiðsheitum. Mælt er með því að nota `Excel` eða ritil með virkni fyrir reglulegar segðir, eins og `Atom`, til að vinna úr gögnum úr líkaninu eða mögulega að skrifa nýtt fall til þess. Passa þarf að flytja gögnin yfir í gagnavinnsluskrána með `export` skipun.

Innan í gagnavinnsluskránni (`dataProcessing.js`) þarf að taka á móti gögnunum með `require` skipun. Þegar það er komið má notfæra sér `processDataMatrix()` fallið til að reikna út meðalbiðtíma fyrir alla hópana með gefinni lausn. Frekari lýsing á fallinu og virkni þess má finna í gagnavinnsluskránni. Reikniritin má finna í viðauka skýrslunnar. Búið er að gefa lýsingar á öllum föllum í gagnavinnsluskránum og sjá má útskýringar á öllum skorðum og ákvörðunum sem hafa verið teknar varðandi hvernig besta eigi próftöfluna.

5 Almenn umfjöllun

Líkanið er sérhannað til þess að eiga við gerð próftaflna en það má nýta í önnur verkefni sem raða atburðum niður á tíma. Ef haldin er ráðstefna eða syrpa af fundum á stuttum tíma þar sem ákveðnir aðilar verða að vera viðstaddir í ákveðnum fundum eða fyrirlestrum mætti nota þetta líkan til þess að raða fundunum og fyrirlestrunum í dagskrá. Einnig mætti nota líkanið, eða hluta úr því, til að raða stundatöflum fyrir námskeið. Þá væri hægt að endurskrifa skorðurnar til þess að hafa sem minnst bil milli námskeiða í stundatöflu og hafa aðrar skorður til að raða námskeiðum sem minnst á kvöldin.

Eflaust eru til fleiri verkefni þar sem þarf að raða ákveðna tíma á ákveðna atburði (vaktaskipulag og annað slíkt) en líklega þyrfti að breyta skorðum að verulegu leyti eða bæta nýjum skorðum við til að verða við slíkum verkefnum.

Þegar verið var að reikna hvíldartíma fyrir hópa í þessu verkefni var eingögnu notast við hvíldartíma fyrir hvern hóp og hver hópur hafði jafnmikið vægi. Til að auka nákvæmni á því hversu mikill hvíldartími ávinnst væri betra að hafa fjölda nemenda í hverjum hópi og taka vegið meðaltal með tilliti til þess. Einnig væri hægt að raða námskeiðum sem fjölmennir hópar taka á "betri" staði í röðuninni með skorðum sem nota fjölda nemenda í hópum. Einnig má nota aðrar leiðir til að meta erfiðleika en fallprósenta. Að öllum líkindum eru til betri aðferðir og mögulega til gögn um erfiðleikastig en ekki var notast við slíkt í þessu líkani.

6 Heimildir

Gögn sem nýtt voru við gerð verkefnisins:

- Gögn sem fylgdu verkefnalýsingu
- Gögn um fallprósentu fengin hjá prófstjóra
- AMPL handbók

7 Viðauki

7.1 AMPL kóði

profrodun2016.mod

```
# Usage:
#  glpsol --check -m profrodun2016.mod -d profrodun2016.dat
#  glpsol -m profrodun2016.mod -d profrodun2016.dat --wlp proftafla.lp
#  gurobi_cl TimeLimit=3600 ResultFile=proftafla.sol proftafla.lp

set cidExam; # Set of courses
set group{1..61} within cidExam; # Defined programs (namsbrautir/leidir)
#set noExamDays;

param n := 16; # Number of days in the exam period
set examSlots := 1..(2*n); # Exam-slots (profstokkar)
set offSlots; #Set of slots that belong to off Days

param sumCount := 2274; # Total numnber of exams taken during this exam period
param cidExamslot2016{cidExam}; # Solution of the University of Iceland, for comparison
param ourBasicSolution{cidExam}; # Calculated solution with 3 basic constraint
param solutionWithoutSeats{cidExam}; # Calculated solution without seat constraint
param cidDifficulty{cidExam}; # Percentage of students that did not pass the exam last year

param cidCount{cidExam} default 0; # Amount of students in each course
param cidCommon{cidExam, cidExam} default 0; # Amount of students that take co-taught courses
param conjoinedCourses{cidExam, cidExam} default 0; # Vector containing courses that are taught jointly

# Parameters to indicate how many students have to be in an exam-clash for constraints to work
param studentsTolerance := 2;
param studentsTolerance2 := 9;
param studentsTolerance3 := 15;
#param studentsTolerance4 := 13;


var slot{cidExam, examSlots} binary; # Variable

# This constraint is used to coerce the solution to be the same as the one of the University
#subject to lookAtSolution{e in examSlots, c in cidExam:
                          #cidExamslot2016[c] == e}:  slot[c,e] = 1;

```

```
# This constraint its used to coerce the solution to be one of our own solutions
# subject to coerceSolution{e in examSlots, c in cidExam:
#  solutionWithoutSeats[c] == e}: slot[c,e] = 1;

# Objective function to place all exams as early as possible in exam-table
#minimize totalSlots: sum{c in cidExam, e in examSlots} slot[c,e]*(e^8);

# Courses with the most students have exams in the beginning of exam period
#minimize totalSlots: sum{c in cidExam, e in examSlots} slot[c,e]*(cidCount[c]*(e^2))^4;

# Our best solution: Difficult exams early
minimize totalSlots: sum{c in cidExam, e in examSlots} slot[c,e]*((cidCount[c]*((cidDifficulty[c] + 1)^4))/ sumCount)*(e^0.25);

# Ensure that no students have exams in two different courses at the same time
 subject to examClashes{c1 in cidExam, c2 in cidExam, e in examSlots : cidCommon[c1, c2] > 0}: slot[c1,e]+slot[c2,e] <= 1;

# Ensure that each course has exactly one exam in the table
 subject to hasExam{c in cidExam}:sum{e in examSlots}slot[c,e] = 1;

# Ensure that all students assigned to slot have a seat to take an exam
 subject to maxInSlot{e in examSlots}:sum{c in cidExam}slot[c,e]*cidCount[c] <= 450;

# Conjoined courses have exams in same slot
 subject to jointlyTaught{c1 in cidExam, c2 in cidExam, e in examSlots: conjoinedCourses[c1,c2] <>0}:slot[c1,e]=slot[c2,e];

#Ensure that there are no exams on weekends and holidays
subject to noExams{c in cidExam, e in examSlots: e in offSlots}: slot[c,e] = 0;

#Ensure that a student is not in exam slots side by side
subject to examSpace{e in examSlots, c1 in cidExam, c2 in cidExam: cidCommon[c1, c2] >= studentsTolerance && e+1 in examSlots}: slot[c1,e]+slot[c2, e+1] <= 1;

#Ensure that a student is not in exam slots e and e+2
subject to examSpace2{e in examSlots, c1 in cidExam, c2 in cidExam: cidCommon[c1, c2] >= studentsTolerance2 && e+2 in examSlots}: slot[c1,e]+slot[c2, e+2] <= 1;

#Ensure that a student is not in exam slots e and e+3
subject to examSpace3{e in examSlots, c1 in cidExam, c2 in cidExam: cidCommon[c1, c2] >= studentsTolerance3 && e+3 in examSlots}: slot[c1,e]+slot[c2, e+3] <= 1;
```

```
73
74 #Ensure that a student is not in exam slots e and e+4
75 #subject to examSpace4{e in examSlots, c1 in cidExam, c2 in cidExam:
       cidCommon[c1, c2] >= studentsTolerance4 && e+4 in examSlots}:
   slot[c1,e]+slot[c2, e+4] <= 1;
76
77 # Does the exam table for 2016 fulfil the demands for programs:
78 check {i in 1..61, c1 in group[i], c2 in group[i]: cidCommon[c1,c2]
   > 0}
79                                cidExamslot2016[c1] <> cidExamslot2016[
                                      c2];
80 # Does the exam table for 2016 fulfil the demands for joined
   students:
81 check {c1 in cidExam, c2 in cidExam: cidCommon[c1,c2] > 0}
82                                cidExamslot2016[c1] <> cidExamslot2016[
                                      c2];
83
84 solve;
85
86 # Check how many students are in each exam-slot...
87 for {e in examSlots} {
88   printf : "Amount of students in exam-slot %d are %d\n", e, sum{c
         in cidExam}
89                                                    slot[c,e] * cidCount
                                                         [c];
90 }
91
92 end;
```

7.2 Javascript kóði

dataProcessing.js

```
1
2 // Fetch data to be processed
3 var dataFile = require("./dataFile.js");
4
5 // Import data
6 dataMatrixWithConstraints = dataFile.dataMatrixWithConstraints;
7 dataMatrixWithoutSeats = dataFile.dataMatrixWithoutSeats;
8 dataMatrixUniversity = dataFile.dataMatrixUniversity;
9 dataMatrixRestEPlusTwo = dataFile.dataMatrixRestEPlusTwo;
10 dataMatrixRestEPlusThree = dataFile.dataMatrixRestEPlusThree;
11 dataMatrixRestEPlusFour = dataFile.dataMatrixRestEPlusFour;
12 dataMatrixOptimumRest = dataFile.dataMatrixOptimumRest;
13 dataMatrixOptimumRestTwo = dataFile.dataMatrixOptimumRestTwo;
14 dataMatrixPartC = dataFile.dataMatrixPartC;
15 dataMatrixStudents = dataFile.dataMatrixStudents;
16 dataMatrixOurSolution = dataFile.dataMatrixOurSolution;
17 dataMatrixOurSolutionTwo = dataFile.dataMatrixOurSolutionTwo;
18 dataMatrixOurSolutionThree = dataFile.dataMatrixOurSolutionThree;
19 dataMatrixOurSolutionFour = dataFile.dataMatrixOurSolutionFour;
20 groupArray = dataFile.groupArray;
```

```
21
22 // Function to calculate how many rest days a group receives on
       average. Takes
23 // the group, which exam setup is to be used (dataMatrix) and a
       boolean value
24 // (addPrecedingDays) to tell whether days/slots before the first
       exam are to
25 // be calculated as rest days or not.
26 // Returns array[2] with average amount of rest in slots (first
       value) and
27 // average amount of rest in days (second value) for the group.
28 var processGroup = function(group, dataMatrix, addPrecedingDays){
29   // Function variables for calculations
30   var restDataArray = [];
31   var restDataArrayInDays = [];
32   var restSumInSlots = 0;
33   var restSumInDays = 0;
34   // Add slots, where group has to take exams, to restDataArray
35   for(g in group){
36     for(d in dataMatrix[1]){
37       if(dataMatrix[1][d].localeCompare(group[g]) === 0){
38         restDataArray.push(dataMatrix[2][d]);
39       }
40     };
41   };
42   // Add first days/slots into calculations if necessary
43   if(addPrecedingDays){
44     restDataArray.push(0);
45   }
46   // Sort restDataArray in ascending order
47   restDataArray.sort(function(x,y){return x-y});
48   // Convert data to days in stead of slots
49   restDataArrayInDays = restDataArray.map(function(x){return Math.
       ceil(x/2)});
50   // Calculate average amount of rest that each group receives
51   for(var r = 0; r < restDataArray.length - 1; r++){
52     restSumInSlots += restDataArray[r+1] - restDataArray[r] - 1;
53     if(restDataArrayInDays[r+1] - restDataArrayInDays[r] > 0){
54       restSumInDays += restDataArrayInDays[r+1] -
           restDataArrayInDays[r] - 1;
55     }
56   }
57   // Divide by amount of periods that are between exams
58   if(restDataArray.length !== 1){
59     restSumInSlots = restSumInSlots/(restDataArray.length - 1);
60     restSumInDays = restSumInDays/(restDataArrayInDays.length - 1);
61   }
62   return [restSumInSlots, restSumInDays];
63 };
64
65 // Function for processing each exam table setup. Takes an array
       with the
66 // courses each group takes (groups), the exam table (dataMatrix),
       and
```

```
67 // a boolean value (addPrecedingDays) to determine whether the days/
       slots
68 // preceding the first exam should be calculated as rest periods.
69 // Prints outcomes to console.
70 var processDataMatrix = function(groups, dataMatrix,
       addPrecedingDays){
71   totalSumInSlots = 0;
72   totalSumInDays = 0;
73   for(g in groups){
74     groupResult = processGroup(groups[g], dataMatrix,
         addPrecedingDays);
75     totalSumInSlots += groupResult[0];
76     totalSumInDays += groupResult[1];
77   }
78   avgSlots = totalSumInSlots/groups.length;
79   avgDays = totalSumInDays/groups.length;
80   if(addPrecedingDays){
81     console.log("Preceding days added: ");
82   }
83   else{
84     console.log("Preceding days not added: ");
85   }
86   console.log("Average amount of rest in slots for groups: " +
       avgSlots);
87   console.log("Average amount of rest in days for groups: " +
       avgDays);
88 };
89
90 console.log("Data with 3 basic constraints (part A): ");
91 processDataMatrix(groupArray, dataMatrixWithConstraints, false);
92 processDataMatrix(groupArray, dataMatrixWithConstraints, true);
93 console.log("
       ----------------------------------------------------------");
94 console.log("Data without seat constraint (part A): ");
95 processDataMatrix(groupArray, dataMatrixWithoutSeats, false);
96 processDataMatrix(groupArray, dataMatrixWithoutSeats, true);
97 console.log("
       ----------------------------------------------------------");
98 console.log("Data from University (part B):");
99 processDataMatrix(groupArray, dataMatrixUniversity, false);
100 processDataMatrix(groupArray, dataMatrixUniversity, true);
101 console.log("
       ----------------------------------------------------------");
102 console.log("Data with rest e+2 (part B):");
103 processDataMatrix(groupArray, dataMatrixRestEPlusTwo, false);
104 processDataMatrix(groupArray, dataMatrixRestEPlusTwo, true);
105 console.log("
       ----------------------------------------------------------");
106 console.log("Data with rest e+3 (part B):");
107 processDataMatrix(groupArray, dataMatrixRestEPlusThree, false);
108 processDataMatrix(groupArray, dataMatrixRestEPlusThree, true);
109 console.log("
       ----------------------------------------------------------");
110 console.log("Data with rest e+4 (part B):");
```

```
processDataMatrix(groupArray, dataMatrixRestEPlusFour, false);
processDataMatrix(groupArray, dataMatrixRestEPlusFour, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with optimum rest solution(1,8,20 tolerance) (part
    B):");
processDataMatrix(groupArray, dataMatrixOptimumRest, false);
processDataMatrix(groupArray, dataMatrixOptimumRest, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with optimum rest solution(2,9,15 tolerance) (part
    B):");
processDataMatrix(groupArray, dataMatrixOptimumRestTwo, false);
processDataMatrix(groupArray, dataMatrixOptimumRestTwo, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with exams as early as possible (part C):");
processDataMatrix(groupArray, dataMatrixPartC, false);
processDataMatrix(groupArray, dataMatrixPartC, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with greater number of students earlier (part D):"
    );
processDataMatrix(groupArray, dataMatrixStudents, false);
processDataMatrix(groupArray, dataMatrixStudents, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with our optimum solution (part E):");
processDataMatrix(groupArray, dataMatrixOurSolution, false);
processDataMatrix(groupArray, dataMatrixOurSolution, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with our optimum solution, version 2 (part E):");
processDataMatrix(groupArray, dataMatrixOurSolutionTwo, false);
processDataMatrix(groupArray, dataMatrixOurSolutionTwo, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with our optimum solution, version 3 (part E):");
processDataMatrix(groupArray, dataMatrixOurSolutionThree, false);
processDataMatrix(groupArray, dataMatrixOurSolutionThree, true);
console.log("
    ----------------------------------------------------------------");
console.log("Data with our optimum solution, version 4 (part E):");
processDataMatrix(groupArray, dataMatrixOurSolutionFour, false);
processDataMatrix(groupArray, dataMatrixOurSolutionFour, true);
```