TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Raul Land 182999IAPM

LÕPUTÖÖ TEEMADE JA TUDENGITE AUTOMATISEERITUD KOKKUVIIMINE NING TÖÖDE PAREMUSJÄRJESTUSE KOOSTAMINE OSALISTE JÄRJESTUSTE PÕHJAL

magistritöö

Juhendaja: Gunnar Piho
PhD

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Raul Land

12.05.2020

Annotatsioon

Antud magistritöö eesmärgiks oli uurida, kuidas lahendada ülikoolis lõputöödega seonduvaid probleeme teemade jagamise ja hindamise osas. Probleemidele otsiti lahendust algoritmide kujul, mida saaks rakendada disainitavas infosüsteemis. Süsteemi eesmärgiks on koondada lõputöödega seonduv informatsioon. Töö tulemusena valmis prototüüp infosüsteemist koos algoritmidega, mis võimaldab tudengite ja juhendajate eelistuste põhjal automatiseeritud teema-tudengi paaride leidmist. Samuti valmis algoritm demonstreerimaks, kuidas retsensentide individuaalsete osaliste järjestuste põhjal moodustada hinnatud töödest globaalne paremusjärjestus, sealjuures omades tähenduslikku järjekorda ka sama hindega tööde puhul.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 5 peatükki, 14 joonist.

Abstract

Automated Matching of Theses Topics to Students and Ranking Submissions Based on Partial Graded Orderings

The goal of this master's thesis was to find solutions for problems related to graduation theses. These problems include finding a suitable thesis topic and automated ranking of graded theses. Currently there is not a system in place to facilitate these kinds of processes. Part of this thesis is a prototype system that was designed and implemented in order to demonstrate practicality of results. It also serves as a basis for testing and analysing these results and for further research on this topic.

Different graph theory algorithms were researched for the purpose of calculating matches based on the preferences of students and professors as the problem closely resembles a bipartite graph matching problem. Students must pick and order a limited set of topics they are interested in and professors must order a set of students, who have applied to their topic. Based on this data a result is computed that attempts to find topics for each student.

Other objective was to find a way how to rank all graded theses based on assessments of reviewing professors. As typically a single reviewer does not assess all the theses, the problem rose of how to aggregate these partial rankings into a global ranking of all theses. Voting algorithms and machine learning were considered to solve this problem to create a global ranking of assessed theses which could improve objectivity of grading process.

It is found that matching algorithms can solve the problem of matching students to their preferred topics and vice versa, but more optimizations regarding the quality of matches could be possible. It is also concluded that modified Borda count can largely solve the problem of ordering, but its effectiveness depends on the amount of found ties and methods used to break them. The thesis is in Estonian and contains 31 pages of text, 5 chapters, 14 figures.

Lühendite ja mõistete sõnastik

CRUD	*Create, Read, Update, Delete,* lühend iseloomustamaks põhioperatsioone andmetega infosüsteemis
CSV	*Comma Seperated Values*, reeglistik andmete hoiustamiseks
HTTP	*Hypertext Transfer Protocol,* rakenduse tasemel andmevahetusprotokoll
JDBC	*Java Database Connectivity*, üldine liides Java rakendustele andmebaasiga suhtluseks
JPA	*Java Persistence API*, Java programmeerimiskeele liides püsivate andmetega opereerimiseks
JSON	*JavaScript Object Notation,* standard struktureeritud andmete esitamiseks
Maven	Programmikoodi kompileerimise tööriist
ORM	*Object Relational Mapping* programmeeritud andmetüüpide teisendus andmebaasi tasandil tabeliteks ja tulpadeks
REST	*Representational state transfer*, võrgurakenduste arhitektuuri tüüp
social choice theory	Kollektiivsete otsuste põhjal eesmärgi saavutamist uuriv teadusharu/teadussuund

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tudengite viimaseks takistuseks diplomi saamisel on üldjuhul lõputöö kirjutamine. Töö enda kirjutamine on iga tudengi isiklik kohustus ja seda on otseselt raske mõjutada. Samas saab ülikool endast tulenevalt tagada keskkonna ja vahendid, et kaasa aidata eduka teadustöö kirjutamisele. Hea töö kirjutamisel on tähis osa ka motiveeritusel – inimesed, kes on teemaga seotud peavad olema motiveeritud ja huvitatud antud valdkonnast, et saavutada parim tulemus. Seega on lõpptulemuse juures oluliseks faktoriks teema valik. Saab eeldada et tudeng, kes sai enda lemmikteema, on palju motiveeritum kui see, kes valis teema juhuslikult või sai olude sunnil teema, mis on talle vastumeelne ja ei paku huvi. Kuid sama kehtib ka teise osapoole kohta – ka teemasid juhendavad õppejõud on huvitatud sellest, et nad saavad juhendada tudengeid, kes on ka nende väljapakutud teemadest reaalselt huvitatud. Vastavalt sellele saab väita, et lõputöö edukuses mängib rolli ka mõlema poole rahulolu antud olukorraga, ehk nii tudeng kui ka õppejõud on valinud endale sellise koosluse, kus mõlemad osapooled tunnevad, et nad on saanud õiglase koostööpartneri. Sellise olukorra saavutamiseks on vajalik, et nii tudengid kui ka juhendajad oleksid teadlikud kogu olukorrast: tudengite soov on teada kõiki valikuvõimalusi pakutavate lõputööde osas ja juhendajate soov on teada kõiki tudengeid, kes on reaalselt nende väljapakutud teemadest huvitatud.

Kvaliteetne hindamine on kõikide huvides – tudengid saavad õiglase hinde vastavalt nende panusele ja ülikool saab läbi selle väärtuslikku tagasisidet õpetamismetoodika ja õppekava edukuse kohta. Lõputööde hindamine informaatika õppekavadel toimub läbi hindamiskomisjoni, mille liikmed määravad töö lõpliku hinde skaalas 0-5, võttes arvesse erinevaid hindamiskriteeriume [1], sealhulgas juhendaja ning retsensendi arvamust. See on hindajate jaoks keeruline protsess, sest suur hulk töid tuleb hinnata õiglaselt ja suhteliselt lühikese perioodi jooksul piiratud informatsiooniga. Üks raskemaid tegevusi selle juures on suure hulga tööde asetamine hindamisskaalale 0-5 selliselt, et individuaalsed hinded oleksid võimalikult objektiivsed ja õiglased ka teiste hinnete suhtes. Paratamatult on töid rohkem kui võimalikke hindeid ja rohkem kui üks töö saab sama hinde osaliseks, kuid samuti tuleb otsustada, kust kohas algab kõrgem või madalam hinne ja millised tööd saavad kõrgema, millised madalama hinde. Selle tõttu võib olla

raske määrata lõplikuid hindeid töödele, mis asuvad kahe hinde vahel ja hinde osas ei valitse komisjonis üksmeel ning objektiivse otsuse tegemiseks oleks vaja ülevaadet kõikidest töödest, mis võimaldaks suhestada küsimuse all olevat tööd teistega.

1.1 Probleem

Praktikas ei ole hetkel kummalgi osapoolel võimalust teha valikut lõputöö teema osas selliselt, et neile oleks teada kogu informatsioon olukorrast. Probleemiks on inimeste erinev töötempo ja platvormi puudumine, kus saaks talletada kogu teadaolevat infot lõputöö teemade ja osapoolte eelistuste osas. Kuna inimesed töötavad erinevas, etteteadmatus tempos, ei ole hetkel võimalik olukord, kus kogu info on mõlemale osapoolele kättesaadav. Näide sellest on teema valiku puhul olukord, kus tudeng valib endale lõputöö teema valikute hulgast, mis on temale teadaolevad ja samal ajal juhendaja on nõus antud tudengit juhendama. Samas võib esineda olukord, et järgmisel päeval avaldab mõni õppejõud tudengi jaoks sobivama lõputöö teema. Samamoodi võib mõni teine tudeng avaldada soovi teha nüüd juba ära võetud teemat – tudeng kes teema juhendaja arvates sobiks paremini teemat tegema, kui hetkel valitud inimene. Taoliste olukordade vähendamiseks puudub infosüsteem ja protsessid, kus oleks võimalik tagada kõigile võrdväärne info kättesaadavaus, mille põhjal esitleda oma valikuid, et saavutada parim võimalik tulemus nii TTÜ tudengitele kui ka juhendavatele õppejõududele. Sellise infosüsteemi kasutamine säästaks mõlemale poolele aega, mis muidu kuluks liigsele individuaalsele suhtlusele ja teeks teemade jagamise läbipaistvamaks.

Hindamine nõuab süstemaatilist lähenemist hindamiskriteeriumite osas, et vältida hilisemaid pretensioone ja selleks, et iga hinne oleks võimalikult õiglane ning täpne. Hetkel puudub süsteem, mida saaks hindamiskomisjon kasutada abivahendina hindamisel, et teha informeeritud ja võimalikult õiglane otsus kõigi tudengite suhtes. Sellest tulenevalt on äriinformaatika õppekava juhil Gunnar Pihol soov uurida võimalusi sedasorti süsteemi juurutamiseks, mis annaks ülevaate ja oleks kasulikuks tööriistaks hindamisel.

1.2 Eesmärk

Töös uuritakse võimalusi, kuidas standardiseerida teemade ning tudengite kokku viimise protsessi ja kuidas saavutada mõlemale poolele õiglane tulemus. Lisaks otsitakse

lahendust probleemile, kuidas eristada suurel hulgal töid üksteisest, kui neil on sama numbriline hinne, võttes arvesse mitme hindaja antud hinnanguid ja erinevaid hindamiskriteeriume, et saavutada paremusjärjestus töödest, kus oleks ka sisuline erinevus sama hinde saanud lõputöödel ja info, millele saaks tugineda hinde põhjendamisel. Eesmärgiks on leitud tulemuste põhjal disainida ja rakendada selline infosüsteemi prototüüp, mida oleks võimalik edasi arendada ja kasutusele võtta Tallinna Tehnikaülikoolis ja mis lahendaks käsitletavad probleemid.

1.3 Töö struktuur

Teises peatükis antakse ülevaade metoodikast ja teemavaldkondadest, mida töös käsitletakse ning ülevaade kasutusele võetavatest tehnoloogiatest. Järgmises peatükis on välja toodud pakutavad lahendused töös käsitletavatele probleemidele algoritmide kujul ja on kirjeldatud rakendatava infosüsteemi prototüübi disain. Neljandas peatükis analüüsitakse leitud tulemusi, nende praktilisust ja võimalikke alternatiive.

2 Metoodika

Töös käsitletavatele probleemidele otsitakse lahendusi algoritmide kujul, mida oleks võimalik rakendada infosüsteemis. Kõnealune infosüsteem disainitakse töö jooksul ja tulemuste illustreerimiseks luuakse prototüüprakendus.

2.1 Uurimisobjektide ülevaade

Järgnevalt on toodud ülevaade töös käsitletavate probleemidega seotud uurimusobjektidest ja defineeritud nende omadused koos teoreetilise taustaga.

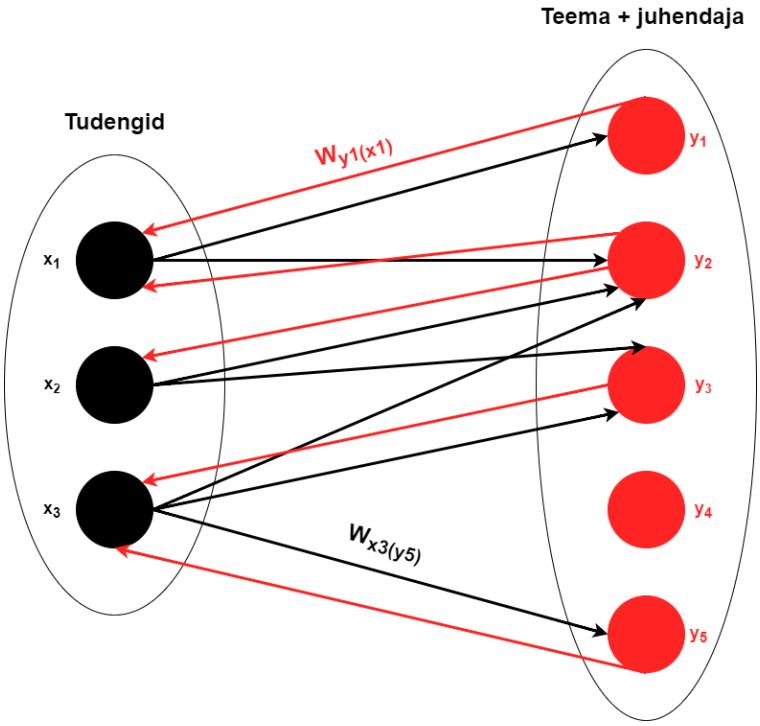
2.1.1 Lõputöö teemade automatiseeritud jaotamine

Selleks et leida lahendus osapoolte kokku viimise probleemile, toetutakse *social choice* teooriale ja graafiteooriale, uurides olemasolevaid algoritme ja teooriat graafides paaride leidmisteks, et leida sobiv moodus probleemi lahendamiseks Tallinna Tehnikaülikooli kontekstis, võttes arvesse praktikas ette tulevaid stsenaariume ja probleemist tulenevaid kitsendusi. Teooriaga tutvumiseks kasutasin David F. Manlove-i selle teemalist raamatut [2]. Probleemi saab vaadelda kui kahepoolset paaripaneku protsessi. Lihtsustatud mudel hõlmab kahte tüüpi süsteemis valikud tegevaid osapooli: tudengeid x ja teema-juhendaja paare y (Joonis 1). Praktikas esineva probleemi puhul saab üldistada, et teemade-juhendajate hulk on suurem või võrdne kui tudengite hulk. Mõlemaid osapooli iseloomustavad isiklikud eelistused (või nende puudumine) teise osapoole suhtes. Sealjuures on tulenevalt probleemi eripärast seatud piirang eelistuste osas teema-juhendaja osapoolele – eelistused saavad olla vaid nende tudengite suhtes, kelle eelistused sisaldavad antud teema-juhendaja paari. Selleks, et eristada tudengi eelistusi teemade suhtes on neile antud kaal $W_{xi}(y_j)$, mis iseloomustab eelistuse tugevust teise osapoole suhtes võrreldes teiste sama tudengi eelistustega. Sama moodi on antud kaalud $W_{yj}(x_i)$ ka igale teema-juhendaja eelistusele tudengi suhtes. Leitava tudengi ja teema-juhendaja paari kaaluks loeme:

$$W_{xy} = W_{x_i}(y_j) + W_{y_j}(x_i)$$

Eesmärgiks on leida lahendus, kus:

- On leitud võimalikult paljudele tudengitele x teema + juhendaja y
- Kõikide tudengite x ja kõikide teema-juhendajate y rahulolu on maksimaalne.



Joonis 1. Tudengite ja teema-juhendaja eelistuste mudel. Tudengid x1 – x3 ja juhendajad koos teemaga y1-y5. Wy1(x1) näitab juhendaja y1 eelistuse tugevust tudengi x1 kohta, Wx3(y5) näitab tudengi x3 eelistuse tugevust teema + juhendaja y5 suhtes.

Tudengi ja teemade kõige optimaalsema lahenduse leidmiseks on vaja moodust hindamiseks, kui hea või halb on iga võimalik paari kombinatsioon ja üldisemalt, kui hea on kogu lahendus kõikide tudengite ja juhendajate seisukohalt. Probleemi mudelis (Joonis 1) on välja toodud kahepoolse graafi näol graafi tipud, milleks on osapooled ja servad, mis on eelistused osapoolte vahel. Irving pakkus välja tulemuse hindamiseks graafi servadele arvulise väärtuse andmise – hinne mis iseloomustab kasutaja eelistuse tugevust võrreldes teiste sama kasutaja eelistustega [3]. Kuna paari moodustamiseks peavad mõlemad pooled omama eelistust teise osapoole suhtes, siis saab moodustatud paari hinde moodustada mõlema individuaalse hinde summana. Mida kõrgemad on individuaalsed hinded, seda kõrgem on nii individuaalne rahulolu kui ka rahulolu tervikuna. Üle kõikide leitud paaride ühes lahendis saab leida keskmise rahulolu individuaalse osapoole suhtes kui ka leitud paaride keskmise rahulolu.

Rahulolu leidmisel üle kõikide tudengite ja juhendajate tuleb arvestada probleemi eripära, mis seab piiranguid selles osas, kuidas hinnata rahulolu üldisemas pildis. Tudengite jaoks on esmatähtis teema leidmine, alles siis järgneb leitud teema rahulolu hindamine võrreldes teiste eelistustega. Seega võib lahenduse leidmisel lugeda kõikide tudengite, kes ei saa üldse teemat, rahulolu lõpmata väikeseks. Juhendajate seisukohalt ei oma see tähtsust, kas paaride leidmise tulemusena välja pakutud teemad leiavad tudengi või mitte. Sellest tulenevalt saab teha üldistuse, et otsime graafi maksimaalselt katvat lahendit, kus on võimalikult suur hulk tudengite ja teema-juhendaja paare. Alles pärast seda saab arvesse võtta keskmist heaolu üle leitud paaride.

Probleemi lahendamisel on tehtud eeldused, et süsteemis valikuid tegevad kasutajad esitavad süsteemile tõepäraseid eelistusi ja seega aktsepteerivad algoritmi tagastatud tulemust neile pakutud teema või tudengi osas. Samuti eeldame, et õppejõudude lisatud teemad, mida nad on nõus juhendama, lähevad kõik juhendamisele, kui nendele on kandideerinud vähemalt üks tudeng. See tähendab, et õppejõul ei ole võimalik keelduda juhendamast tema välja pakutud teemat ja lisatud on vaid selles koguses teemasid, mida ta on suuteline samaaegselt juhendama.

2.1.2 Lõputööde õiglane hindamine ja paremusjärjestuse moodustamine

Uuritakse erinevaid viise, kuidas moodustada retsensentide hinnangute põhjal terviklikku paremusjärjestust, mis oleks objektiivne ja annaks tagasisidet nii tudengitele kui ka õppejõududele tehtud töö kohta. Paremusjärjestuse moodustamisel on subjektiks *n* valminud lõputööd, mida hindab *k* retsensenti. Iga tööd hindab rohkem kui üks retsensent, et hinne ei kujuneks vaid ühe indiviidi arvamuse põhjal. Probleemile lahendust otsides eeldame, et $n \gg k$ ja hindamine toimub skaalal 0-5. Eesmärgiks on reastada *n* lõputööd selliselt, et järjestus esitaks objektiivselt tööde taset üksteise suhtes ning esindaks individuaalsete hinnete puhul mitte ainult retsensendi, vaid teaduskonna, kui tervikliku kollektiivi arvamust sellest tööst. Lõputööde hindamiseks ja paremusjärjestuse moodustamiseks uuritakse erinevaid hääletamisalgoritme ning tehisnärvivõrke, et leida erinevaid alternatiive, kuidas hinnata ja järjestada lõputöid osaliste hääletustulemuste põhjal. Erinevate lähenemiste jaoks pakutakse välja mooduseid, kuidas antud algoritmi disainitavas süsteemis rakendada.

Hääletusalgoritme kasutatakse poliitikas ja teistes valdkondades, kus hulk hääletajaid valivad kandidaatide hulgast võitja või võitjad, mis rahuldab kindlaid kriteeriume

sõltuvalt algoritmist. Sellistel algoritmidel võib olla potentsiaali ka lõputööde paremusjärjestuse moodustamisel. Kuigi tehniliselt on ka selles paremusjärjestuses võitja, kõige parem lõputöö, siis oleme huvitatud leidmast lahendust, mis ei pakuks ainuüksi võitjat vaid kogu järjestust parimast tööst halvimani. Retsensendid saavad hinnata vaid realistlikus koguses töid, see tähendab, et rakendatavad on ainult hääletusalgoritmid, mis tulevad toime osaliste hääletustulemustega. Hääletusmeetodite võrdluses on ka tähtsal kohal strateegiakindlus, mis takistaks hääletajaid strateegiliselt hääletamast. Töös käsitletavas probleemis eeldame, et retsensendid ja teised hindajad esitavad siiralt ja professionaalselt oma arvamust hinnatavate tööde kohta ega oma mingit tagamõtet ning seega strateegilist hääletamist ei käsitleta. Hääletamisalgoritmidest ülevaate saamiseks kasutasin Vikipeedia ülevaadet ja võrdlust erinevatest hääletamisalgoritmidest [4].

Arvutisüsteemid keerukamate probleemide lahendamiseks, mis on inspireeritud bioloogilistest närvivõrkudest nagu inimese aju, on eksisteerinud juba mõnda aega ja leidnud praktikas kasutust erinevates valdkondades ning nende populaarsus on kasvamas. Üheks kasutusalaks on ka mustrite tuvastamine ja sisendi põhjal klassifitseerimine, mis sisult sarnaneb ka selles töös uuritava lõputööde hindamise probleemiga, kus sisendiks on erinevad hindamiskriteeriumid, mille põhjal määratakse töö klass ehk hinne [5].

2.1.3 Lõputöö teemade haldamise rakendus

Hetkel puudub tehnoloogiline vahend, kuidas automatiseerida Tallinna Tehnikaülikoolis Infotehnoloogia teaduskonnas protsessi, mille käigus tudengid leiaksid endale meelepärase lõputöö teema ja juhendaja ning õppejõud leiaksid ka endale sobiva tudengi. Samuti ei eksisteeri ühist lahendust välja pakutud teemade hoiustamiseks ning töötlemiseks, mille põhjal saaksid tudengid ja õppejõud otsuseid teha. Infosüsteem, mis annab ülevaate lõputöö teemadest teeb lihtsamaks meelepäraste teemade leidmise ning valiku tegemise nende seast ja lihtsustab uute teemade lisamist, säästes aega kõigi osapoolte jaoks. Lisaks võimaldab see jälgida kogu lõputöö protsessi ja süsteemis olevaid valmis töid hinnata erinevate kriteeriumite alusel, mis oleks väärtuslik informatsioon kaitsmiskomisjonile lõpliku hinde määramisel.

2.2 Kasutatavad tööriistad

Algoritmid implementeeritakse Java programmeerimiskeeles loodud rakenduses, mille struktuur põhineb Spring Boot raamistikul ning graafidega töötamiseks on kasutusel

JGraphT teek, mis sisaldab nii andmestruktuure kui ka algoritme levinud graafitüüpide kohta [6]. Statistika koostamisel kaasati Apache Commons CSV teek CSV failidega töötamiseks [7]. Andmete salvestamine peasüsteemis toimub läbi Object Relational Mappingu (ORM) vastavalt JPA spetsifikatsioonile, kasutades Spring raamistikuga kaasas olevat Hibernate teeki. Andmed salvestatakse relatsioonilises andmebaasis. Spring Boot raamistik teeb uue rakenduse loomise kiireks, omades suurel hulgas baasseadistusi, mida vastavalt vajadusele saab muuta. Prototüüprakenduse arendati kasutades H2 muutmälu sisest andmebaasi, kuid on disainitud töötama ka traditsioonilisel SQL andmebaasil nagu näiteks PostgreSQL. Nende raamistike kombineerimisel on võimalik arendada Java serverirakendus, mida on lihtne käivitada ja vastavalt vajadusele muuta nii andmebaasi kui ka JPA implememtentatsiooni. Süsteemi disaini dokumenteerimiseks kasutati UML modelleerimiskeelt. Algoritmide valideerimiseks kasutati Javale omast JUnit raamistikku ühiktestimiseks.

Juhuslikkusel põhinevaid testandmeid salvestatakse JSON struktuuriga tekstifailidesse hilisemaks kasutamiseks algoritmide võrdlemisel ja valideerimisel, selleks et sisendandmed oleksid kõikidel testimistel samad. Tulemuste valideerimiseks kasutatakse juhuslikkusel põhinevaid testandmeid, mis on sisendiks ühiktestidele. Eesmärgiks on katta võimalikult suur hulk stsenaariume, selleks et hinnata tehtud töö tulemusi võimalikult täpselt, mida manuaalsel testimisel ja üksikute testjuhtudega ei ole võimalik saavutada praktikas esinevate sisendandmete suuruse juures. Arendamise juures kasutati ka väiksemaid manuaalselt loodud testjuhte algoritmide paremaks mõistmiseks ja vigade silumiseks.

Valisin Java programmeerimiskeele ja Spring Boot raamistiku seoses oma varasema töökogemusega ning asjaolul, et nende puhul on tegemist väga laialdaselt levinud programmeerimiskeelega ja populaarse raamistikuga, mis lihtsustab töö käigus paratamatult tekkivatele tehnilistele probleemidele lahenduste leidmist ning tänu laiale levikule on kergem leida arendajaid, kes saavad vajadusel programmi edasi arendada või teha veaparandusi [8]. Versioonihaldus toimub läbi Git programmi. Teised teegid on valitud sõltuvalt vajalikust funktsionaalsusest ja populaarsusest otsingumootoris.

2.3 Tööprotsess

Esimese sammuna kaardistati süsteemi peamised kasutusjuhud ja programmeeriti süsteemi üldine struktuur kasutades Spring Boot raamistikku, mis sisaldas algelist andmemudelit ja CRUD operatsioone läbi HTTP päringute. Järgnevalt uuriti erinevaid algoritme, mis lahendaks töös käsitletavaid probleeme, millest sobivaid algoritme programmeeriti süsteemis. Paralleelselt algoritmide uurimisega täiendati andmemudelit, sest töö käigus sai paremini selgeks, milline on täpsem andmete struktuur ja milliseid andmeid vajab iga algoritm. Valminud algoritmilisi lahendusi testiti läbi ühiktestide ja lisati testimist toetavat funktsionaalsust, mis võimaldab paremini tulemusi analüüsida.

3 Peamised tulemused

Järgmistes alampunktides on toodud välja peamised tulemused, milleks on infosüsteem koos algoritmidega tudengite ja teemade paaride leidmiseks ning tööde järjestuse moodustamiseks.

3.1 Süsteemi kasutusjuhud ja disain

Süsteemis on peamisteks rollideks tudengid ja ülikooli töötajad. Prototüüprakenduses on üldistatud ülikooli töötajad üheks rolliks, mis sisaldab juhendajaid, administratiivseid töötajaid ja retsensente koos kaitsmiskomisjoni liikmetega, et töö fookusesse jääks teemade haldamisega seonduv süsteemi disain (Lisa 1 – Infosüsteemi peamised kasutusjuhud).

Üldised kasutusjuhud, mis ei sõltu rollist:

- Lõputööde otsimine ja filtreerimine
- Lõputöö teema detailide kuvamine

Tudengitel lisanduvad järgmised kasutusjuhud:

- Lõputöö teemale kandideerimine
- Süsteemi pakutud teema aktsepteerimine / tagasi lükkamine
- Uue teema lisamine koos soovitud juhendajaga

Ülikooli töötaja rollis kasutajatel on lisaks valikute tegemisele ka teemade haldusega ning valmis tööde hindamisega seotud kasutusjuhud:

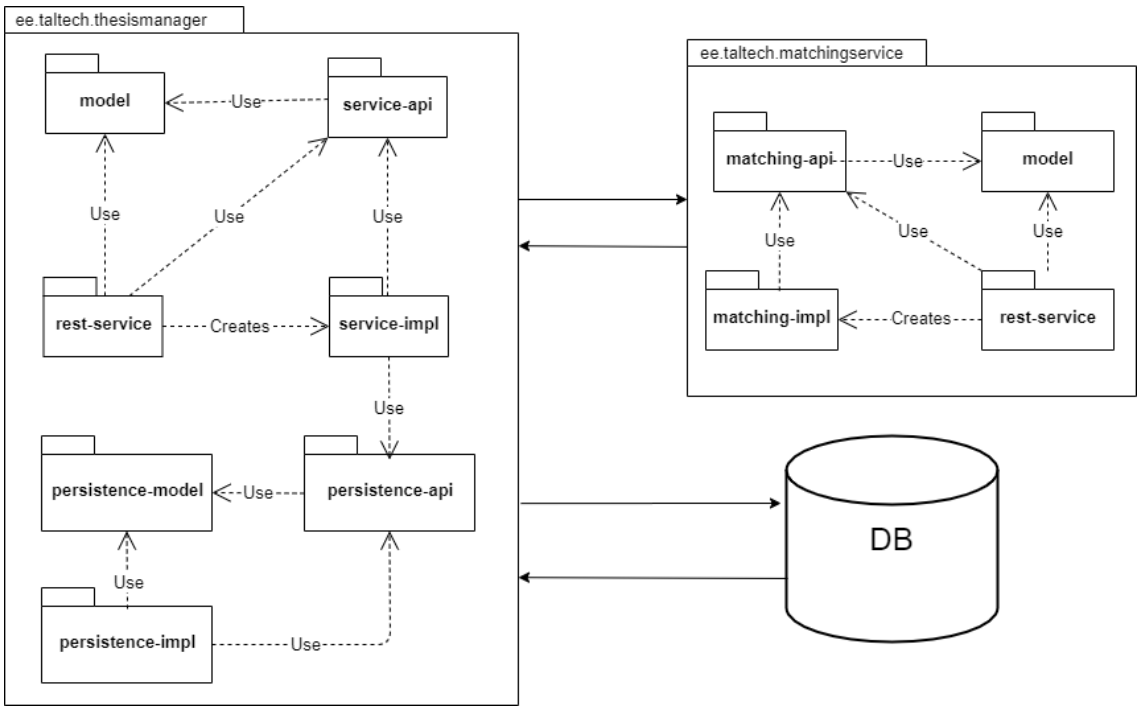
- Kandideerinud tudengite hulgast eelistuste valimine
- Uue teema lisamine

- Teemade redigeerimine ja kustutamine
- Tööde hindamine ja paremusjärjestuse genereerimine

Lisaks funktsionaalsetele nõuetele valisin ka järgnevad mittefunktsionaalsed nõuded mida prototüübis järgida:

- Auditeeritavus, et tähtsamad tegevused oleksid logitud ja tagantjärgi tuvastatavad.
- Modulaarsus selleks, et koodibaas oleks kergemini hallatav ja süsteem oleks paremini skaleeritav.
- Konfigureeritavus süsteemi sätete muutmiseks vastavalt vajadusele.

Valminud prototüüp koosneb komponentidest, mis suhtlevad läbi HTTP päringute ja andmebaasiga läbi JDBC draiveri. Komponendid on jagatud Maven mooduliteks, mis järgib kihilise arhitektuuri põhimõtet (Joonis 2).



Joonis 2. Infosüsteemi teemade haldamise komponendi disain ja seos teiste süsteemi komponentidega.

Kõikide paaripaneku algoritmide programmeerimiseks, analüüsiks, testimiseks ja teiste süsteemi osadega liidestamiseks valmis eraldiseisev komponent, mis keskendub vaid paaride leidmisega seonduvatele ülesannetele. See algoritmide komponent ei vaja andmete salvestamist ja on eraldiseisev peamisest süsteemist. Selleks implementeeriti

Java rakendus, mida on võimalik serverisse paigaldada ülejäänud lõputööga seotud infosüsteemist eraldiseisva rakendusena ja kommunikatsioon toimub läbi HTTP päringute. Rakenduses kasutatakse erinevaid kahepoolsete graafidega seotud algoritme. Programmis kasutatakse algoritmide ise loodud implementatsioone ja teeki JGraphT, mis sisaldab graafidele omaseid andmestruktuure ja levinud algoritme [6]. Lisaks defineeriti ise andmetüüp graafi tippude jaoks.

Paaride leidmise komponent on jaotatud Maveni moodulitesse (Joonis 2):

- model – andmete mudel algoritmide sisendiks ja väljundiks ning REST liideste mudel
- matching-api – üldine liides implementeeritud algoritmidele ja neid kasutavatele teenustele
- matching-impl – teenuste ja algoritmide implementatsioon
- rest-service – REST kontrollerite moodul teenuste välja kutsumiseks ja andmete teisenduseks

Antud komponent keskendub algoritmide implementatsioonidele ning andmete salvestamist selles rakenduses ei toimu, seda tehakse peamises infosüsteemis, kus on täielikum info sisendandmete kohta. Nii sisendandmed HTTP päringus kui ka vastuses on kodeeritud JSON formaati. Antud rakendus nõuab vaid minimaalset infot töötamiseks, milleks on unikaalsed identifikaatorid, hulkadesse jaotamine vastavalt rollile ja eelistuste järjestused teise hulga suhtes (Joonis 3).

```
{
  "students": [{
      "id": "student 1",
      "orderedPreferences": [
        "topicA",
        "topicB",
        "topicC",
        "topicD"
      ]
    }, ... ],
  "topics": [{
      "id": "topicA",
      "orderedPreferences": [
        "student 4",
        "student 3",
        "student 1",
        "student 2"
      ]
    }, ... ],
  "teachers": [{
      "id": "teacher1",
      "topics": ["topicA"]
    },{
      "id": "teacher2",
      "topics": ["topicB"]
    },{
      "id": "teacher3",
      "topics": ["topicC", "topicD"]
    }]
}
```

Joonis 3. JSON sisend paaride leidmise komponendile

Vastuses tagastatakse lisaks leitud paaridele ka statistika tulemuste kohta: protsentuaalsed keskmised tulemused nii tudengite kui ka õppejõudude kohta, tulemuse stabiilsus tõeväärtusena. Samuti loetletakse ka kõik teemad ja tudengid, keda ei õnnestunud paaridesse jaotada (Joonis 4).

```
{
  "resultType": "COMPLETE_MATCHES",
  "matches": [
    {
      "student": "student 3",
      "topic": "topicA",
      "teacher": "teacher1",
      "studentWeight": "3.0000",
      "topicWeight": "3.0000"
    },{
      "student": "student 4",
      "topic": "topicB",
      "teacher": "teacher2",
      "studentWeight": "3.0000",
      "topicWeight": "3.0000"
    },{
      "student": "student 1",
      "topic": "topicC",
      "teacher": "teacher3",
      "studentWeight": "2.0000",
      "topicWeight": "3.0000"
    },{
      "student": "student 2",
      "topic": "topicD",
      "teacher": "teacher3",
      "studentWeight": "3.0000",
      "topicWeight": "3.0000"
    }
  ],
  "unmatchedStudents": [],
  "unmatchedTopics": [],
  "studentAverageSatisfaction": "68.7500",
  "teacherAverageSatisfaction": "75.0000",
  "stableMatchingResult": true
}
```


Joonis 4. JSON väljund leitud paaridega ja statistikaga

Iga algoritmi jaoks on loodud nii üldised teststsenaariumid kui ka spetsiifilised testid. Gale-Shapely algoritmi puhul testiti täieliku ja osalise lahendi leidmist väikeste ning suurte sisendandmetega. Testide läbimise kriteeriumiks oli nõue, et kõik leitud lahendid oleksid stabiilsed. Väiksematel käsitsi loodud testandmetel põhinevatel testidel kontrolliti lisaks stabiilsusele ka tingimust, et kõik paarid on leitud, kui see oli kontrollitud käsitsi ja eksisteeris. Suurematel andmetel, kus ei ole ajaliselt mõistlik käsitsi kontrolli teostada, kontrolliti vaid tulemuse stabiilsust, sest tõenäoliselt on tulemuseks osaline lahend, kuid mis peab olema stabiilne.

Maksimaalse katvusega algoritmi testimisel kasutati lisaks suurtele genereeritud testandmetele ka stabiilse algoritmi testandmeid jättes välja stabiilsuse kriteeriumi. Mõlema algoritmi puhul kontrolliti, et moodustatud paarid oleks korrektsed, ehk mõlemad osapooled omaksid tulemuses eelistust teise suhtes.

Algoritmi korrektsuse kontrolliks piisas väiksemate sisendandmetega testidest, kus saab arendaja tulemuste korrektsust käsitsi lahendades hinnata. Tüüpiline sisendandmete suurus oli sellisel juhul 5 tudengit, 1 kuni 5 õppejõudu, 3 kuni 5 eelistust teise osapoole suhtes. Need testandmed võimaldavad käsitsi korrektsuse kontrollimist ja lihtsustasid programmikoodi silumist (*debug*) implementatsioonifaasis. Praktikas võib süsteemi sisendiks olla aga suur hulk tudengeid ning õppejõude ja veel suurem hulk teemasid. Samuti oli vaja hulgaliselt testandmeid saavutatud tulemuste põhjalikumaks analüüsimiseks. Kuna sellist süsteemi ei ole veel rakendatud TTÜ-s, siis puuduvad ka andmed varasemate tudengite eelistuste kohta ning sedasorti andmete käsitsi loomine ja testimine ei ole mõistlik.

Probleemi lahendamiseks on loodud samasse rakendusse eraldi abifunktsionaalsus, mis tegeleb testandmete genereerimisega ja tulemuste raporteerimisega. Peamiste funktsioonide hulka kuuluvad:

1. Spetsiifilise väikse testjuhtumi hoidmine koodis, mida saavad kõik testid kasutada, et vältida samade testjuhtude duplitseerimist iga algoritmi jaoks ja vähendada viga tekkimise ohtu.

2. Vastavalt parameetritele loodud juhuslikkusel põhinevate testjuhtude loomine

3. Testjuhtude hoiustamine JSON failides ja failidest testjuhtude lugemine

4. Tulemuste raporteerimine CSV formaadis, statistika koostamine

Testjuhtumid on genereeritud Java juhuslikkuse mooduli alusel, mis on sobilik juhuslike valikute simuleerimiseks. Loodi eraldi funktsionaalsus testjuhtude salvestamiseks JSON failidesse. Selle tulemusena saab genereeritud andmeid taaskasutada säästes aega ja tagades järjepidevuse algoritmide võrdluses. Testjuhtude käivitamisel kontrollitakse esmalt faili olemasolu – sel juhul kasutatakse andmeid JSON failist, vastasel juhul genereeritakse uus fail ja kasutatakse neid andmeid edaspidi. Arendajal on võimalus ka tahtlikult sundida süsteemi uut faili genereerima, kui selleks on soov. JSON failid on

nimetatud vastavalt sisendparameetritele, mille alusel testjuht genereeriti. Põhjalikumaks analüüsiks oli tarvis samade sisendparameetritega mitmeid testjuhte, et saada täpsemat tulemust. Selleks on lisatud faili nimesse ka iteratsiooni number, et samade parameetritega testjuhtudel vahet teha (Joonis 5).

Joonis 5. JSON failide nimede ülesehitus

3.2 Algoritmiline lahendus paaride leidmiseks tudengite ja lõputöö teemade vahel

Töö tulemusena võeti kasutusele kaks eri eesmärki optimeerivat algoritmi ja kombineeriti need teenuskihis töö kontekstis parima tulemuse saavutamiseks.

3.2.1 Stabiilse paaripaneku algoritm

Olgu kahepoolne graaf G = {X, Y, E}, kus X on tudengite hulk, Y teemade hulk ja E servade hulk, mis iseloomustavad vastastikku eelistust X ja Y vahel. Paaripaneku lahendiks kahepoolses graafis loetakse selliseid servade E alamhulki, kus iga tipp esineb hulgas maksimaalselt üks kord. Stabiilsest lahendit on esmalt maininud David Gale ja Lloyd Shapely [9], kes leidsid, et igas kahepoolses graafis, kus on täielikult defineeritud mõlema poole eelistused, eksisteerib lahendite hulgas ka stabiilne lahend. Sõna stabiilsus siin kontekstis tähendab servade hulga omadust, kus ei eksisteeri $x \in X$ ja $y \in Y$ selliselt, et nii x kui ka y eelistavad vastastikku üksteist oma praegustele paarilistele lahendis [9]. Kui selline paar eksisteerib (*blocking pair*), siis see võimaldab nendel osapooltel süsteemi väliselt privaatne kokkulepe saavutada ja lõhkuda lahendi korrektsust, sest kuigi x ja y said parema tulemuse, siis nende vanad partnerid (x’ ja y’ vastavalt) ei pruugi omada üldse eelistusi üksteise vastu ja selle tulemusena jäävad need kaks graafi tippu paari panemata või saavad halvema kaalu uue paaris. Seega saab ka lahendi stabiilsust arvestada ka ühte tulemuse kvaliteedi näitajat. Stabiilne lahend säilitab oma stabiilsuse ka siis, kui üks või enam leitud paaridest keeldub saavutatud tulemusest ja selle tulemusena on süsteemist välja arvatud [10].

Antud töö kontekstis vajas Gale-Shapely algoritm kohandamist, mis oli esialgselt disainitud vaid sellistele paaripaneku ülesannetele, kus mõlemal osapoolel olid täielikud eelistuste järjestused teise osapoole suhtes, mis tagas selle, et iga osapool leidis endale paarilise algoritmi töö lõpuks. Originaalne implementatsioon põhjustas aga algoritmis lõpmatu töötamise, kui eelistused olid osalised ja nende abil ei olnud võimalik leida kõiki paare. See on reaalne situatsioon disainitavas süsteemis, juhul kui suur osa tudengitest valib üle oma kõikide valikute väikese alamhulga kõikidest lõputöödest – valitud töid ei jagu kõikidele tudengitele piisavalt ja mõnele teemale ei kandideeri keegi. Sellest võimalikust situatsioonist kujunes välja ka uus näitaja, mille järgi erinevaid paaripaneku algoritme konkreetse süsteemi raames võrdlesin.

3.2.2 Maksimaalse katvusega paaride leidmine

Selle paaripaneku implementatsiooni eesmärgiks on leida lahend, kus on maksimaalne arv tudengeid ning kogu lahendi hinne oleks maksimaalne. Selleks leitakse esmalt kahepoolsest graafist lahend, kus on suurim arv tudengite/teemade paare kasutades Hopcroft-Karp algoritmi [11]. Saadud tulemuses olnud graafi tippudest ja nende eelistustest moodustatakse uus alamgraaf, millel on omadused, et seal esineb täielik lahend ja mõlemas graafi pooles on võrdne arv tippe. Järgmisena rakendatakse algoritmi, mis leiab maksimaalse kaaluga täieliku lahendi uues graafis, mida süsteem tagastab tulemusena. Selleks saab kasutada Blossom V algoritmi – Jack Edmonds-i originaalse Blossom algoritmi edasiarendust Vladimir Kolmogorov-i poolt [12]. Kuigi algoritmi eesmärgiks on leida minimaalse kaaluga lahend, siis saame meie probleemi teisendada vastavale kujule korrutades kõik kaalud -1-ga. Algoritm on võimeline leidma maksimaalse katvusega lahendi ja selles osalevate tippude põhjal leidma maksimaalse tulemuse mis on alamgraaf algsest probleemist.

3.2.3 Algoritmide kombineerimine

Mõlemad algoritmid on rakendatud paaride leidmise programmi teenusekihis paralleelselt. Algoritmid programmeeriti sama liidese järgi ning seega on sisend- ja väljundandmed samad. Sisendprobleemi korral tehakse andmetest koopia ja käivitatakse mõlemad algoritmid paralleelselt. Seejärel kogutakse tulemused ja nende statistika ning tagastatakse üks tulem kahest. Tagastatava vastuse valikul eelistatakse esimesena seda tulemust, kus on rohkematel tudengitele teema jagatud. Juhul kui mõlemas tulemuses on võrdne arv leitud paare, siis tagastatakse vastavalt sisendiks antud kriteeriumile ühe

algoritmi vastus. Vastavalt kriteeriumile võrreldakse mõlema algoritmi tagastatud metaandmeid (statistikat). Näiteks saab selle põhjal valida tulemuse, kus keskmine tudengi või õppejõu kaal on kõrgem või kus on tulemustes ka stabiilsuse kriteerium täidetud.

3.3 Hinnatud tööde paremusjärjestuse moodustamine

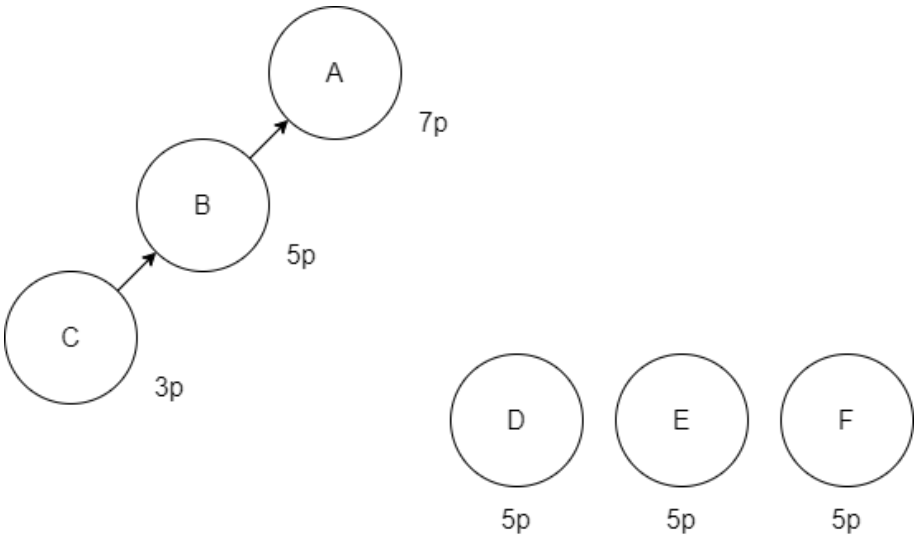
Borda count on hääletusmeetod, kus hääletaja peab reastama kandidaadid eelistuste järgi. Selle alusel saab iga kandidaat kindla arvu punkte vastavalt sellele, kui kõrgel eelistustes ta asub. Kandidaadi punktid summeeritakse üle kõikide hääletajate ning suurima punktiarvuga kandidaat on võitja. See meetod leiab arvutamise käigus kõikide kandidaatide punktid, mis võimaldab ka sorteerida need kandidaadid, moodustades paremusjärjestuse. Modifitseeritud Borda count võimaldab agregeerida hääletustulemusi paremusjärjestuseks ka osaliste eelistuste alusel, olenemata sellest, kas neil on ühisosa või mitte [13]. Lõputööde hindamisel võimaldab see jagada tööd retsensentide vahel ja nende hääletustulemused agregeerida paremusjärjestuseks selliselt, et alamhulkades ei pea tingimata kattuvusi olema. Kuigi selle tulemusena saame paremusjärjestuse, ei anna see hääletustulemus hindelist väärtust töödele, sest toimib lineaarsel skaalal ühe alamhulga mõistes. See tähendab et retsensent määrab ainult talle antud tööde hulgast paremusjärjestuse, kuid ta ei saa märkida kui palju need tööd omavahel erinevad, sest see on algoritmi poolt määratud, et alamjärjestuses kõrvuti asetsevad tööd erinevad täpselt 1 hinde võrra. Kuigi retsensent võib võtta arvesse erinevaid kriteeriume töö hindamisel, siis need ei salvestu lõplikes tulemustes. Selle meetodi puhul on võimalikud ka stsenaariumid, kus paremusjärjestuses võib esineda üks või rohkem viigiseisu, mis vajavad eraldi meetodeid viikide eemaldamiseks. Viikide eemaldamiseks on välja pakutud mitmeid meetodeid, mida saab rakendada Borda count tulemustel [14] [15]:

- Paarikaupa võrdlused viigis olevate kandidaatide vahel
- Kõrgeimate hääletustulemuste järgi võrdlemine
- Madalaimate hääletustulemuste järgi võrdlemine

Valminud rakenduses kasutatakse kombinatsiooni nendest viikide murdmise lahendusest ja keskmise hinde põhjal eristamisest. See tähendab, et lisaks individuaalsetele

paremusjärjestustele peab retsensent andma ka hinde tööle. Hinnet ei kasutata hääletusalgoritmi sees välja arvatud juhul, kui on vaja murda viike.

Algoritmi sisendiks on kõik retsensentide osalised paremusjärjestused koos hinnetega. Sellele järgneb iga alamjärjestuse kontekstis kaalude arvutamine, kus järjestamata tööd saavad sama arvu alustuspunkte sõltuvalt kõikide tööde hulgast. Järjestusse pandud töödele pannakse samuti sama palju alustuspunkte, kuid sellele järgneb punktide ümberjagamise protsess, kus iga hinnatud töö loovutab ühe oma punktidest igale temast järjestuses kõrgemal olevale tööle. Tulemuseks on hulk sama kaaluga töid, mida ei järjestatud ja teine hulk töid, mis nüüd on erinevate kaaludega (Joonis 6). Kõik punktid liidetakse globaalsesse tabelisse kokku ja pärast kõiki alamjärjestusi saab globaalsete kaalude järgi tööd järjestada lõplikus paremusjärjestuses [13].



Joonis 6. Punktide jagamise süsteem osalistel järjestustel. A, B, C esindavad hinnatud töid ja D, E, F on järjestamata tööd antud alamjärjestuses.

4 Tulemuste analüüs

Järgnevalt on välja toodud võrdlus varasemalt disainitud süsteemidega ja programmeeritud süsteemi analüüs koos edasiste sammudega. Algoritme võrreldakse erinevate kriteeriumite alusel ja tuuakse välja teisi algoritme, mida saab edasi uurida samas teemavaldkonnas tulemuste võimalikuks parandamiseks.

4.1 Rakendus

Lõputöödega seonduvate protsesside digitaliseerimine ja koondamine ühte infosüsteemi on uuritud ja leidnud rakendust ka teistes ülikoolides, kus on vastavalt institutsiooni nõuetele disainitud ja loodud süsteemid, mis toetaks erinevaid toiminguid lõputöödega. Nende hulgas on traditsiooniline teemade haldus ehk CRUD operatsioonid, arhiveerimine, esitatud tööde vastu võtmine, kaitsmiskavade koostamine, suhtlus osapoolte vahel ja ka hindamine. Sellised veebipõhised süsteemid on ka praktikas juba kasutust leidnud erinevates teadusasutustes [16] [17] [18]. Iga süsteem on disainitud vastavalt konkreetse ülikooli vajadustele ja seega ei ole universaalset lahendust, mida saaks kasutusele võtta kõikides ülikoolides. Nendest süsteemidest sarnaneb selles töös disainitavale infosüsteemile kõige rohkem Lodz-i Tehnikaülikooli jaoks loodud süsteem, kus on kasutusel sarnased tehnoloogiad (Java, ORM, SQL) ja funktsionaalsus keskendub teemade haldamisele ning teemadele kandideerimisele [19]. Erinevalt sellest disainitud süsteemist, mis on mõeldud töötama omaette rakendusena, on Tallinna Tehnikaülikoolis mitmeid süsteeme, mille teenuseid saab kasutada selles töös disainitud rakenduses ja seega ei vajanud uuesti implementeerimist. Nende süsteemide hulka kuuluvad:

- ÕIS ehk Tallinna Tehnikaülikooli õppeinfosüsteem tudengite ja lõputöödega seonduva info pärimiseks, hinnete haldamiseks.

- Õpikeskkond ained.ttu.ee, mida saab kasutada kasutajaliidesena disainitava süsteemi jaoks ja mis sisaldab juba foorumi funktsionaalsust ning lisafunktsionaalsust, mida uuesti implementeerima ei pea.

- Tallinna Tehnikaülikooli raamatukogu digikogu, kus on talletatud juba tehtud lõputööd ja metaandmed.

Sellest tulenevalt on disainitaval süsteemil peamisteks ülesanneteks uute lõputöödega seonduvate andmete ja protsesside haldus ja liidestamine selles töös uuritud algoritmidega. Integratsioon ülejäänud Tallinna Tehnikaülikooli süsteemidega on edasiseks arenduseks ja ei olnud vajalik selles töös käsitletud probleemide lahendamiseks.

Ühe monoliitse rakenduse asemel eelistati eraldi töötavaid komponente peamiselt paindlikkuse huvides. Selle plussiks on võimalus paaride leidmise komponenti eraldi installeerida teise süsteemi ja viia sisse muudatusi põhisüsteemi tööd häirimata. Prototüübi edasi arendamise puhul on võimalik, et edasi arendajad leiavad, et on vajalik programmeerida uus lahendus mõnes muus programmeerimiskeeles - sellisel juhul saab teises programmeerimiskeeles programmeerida ainult selle algoritmide komponendi. Lisaks on võimalus laiendada ja kasutada seda komponenti ka väljaspool lõputööde konteksti, et kasutada vastastikuste eelistuste põhjal lahenduste otsimist näiteks erinevate rühmaprojektide jaoks, kus on vaja moodustada tudengite grupid ja teised sarnased inimeste automatiseeritud jaotamiste ülesanded. Samuti on algoritmid programmeeritud selliselt, et vajavad sisendandmeteks vaid unikaalseid identifitseerivaid tunnuseid ja eelistusi, seega ei ole vaja sisendiks spetsiifilisi andmeid nagu lõputöö pealkiri, tudengite ja töötajate nimed - need võivad olla asendatud tehniliste identifikaatorite või anonümiseeritud andmetega, mis võivad välises süsteemis viidata ükskõik millisele objektile. Antud komponent saab olla põhjaks tulevastele samateemalistele algoritmidele, sest on defineeritud üldised liidesed, mida kasutades tagatakse ühilduvus ülejäänud süsteemiga. Samuti loodi ühtne sedasorti algoritmidele mõeldud testimisraamistik, mis sisaldab testandmete genereerimist, nende salvestamist ja lugemist ning neid andmeid kasutavaid ühikteste. Kasutades sama struktuuri kõikide algoritmide jaoks tagab nende objektiivse võrreldavuse, sest saab analüüsida samu parameetreid.

Genereeritud testandmed salvestati JSON failidesse selleks, et need oleksid taaskasutatavad mitme algoritmi poolt, sest vastasel juhul oleks juhuslikkusel genereeritud andmed iga kord erinevad ja selle tõttu ei saaks võrrelda algoritme objektiivselt. Samuti on genereerimine aeganõudev protsess, mis suurendab testimisele kuluvat aega mis omakorda suurendab kompileerimisele kuluvat aega.

Valminud prototüüp on mõeldud kasutamiseks Docker konteineris kergeks installeerimiseks serverisse, et tagada identne rakenduse töötamise keskkond sõltumata välisest keskkonnast, olgu selleks siis arendaja arvuti või erineval operatsioonisüsteemil jooksev server. See võimaldab ka valmislahenduses teha muudatusi, hoides teisi süsteemi komponente töös. Komponentideks jagamine teeb võimalikuks ka rakenduse komponentide käivitamise erinevates masinates. Sellest tulenevalt on projekti Git-i repositooriumisse lisatud konfigureeritud Dockerfile.

Järgmiseks sammuks rakenduse kasutusele võtul on selle ühendamine kasutajaliidesega ja Dockeri konteinerite installeerimine serverikeskkonda. Prototüübis ei ole arvesse võetud rollipõhist ligipääsu erinevatele REST liidese osadele, mida on samuti vaja lisada edasises arenduses. Prototüübis käsitletud funktsionaalsed ja mittefunktsionaalsed nõuded on osaliselt välja toodud ja neid on vaja täiendada süsteemi edasises arenduses. Tähtsaimateks nõueteks, mis on prototüübist välja jäetud on turvalisus ja konfidentsiaalsus, mille järgimine on väljaspool selle magistritöö teemat.

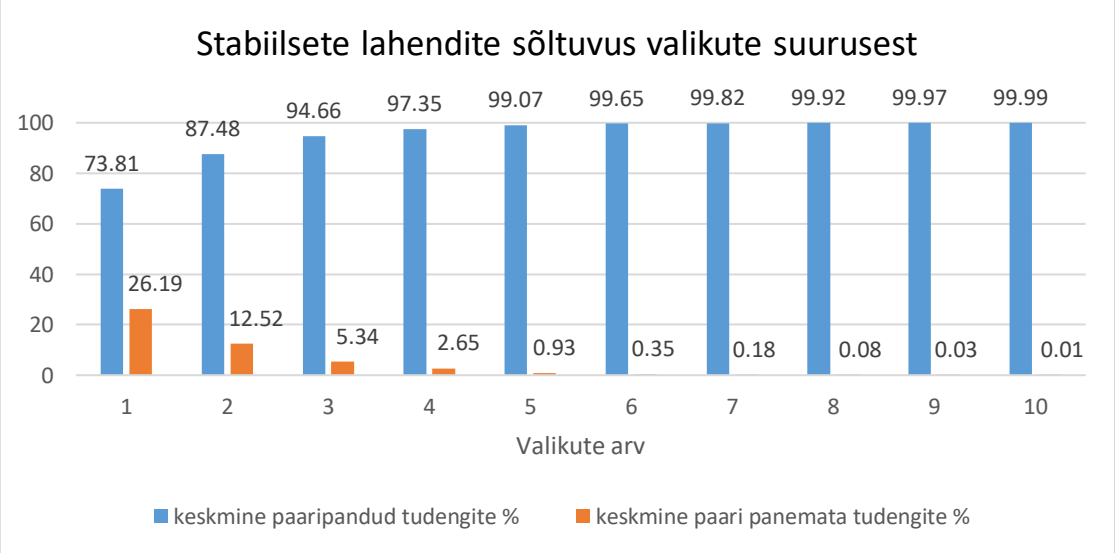
Rakenduse tuumfunktsionaalsus on lihtsasti laiendatav läbi uute äriloogika teenuste. Seoses lõputööde haldamisega on tuntud huvi ka lisafunktsionaalsusest, mida selles magistritöös veel ei jõutud käsitleda, kuid mida disainitud süsteemis olevate andmete põhjal saab realiseerida. Nende hulka kuulub ka automatiseeritud tagasiside andmise süsteem, mis kontrollib teatud kriteeriume lõputöös, mida on võimalik automatiseerida ja läbi selle säästa ülikooli töötajate aega ning andes tudengile kiiremat tagasisidet. Näitena varasematest uuringutest saab automatiseerida dokumendi struktuuri kontrollimist, viitamise korrektsust, viidete arvu ja töö pikkust. Need on üldlevinud parameetrid, mis peaks üldjuhul kehtima kõikide tehtavate tööde jaoks ning seeläbi automatiseeritavad [20] [21].

4.2 Lõputöö teemade ja tudengite paaride leidmine

Süsteemis rakendatud algoritmid tagastavad erinevaid tulemusi nende omaduste poolest. Neid tulemusi võrreldakse erinevate kriteeriumite põhjal ja leitakse, miks oli vajalik algoritmide kombineerimine, et saavutada parim tulemus.

4.2.1 Stabiilsete paaride leidmine

Testimise tulemus näitas, et algoritm suudab edukalt lahendada probleemi, kui on antud piisavalt suur arv valikuvõimalusi osapooltele ja nende tehtud individuaalsed valikud on piisavalt erinevad nende kaaslaste omadest. Tulemustest on näha, et keskmiselt 5 valikuga suudab kohandatud Gale-Shapely algoritm leida peaaegu kõikidele tudengitele nende eelistustes olnud teema (Joonis 7).

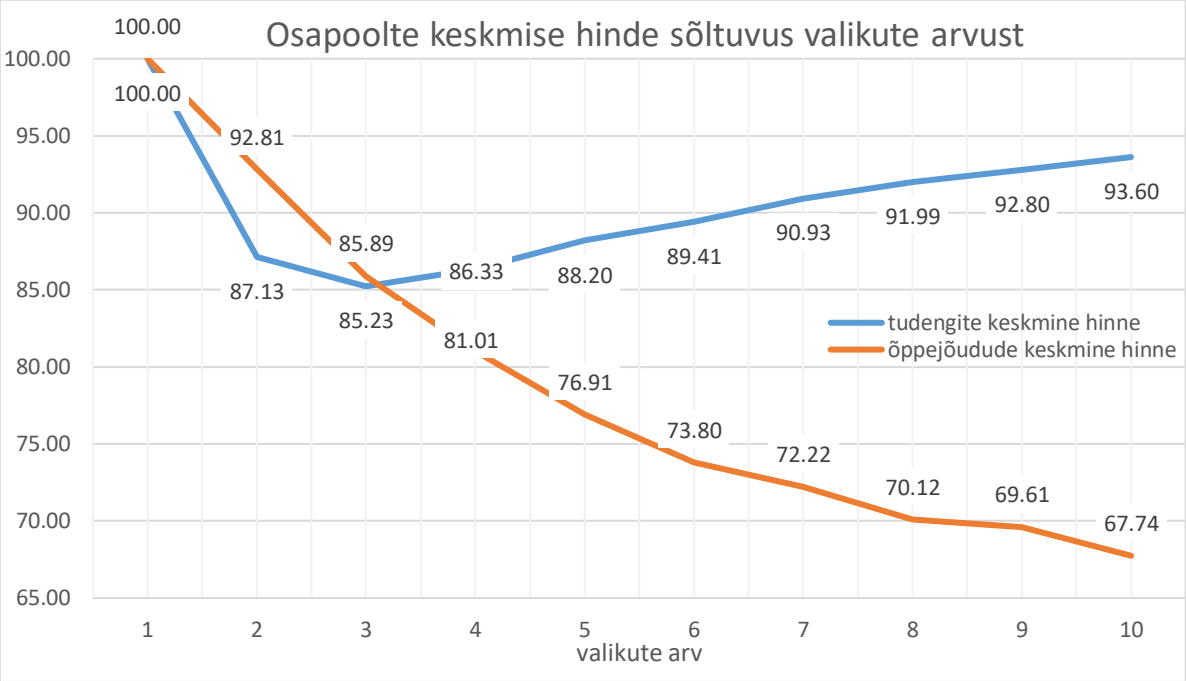


Joonis 7. Leitud stabiilsete lahendite suuruse sõltuvus valikute e arvust.

Suurem valikute arv ei kasvata enam märgatavalt lahendite tulemust. Kuna eelistused on juhuslikult genereeritud ja ühtlaselt jagunenud kõikide teemade vahel, siis ei saa kindlaks teha, kas 5 valikut on piisav suure hulga lahendite leidmiseks igas päriselus esinevas stsenaariumis – on võimalik, et sama edukuse saavutamiseks reaalses elus on vaja rohkem või vähem eelistusi, olenevalt eelistuste välja kujunemisele, mida on raske ette modelleerida.

Lisaks lahendis leiduvate osapoolte protsendile saab vaadelda ka iga moodustatud paari hinnet ja osapoolte hindeid, mis iseloomustab seda, kui hea või halva tulemuse kumbki osapool sai. Hinne on arvutatud protsendilise väärtusena, mis näitab analüüsis, kui kõrgel eelistustes osapool tulemuse sai, kus 100 tähistab oma kõige eelistatuma valiku saamist. Valisin sellise hindamise meetodi, et mõlemad osapooled oleksid võrreldavad hinde põhjal, sest vastasel juhul oleks olukord, kus tudengil võib olla rohkem valikuid kui õppejõul, sest teemale kandideeris vähem tudengeid kui lubatud eelistuste arv ja keskmise põhjal võrdlus ei oleks enam täpne.

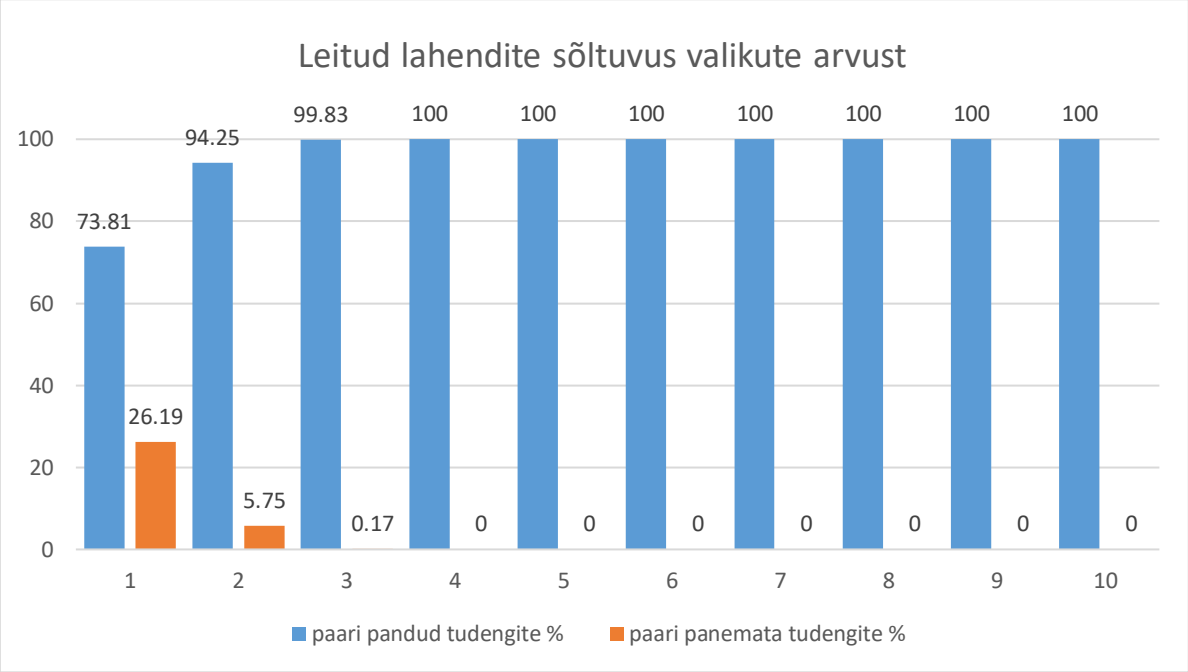
Stabiilsete lahenduste puhul leidsin, et valikute suurenedes tudengite keskmine hinne tõuseb, kuid õppejõudude keskmine hinne langeb, mida rohkem on lubatud valikuid teise osapoole suhtes teha (Joonis 8). Maksimaalne hinne 100 ühe valiku juures on tingitud sellest, et kui osapoolel on eelistustes vaid üks valik, siis lahendi leidmisel saab öelda, et kõik paari pandud osapooled ongi saavutanud maksimaalse ehk kõige rohkem eelistatud tulemuse, kuid arvesse ei võeta varem leitud tulemust, kus on leitud et 1 valiku juures on märkimisväärne osa tudengeid jäänud ilma teemast (Joonis 7).



Joonis 8. Osapoolte tulemuste hinde sõltuvus valikute arvust

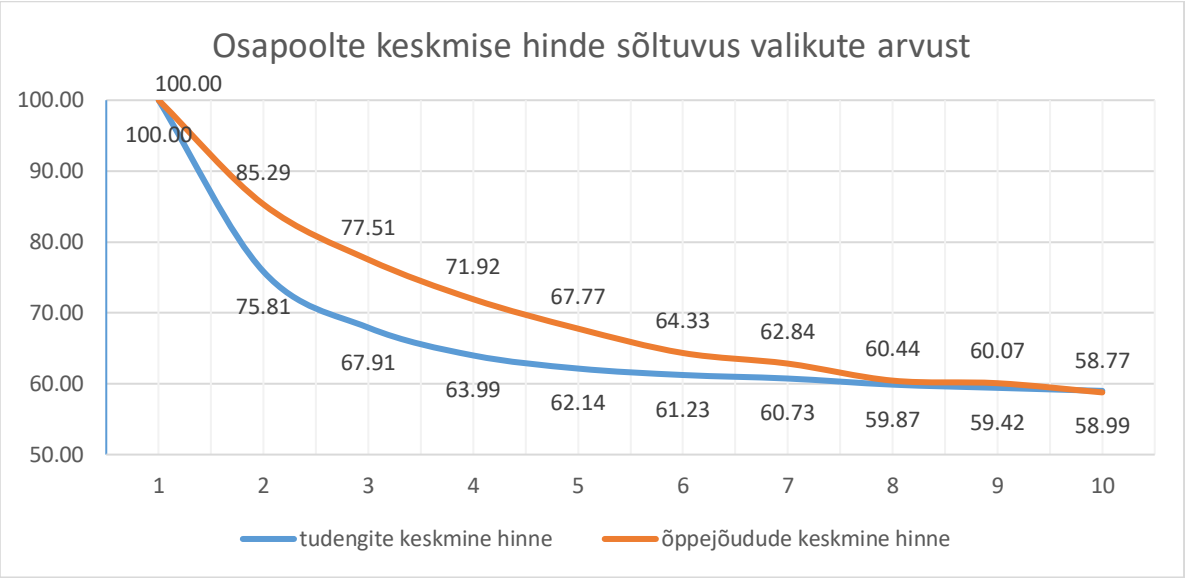
4.2.2 Maksimaalse katvusega paaride leidmine

Järgnevalt on toodud statistika algoritmi leitud lahenduste kohta, mis näitab, et võimalik on leida peaaegu kõik paarid 3 või enama valikuga (Joonis 9).



Joonis 9. Maksimaalse katvusega implementatsiooni lahendite sõltuvus valikute arvust

Lisaks paaride arvu sõltuvusele vaadeldi ka mõlema osapoole keskmist hinnet ja eelistuste suurenemise mõju sellele (Joonis 10). Tulemustes on näha selget langust leitud paaride hindes mõlema osapoole jaoks. Langus aeglustub valikute kasvades ja hinne läheneb 50%-le mis tähendab, et praktikas, kui on kasutusel viis või vähem eelistust, saavad mõlemad osapooled üle keskpärase tulemuse nende tehtud valikute hulgast. Õppejõudude keskmine hinne ei lange sama kiirusega, mis tudengite keskmine hinne, selle peamiseks põhjuseks on testandmete eripära, kus igal õppejõul on rohkem kui 1 teema, mille tõttu ei mõjuta üks madalama hindega paar keskmist tulemust suurel määral. See väljendub ka tulemustes, kus õppejõu keskmine hinne on tudengi omast kõrgem valikute hulga 2 kuni 9 puhul (Joonis 10).

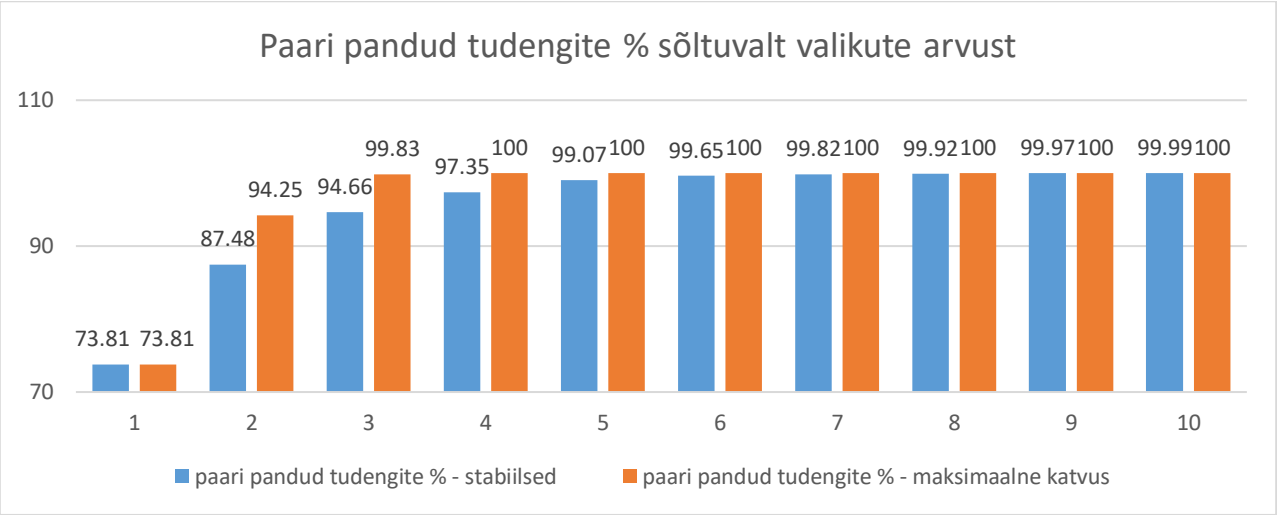


Joonis 10. Osapoolte keskmise hinde sõltuvus valikute arvust

4.2.3 Implementatsioonide võrdlus

Implementatsioonidele pandud erinevate kriteeriumite tõttu erinevad ka saavutatud tulemused. Gale-Shapely algoritmil põhinev stabiilsete paaride leidmise algoritm annab samade testülesannete juures keskmiselt parema tulemuse individuaalsetele osapooltele mõlemas paari pooles, kuid kõikide tippude paari panekuks nõuab algoritm rohkemaid eelistusi kasutajatelt, mis on peamiselt tingitud stabiilsuse kriteeriumist. Teine implementatsioon, mis koosneb Hopcroft-Karp ja Blossom V algoritmide kombinatsioonist nõuab sama katvuse saavutamiseks vähem eelistusi, sest stabiilsuse kriteerium ei pea olema täidetud, kuid osapoolte individuaalne keskmine hinne on madalam. Võrdlus põhineb samadel testandmetel ja tulemustest on arvutatud keskmine üle 100 iteratsiooni, et vältida juhuslikkusest tekkinud anomaaliaid testandmetes.

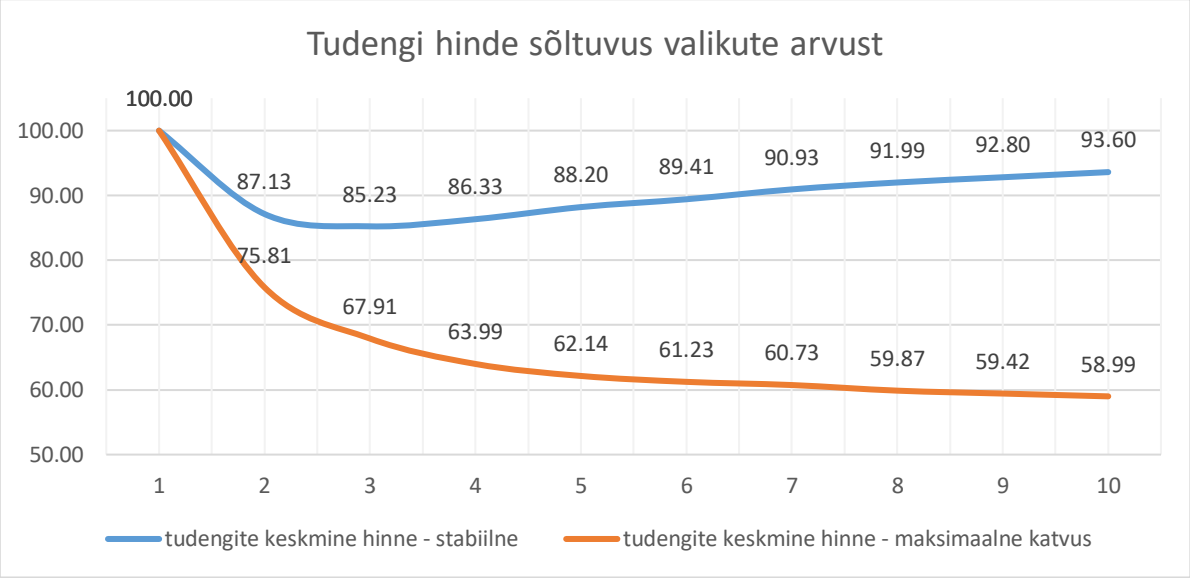
Kuna esmatähtis on suurima arvu paaride leidmine, saab võrrelda algoritme selle järgi, milliste parameetritega suudab algoritm saavutada suurima paaride arvu, ehk kui palju valikuid peaks süsteem keskmiselt nõudma osapooltelt, et tagada suurim kardinaalsus graafis. Mõlema lähenemisega kasvab leitud paaride protsent valikute suurenedes. 1 valiku juures on tulemused identsed, sest eksisteerib ainult üks lahendus, mis on samal ajal maksimaalse katvusega ja kõik paarid on ka stabiilsed. Valikute kasvades on näha, et maksimaalset katvust otsiv algoritm suudab leida kõikidele tudengitele paarid väiksemate valikute arvuga, kui stabiilseid paare otsiv algoritm, mis ei suutnud isegi 10 valikuga leida kõigis 100 iteratsioonis stabiilset lahendit, mis sisaldaks kõiki tudengeid (Joonis 11).



Joonis 11. Algoritmide võrdlus paari pandud tudengite % alusel

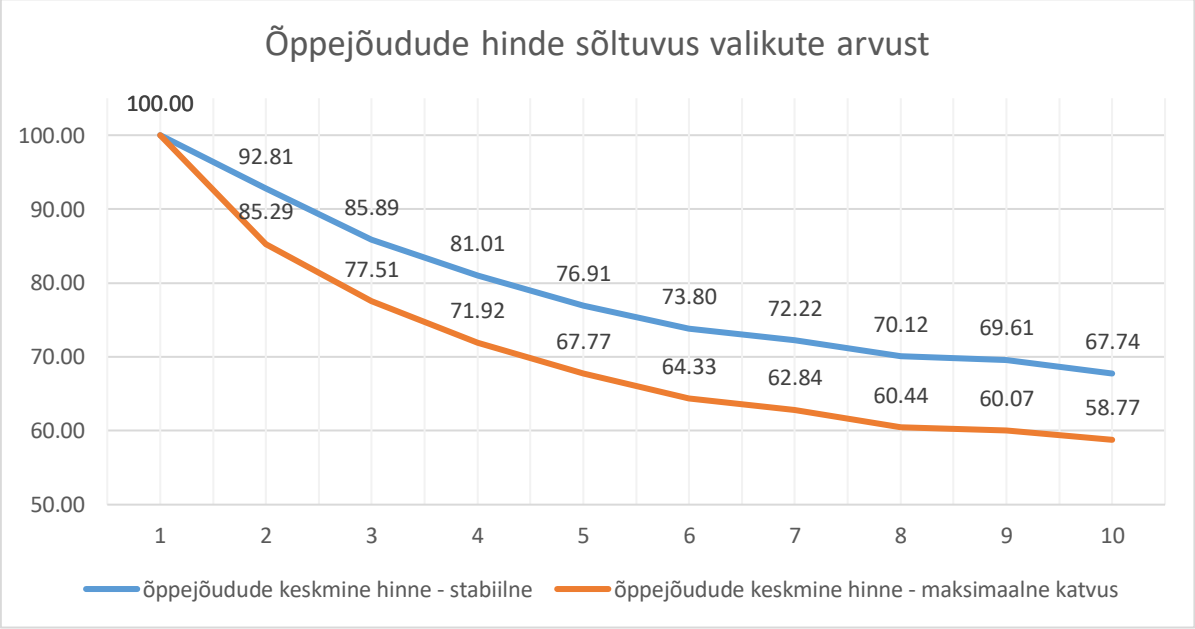
Gale-Shapely algoritmi puhul on näha, et alates 3 valikuga ülesannete puhul hakkab tudengite keskmine hinne tõusma valikute arvu suurenedes. See tuleneb sellest, et suuremate valikute arvu puhul on vähemtõenäoline, et tudengite vahel eelistused kattuvad suurel määral, mis tähendab, et keskmiselt saab iga tudeng endale parema tulemuse. Vähemate eelistuste puhul on olukord, kus kõrgema hindega eelistused võivad kattuda rohkematel tudengitel ja selle tõttu saavad suur osa tudengitest tulemusena teema, mis ei kattu teiste tudengitega, kuid asub madalamal nende eelistustes (Joonis 12).

Maksimaalse katvusega algoritmi puhul langeb keskmine hinne valikute kasvades, mis on tingitud sellest, et algoritm leiab maksimaalse katvusega lahendi, kuid olukordades, kus eksisteerib mitu maksimaalset lahendit, ei ole garanteeritud et see on nendest parima tulemusega (Joonis 12).



Joonis 12. Algoritmide võrdlus tudengite hinnete alusel

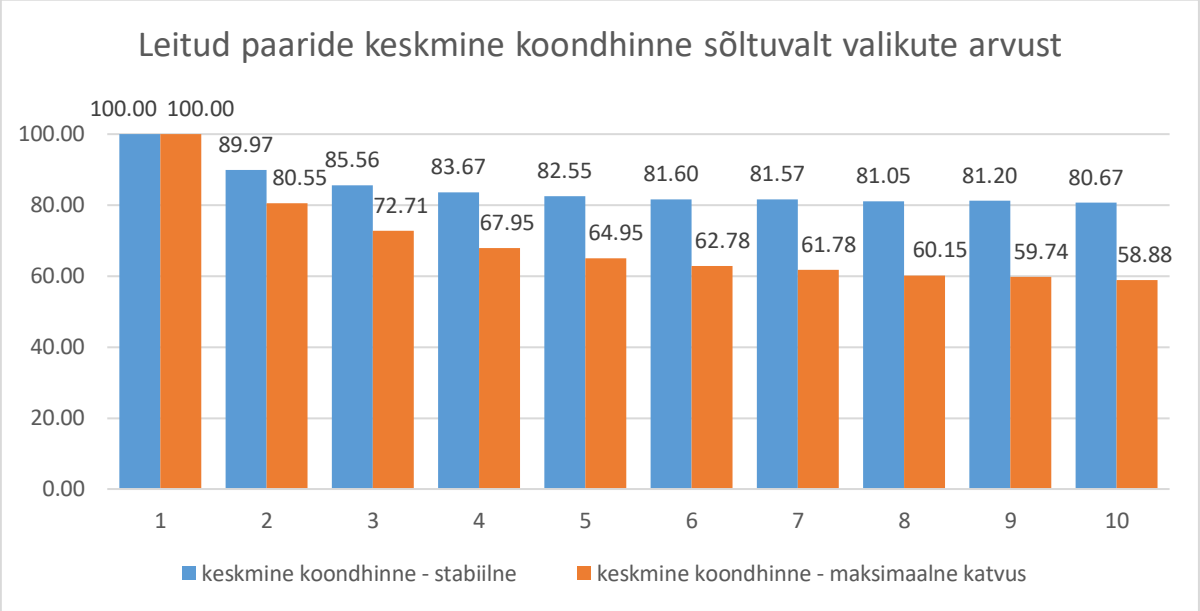
Mõlema implementatsiooni puhul õppejõudude hinne langeb valikute kasvades, sest valikute arvu suurenedes kandideerib igale teemale rohkem tudengeid, kes väiksemate valikute arvu korral oleks teema valimata jätnud, kuid nüüd valivad teema eelistustes viimastele kohtadele. Selle tõttu kasvab ka leitud paaride arv, mis tähendab, et rohkem õppejõude saavad kehvema eelistuse. Väiksemate valikute korral jääks teema vabaks (Joonis 13).



Joonis 13. Algoritmide võrdlus õppejõudude hinnete alusel

Koondhinne, mis on moodustatud individuaalsete hinnete keskmise põhjal iseloomustab leitud tulemuse kvaliteeti, kus on arvestatud mõlema osapoole saavutatud individuaalseid

tulemusi. Stabiilsete lahendite puhul on näha, et koondhinne on kõikide valikute hulkade juures suurem või võrdne, kui maksimaalse katvusega lahenditega tulemustes (Joonis 14).



Joonis 14. Leitud paaride keskmine koondhinde sõltuvalt valikute arvust

Lisaks tulemuse stabiilsusele on tähtis ka see, et leitav lahendus katab võimalikult suure hulga tippe graafis, et praktikas oleks kasutatavast süsteemist kasu võimalikult paljudele inimestele. Teadmine, et paljud eelistused esitanud tudengid võivad jääda ilma teemata motiveerib vähem osapooli süsteemi kasutama ning suur hulk lahendeid automaatselt mitte leidnud tudengeid tähendab ka seda, et süsteem ei täida ühte oma eesmärki milleks on automatiseerida ja kiirendada kogu protsessi, sest teemata jäänud tudengid peavad edasi tegutsema süsteemi väliselt ja nende jaoks on kogu protsessist osavõtt olnud tulemuseta ajakulu. Jättes stabiilsuse kriteeriumi kõrvale, on võimalik kasutada mitmeid graafiteooria algoritme, mis on loodud kahepoolsetes graafides paaride leidmiseks. Tulemuste põhjal kombineeriti algoritmid eesmärgiga leida suurima katvusega lahend ja võimalusel valida tulemustest stabiilne lahend, mille paaride ja individuaalsed hinded osutusid kõrgemaks kui maksimaalse katvusega algoritmil.

Sõltuvalt algoritmi valikust saame otseselt mõjutada seda, milliseks kujunevad moodustatud paaride omadused ja kui palju neid moodustatakse. Ühtlasi mõjutab see ka mittefunktsionaalseid parameetreid nagu jõudlus, kuid inimeste lõplikke otsuseid algoritm ei saa mõjutada ega ka ette ennustada. Selle tõttu on võimalikud ka stsenaariumid, kus ükski algoritm ei ole võimeline rahuldavat tulemust andma. Näiteks liiga väikse koguse teemade puhul, kui teemasid on vähem üles pandud kui

kandideerivaid tudengeid, ei ole võimalik saavutada optimaalset tulemust. Sellega seotud probleemiks on ka olukord, kus teemasid võib olla küll piisavalt, kuid tudengite valikud koonduvad väikse koguse teemade ümber, mis on väiksem kui tudengite hulk. Sama kehtib ka õppejõudude valikute osas, kui teatud tudengid on palju rohkem eelistatud õppejõudude seas kui teised tudengid. Selliste stsenaariumite puhul ei ole võimalik kõikidele kandideerinud tudengitele teemat leida.

Graafis võib eksisteerida mitu maksimaalse katvusega lahendit, millest saaks tulemuse leida. Hopcroft-Karp algoritm leiab vaid ühe nendest võimalikest tulemustest. Võimalik on, et eksisteerib parem maksimaalne katvus, kus osapoolte individuaalsed tulemused on paremad, kui Hopcroft-Karp leitud tulemuses. Selleks on vaja loendada kõik maksimaalsed lahendid, mille hulgast valida maksimaalse hindega lahend. Selleks on Takeaki UNO kirjeldanud algoritmi, mida saab süsteemis implementeerida parima lahendi leidmiseks [22]. Praegu eksisteerib sellest implementatsioon Pythonis Guangzhi XU poolt, mida pole valideeritud ning selle liidestamine Java rakendusega on ebaotstarbekas [23].

Järgmiseks sammuks saab uurida võimalusi, kuidas efektiivselt loendada kõiki maksimaalse katvusega lahendeid, et teha kindlaks, kas kõikide lahendite loendamine ja nende hulgast parima leidmine parandab märgatavalt tulemust paaride leidmisel. Praegune lahendus juba leiab suurima katvusega lahenduse, seega saaks parem tulemus tulla vaid osapoolte hinnete juures. Erinevatesse maksimaalsete katvusega lahenditesse ei kuulu ilmtingimata alati samad tudengid ja teemad. Selle tõttu vajab kaalumist ka see aspekt, milline lahend valida olukorras, kui näiteks kaks lahendit sisaldavad sama palju paare, on hinnetelt sama head, kuid mõlemal juhul jääb teatud hulk erinevaid tudengeid teemast ilma.

4.3 Teemade hindamine

Valisin modifitseeritud Borda count algoritmi lõputööde järjestamiseks, sest testimisel sain kinnitust, et algoritm suudab ka omavahel mitteseotud alamjärjestused agregeerida üldiseks paremusjärjestuseks. Hääletusalgoritmi kasutusele võtul ei vaja programm lisaandmeid tulemuse saavutamiseks, piisab vaid kõikidest hääletustulemustest, mille põhjal saab tagastada paremusjärjestuse. Samuti ei vajanud Borda count rakendamine spetsiifilisi raamistikke, sest on lahendatav kasutades tüüpilisi

programmeerimismeetodeid. Selline lähenemine võimaldab programmi koheselt kasutusele võtta ja ei vaja lisatööd.

Leidsin, et Borda count suudab leida paremusjärjestuse ja kasutades uuritud viikide murdmise meetodeid, suudab ka enamus viike murda. Siiski leidsin stsenaariume, kus jääb üks või rohkem paari töid võrdsesse seisu (Lisa 2 – Borda count näide viigiga). Selles näites ei suutnud hetkel rakendatud viikide murdmise meetodid lahendada ühte viigiseisu, sest mõlemad lõputööd on sama hindega ning algoritmi töö käigus said sama arvu punkte järjestuste põhjal.

Vajab edasi uurimist, kas hindajate arvu suurendamine vähendab viikide tekkimise tõenäosust. Kuna peamiseks viikide murdmise meetodiks on paarikaupa võrdlemine, siis on vaja uurida, kuidas on seotud võrreldavus ja loogika, mille alusel määratakse hindajatele töid. Vajab täiendavat uurimist, kuidas Tallinna Tehnikaülikoolis käsitleda viikide murdmist, nagu näiteks kontekstipõhine viikide murdmine, kus võrreldakse ka lisaks kahele viigis olevale tööle mõlemat tööd ümbritsevaid kontekstuaalseid andmeid nagu näiteks teiste tööde kvaliteeti alamjärjestuse piirides.

4.3.1 Tööde hindamine tehisnärvivõrkude põhjal

Hääletusalgoritmidest erinev lähenemine on kasutada masinõpet ja tehisnärvivõrke, et automatiseerida hindamisprotsessi, jättes kriteeriumipõhise hindamise loogika alles. See nõuaks treenimist juba hinnatud tööde põhjal. Rong Li, Xiangjun Wang ja Dingyuan Li leidsid oma avaldatud uurimuses, et Hopfield-i närvivõrgul põhinev hindamisalgoritm suutis hinnata lõputöid sama hästi kui traditsiooniline kaalutud kriteeriumite põhjal hindamine [24]. Selline lähenemine nõuaks retsensentidelt kriteeriumite hindamist, mis oleks närvivõrgule sisendiks, et saada tulemuseks terviklik hinne. Samuti oleks vajalik mudeli treenimine treeningandmetega, mis hetkel puuduvad, seega ei saa süsteemi koheselt rakendada, vaid on vaja viia läbi eelnev andmete kogumine, kas siis vanade tööde uuesti hindamise põhjal või võttes sisendiks tulevaste aastate lõputööd, eeldades, et need on hinnatud vastavalt närvivõrgus olevatele kriteeriumitele. Selline lähenemine võimaldab ka treenida erinevad mudelid vastavalt vajadustele, näiteks saab treenida mudelid eraldi nii magistri- kui ka bakalaureusetööde jaoks ja eraldi mudelid vastavalt teaduskonnale, erialale või muule kriteeriumile. Nagu on näidanud [24] [25], võib hindamismudel sisaldada mitte ainult retsensendi sisendit, vaid ka juhendaja ja hindamiskomisjoni sisendeid. Kuigi uurimuse autorid väidavad, et see meetod

hindamiseks suurendab objektiivsust ja hinde aksepteeritavust, siis täielikult pole seda võimalik elimineerida, sest sisendiks antavad kriteeriumite põhjal punktid on siiski üksikute inimeste arvamused. Kui aga mudel on treenitud kõikide õppejõudude hinnangute põhjal, siis võib öelda, et see osa otsustamismehhanismist esindab kogu teaduskonna arvamust, mis ongi eesmärgiks ja see aitab vähendada subjektiivsust. Võimalik oleks ka pidev mudeli treenimine selleks, et aastatega kogunenud hinnangud peegeldaks ka tulevikus tehtavaid otsuseid hindamisel. Kuigi selle süsteemi eesmärgiks ei ole otseselt järjestuse koostamine, siis saab siiski seda närvivõrku selleks kasutada, sest väljundiks on ujuvkomaarv, mis on sarnaselt kaalutud hindele saadud ja seega oleks vaja lihtsalt kõik väljundid sortida naturaaljärjestuse järgi. Kuigi hindamismudel ja kriteeriumid erinevad, saab selliseid süsteeme kohandada vastavalt vajadusele sest need on ülesehituselt ja tööpõhimõttelt piisavalt üldised [25]. Tehisnärvivõrgu kasutusele võtuks ja kohandamiseks Tallinna Tehnikaülikoolis juba eksisteerib kriteeriumite loend, mida eeskujuks võttes saab ka disainida meile sobiva süsteemi [1]. Treeningandmete kogumine ning treenimine piisava täpsuse saavutamiseks ja kriteeriumite formuleerimine närvivõrgu jaoks nõuab täpsemat edasi uurimist. Samuti tuleb kaaluda ka seda, millist tehisnärvivõrku kasutada konkreetses ülesandes, võttes arvesse hindamise eripära [26].

4.3.2 Rakendamine infosüsteemis

Kummagi pakutud lahenduse rakendamine süsteemis vajab erinevat lähenemist nende sisulise erinevuse tõttu. Sõltuvalt lähenemisest vajab rakendus erinevaid andmeid ja teeke nendel andmetel opereerimiseks. Praegu programmeeriti see osana peasüsteemist, kuid edasiseks analüüsiks ja ka rohkemate algoritmide lisamiseks on otstarbekas ka see funktsionaalsus eraldi komponenti viia sarnaselt paaripaneku algoritmide lahedusele.

Masinõppe ja tehisnärvivõrkude põhjal hindamissüsteemi rakendamine nõuab rohkem eeltööd, rohkem andmeid ja efektiivseks lahendamiseks ka spetsiaalseid raamistikke. Peamiselt on vajadus treeningandmetele, mille põhjal mudelit treenida, mille tõttu on vaja ka toetavat funktsionaalsust nende andmete kogumiseks ja käitlemiseks. Masinõppes on sageli kasutusel ka spetsiifilised teegid ning raamistikud ja sõltuvalt vajadusest võib nende valik määrata ka programmeerimiskeele valiku. Näiteks Python programmeerimiskeele valikul tuleks see osa süsteemist teha eraldi komponendiks, sest ülejäänud süsteem on programmeeritud Javas. Standartseks suhtlusviisiks komponentide vahel sel juhul oleks HTTP päring.

Lõputööde hindamise kontekstis vajab uurimist, kui tõenäoline on, et tekivad viigid ja kui palju aitab retsensentide, kes loeavad ühte konkreetset tööd, arvu kasvatamine viikide vähendamisel. Lisaks on võimalik ka rakendada kokkuleppelisi ärilisi meetodeid viikide murdmiseks, nagu näiteks otsustamise delegeerimise järgmisele hindajale. Kuigi selline lähenemine pole eelistatud, sest paljude viikide puhul kaotab süsteem ja algoritm oma esialgse mõtte, siis võib see osutuda siiski efektiivseks, kui viike esineb harva. Suure hulga tööde hindamisel on alamjärjestus suhteliselt väike võrreldes tööde arvuga, siis on leitud, et Harmonic hääletamise reegel on edukamalt hääletustulemusi ennustanud kui Borda count ja seega üks võimalik meetod, kuidas praegu saavutatud tulemusi parandada [27].

5 Kokkuvõte

Lõputöö teema valik ja tööde hindamine on tähtsad protsessid kõrgkooli lõpetamisel, mille läbipaistvamaks muutmine ja automatiseerimine tuleb kasuks kõigile osapooltele. Praegu puuduvad süsteemid automatiseeritud teemade jaotamiseks ja tööde paremusjärjestuse moodustamiseks Tallinna Tehnikaülikoolis. Töö eesmärgiks oli uurida viise, kuidas lahendada teemade jagamise probleemi, võttes arvesse nii tudengite kui juhendajate eelistusi ning kuidas luua paremusjärjestus töödest, kus paljudel töödel on sama hinne.

Eelistuste põhjal paaride leidmiseks uuriti erinevaid graafiteoreetilisi algoritme, et leida võimalikult paljudele tudengite automaatselt teema. Tööde paremusjärjestuse moodustamiseks uuriti erinevaid hääletusalgoritme ja masinõppealgoritme, et reastada kõik tööd võttes arvesse retsensentide osalised paremusjärjestused. Lahenduste praktiliseks kasutuselevõtuks disainiti infosüsteem.

Leidsin, et teemade ja tudengite kokku viimine on lahendatav kasutades kombinatsiooni mitmest paaride leidmise algoritmist. Tudengite tööde reastamine retsensentide hinnangute põhjal on võimalik kasutades modifitseeritud Borda count algoritmi. Need algoritmid leidsid rakendust prototüüp infosüsteemis, mille programmeerisin ja disainisin töö käigus Java programmeerimiskeeles.

Leidsin, et alates 3 eelistusega suudavad kombineeritud algoritmid leida üle 99% paaridest, kuid stabiilse lahendi leidmine nõuab keskmiselt 5 või enam eelistust. Rakendatud algoritme on võimalik edasi arendada, leides võimalikke parema kvaliteediga tulemusi. Rakendatud Borda count algoritm suudab järjestada enamiku töödest, kuid sõltuvalt hinnetest ja tööde jagamise loogikast, ei pruugi lahendada kõiki viigiseise hetkel implementeeritud viikide murdmise meetoditega.

Kasutatud kirjandus

[1] Tallinna Tehnikaülikool, „Lõputööde hindamine informaatika õppekavadel,“ [Võrgumaterjal]. Available: https://www.ttu.ee/teaduskond/infotehnoloogia-teaduskond/it-tudengile/bakalaureuseope-48/informaatika-27/loputoode-hindamine-2/. [Kasutatud Veebruar 2020].

[2] D. F. Manlove, Algorithmics of Matching Under Preferences, kd. 1, 2013.

[3] R. Irving, P. Leather ja D. Gusfield, „An efficient algorithm for the “optimal” stable marriage.,“ *ACM,* nr 34, pp. 532-543, 1987.

[4] W. contributors, „Comparison of electoral systems,“ Wikipedia, The Free Encyclopedia, 31 Märts 2020. [Võrgumaterjal]. Available: https://en.wikipedia.org/w/index.php?title=Comparison_of_electoral_systems&oldid=948343913. [Kasutatud 10 Aprill 2020].

[5] M. T. Hagan, H. B. Demuth ja M. Beale, Neural Network Design, 1995.

[6] „JGraphT,“ [Võrgumaterjal]. Available: https://jgrapht.org/. [Kasutatud jaanuar 2020].

[7] The Apache Software Foundation, „Using Apache Commons CSV,“ 2020. [Võrgumaterjal]. Available: http://commons.apache.org/proper/commons-csv/. [Kasutatud märts 2020].

[8] Stack Exchange Inc, „Developer Survey Results,“ 2019. [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2019#technology. [Kasutatud märts 2020].

[9] D. Gale and S. S. Lloyd, “College admissions and the stability of marriage.,” *The American Mathematical Monthly,* pp. 9-15, 1962.

[10] N. Nisan, T. Roughgarden, E. Tardos ja V. V. Vazirani, „Algorithmic Game Theory: Algorithmic Mechanism Design,“ *,* 2007.

[11] J. E. Hopcroft ja R. M. Karp, „A n5/2 algorithm for maximum matchings in bipartite,“ in *12th Annual Symposium on Switching and Automata Theory (swat 1971),* 1971.

[12] V. Kolmogorov, „Blossom V: a new implementation of a minimum cost perfect matching algorithm,“ *Mathematical Programming Computation,* kd. 1, nr 1, pp. 43-67, 2009.

[13] J. Cullinan, S. K. Hsiao ja D. Polett, „A Borda count for partially ordered ballots,“ *Social Choice and Welfare,* kd. 42, nr 4, pp. 913-926, 2014.

[14] J. C. Mayer, „Voting and Fairness 4,“ [Võrgumaterjal]. Available: https://people.cas.uab.edu/~jcmayer//Voting%20Methods%204.pdf. [Kasutatud 10 Aprill 2020].

[15] Y. K. Kwok, „Mathematics and Social Choice Theory,“ [Võrgumaterjal]. Available:

https://www.math.ust.hk/~maykwok/courses/MATH4823/MATH4823_Topic4.pdf. [Kasutatud 10 Aprill 2020].

[16] G. R. L. Franco ja C. Y. C. D. Guzman, „Design and implementation of a web-based thesis coordinator system (TCS),“ in *2016 IEEE Region 10 Conference (TENCON)*, 2016.

[17] C. Meimei, „Design and implementation of web-based system for graduation thesis management,“ in *2011 International Conference on E-Business and E-Government (ICEE)*, 2011.

[18] Z. Chen, L. Tian, D. Li, Z. Peng ja X. Wang, „WEB-based bidirectional subject-selection system for graduation thesis,“ in *2010 3rd International Conference on Computer Science and Information Technology*, 2010.

[19] M. Gutkowski, J. Wojciechowski, B. Sakowicz ja A. Napieralski, „Thesis Management Supporting System based on J2EE Platform,“ in *2007 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics*, 2007.

[20] B. Li, „Design and Implementation of the Thesis Evaluation System Based on Web Technology,“ in *6th International Conference on Information Engineering for Mechanics and Materials*, 2016.

[21] M. Sulir ja J. Poruban, „Towards automated assessment of students' preliminary thesis submissions,“ in *2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2015.

[22] T. Uno, „Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs,“ *international symposium on algorithms and computation*, pp. 92-101, 1997.

[23] G. XU, „Xunius / bipartite_matching: Enumerate all maximum matchings in bipartite graph in Python,“ 29 juuni 2017. [Võrgumaterjal]. Available: https://github.com/Xunius/bipartite_matching. [Kasutatud jaanuar 2020].

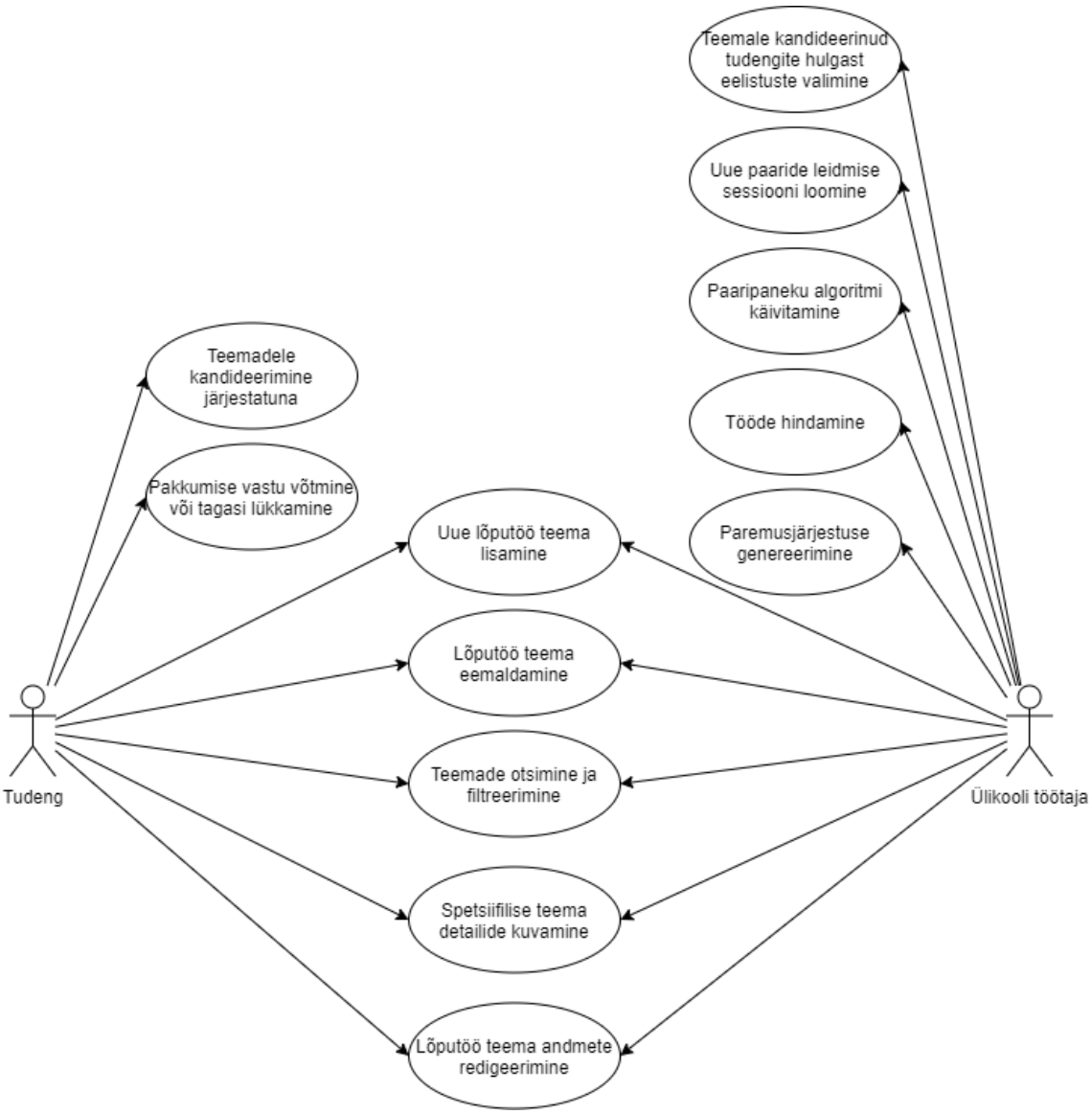
[24] R. Li, X. Wang ja D. Li, „Research on Graduation Thesis Evaluation Based on Hopfield Neural Network,“ in *2014 Seventh International Symposium on Computational Intelligence and Design*, 2014.

[25] H.-b. Mao, Q. Jiang, A.-h. Zhou ja S.-f. Wang, „Model and research on score evaluation model of graduation thesis,“ in *2014 9th International Conference on Computer Science & Education*, 2014.

[26] J. K. Basu, D. Bhattacharyya ja T.-h. Kim, „Use of Artificial Neural Network in Pattern Recognition,“ , 2010.

[27] M. Ayadi, N. B. Amor ja J. Lang, „Approximating Voting Rules from Truncated Ballots.,“ *arXiv preprint arXiv:2002.06009*, 2020.

Lisa 1 – Infosüsteemi peamised kasutusjuhud



Lisa 2 – Borda count näide viigiga

Hinnatav töö	Töö 1	Töö 2	Töö 3	Töö 4	Töö 5	Töö 6	Töö 7	Töö 8	Töö 9	Töö 10
Hindaja 1	A	B	C	D	E	A	B	C	D	E
Hindaja 2	C	D	E	A	B	C	D	E	A	B

Prognoositavad hinded ja paremusjärjestus:

Hinne 5	Töö 1, Töö 9
Hinne 4	Töö 2, Töö 6, Töö 7
Hinne 3	Töö 3, Töö 8, Töö 10
Hinne 2	Töö 4
Hinne 1	Töö 5

Borda count tulemus:

1	9	2	3	6;7	8	10	4	5