TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Ehituse ja arhitektuuri instituut

KORRUSHOONETE JÄIKUSSEINTE PAIKNEMISE AUTOMATISEERIMINE LÄBI PARAMEETRILISE DISAINI

AUTOMATION OF SHEAR WALL PLACEMENT IN MULTI-STOREY BUILDINGS USING PARAMETRIC DESIGN

MAGISTRITÖÖ

Üliõpilane: Matthias Rimm

Üliõpilaskood 165203

Juhendaja: Raido Puust

Tallinn 2023

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

15. mai 2023

Autor: .....
/ allkiri /

Töö vastab magistritööle esitatud nõuetele.

".....” 2023

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud

".....”. 2023

Kaitsmiskomisjoni esimees:

.....
/ nimi ja allkiri /

LIHTLITSENTS LÕPUTÖÖ REPRODUTSEERIMISEKS JA LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS

Mina, **Matthias Rimm,**

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Korrushoonete jäikusseinte paiknemise automatiseerimine läbi parameetrilise disaini,

mille juhendaja on Raido Puust

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

15.05.2023

Ehituse ja arhitektuuri instituut

LÕPUTÖÖ ÜLESANNE

Üliõpilane: **MATTHIAS RIMM** Üliõpilaskood **165203**

Õppekava: **EAEI02 Ehitiste projekteerimine ja ehitusjuhtimine**
Peaeriala: Ehitiste projekteerimine

Lõputöö teema:

KORRUSHOONETE JÄIKUSSEINTE PAIKNEMISE AUTOMATISEERIMINE LÄBI PARAMEETRILISE DISAINI

Automation of shear wall placement in multi-storey buildings using parametric design

Juhendaja: **Raido Puust** raido.puust@taltech.ee

Lõputöö konsultandid:

Tiitel või ametikoht, Ees- ja Perekonnanimi	Kontakt (e-post või telefon)	Allkiri ja kuupäev

Lõputöö põhieesmärgid:

1. Automatiseeritud jäikusseinte projekteerimise eelised võrreldes manuaalse protsessiga.
2. Olemasolevate jäikusseinte automaatset projekteerimist käsitletavate uurimistööde analüüs.
3. Programmi koostamine optimaalselt paiknevate jäikusseinte skeemide genereerimiseks.

Töö keel: eesti keel

Lõputöö etapid ja ajakava:

	Ülesande kirjeldus	Tähtaeg
1.	Kirjanduslik ülevaade ja olemasolevate tööd analüüs	26.02.2023
2.	Jäikusseinte paiknemise automatiseerimise programmi loomine	09.04.2023
	• Seinte võimalike asukohtade määramise algoritm	….
	• Horisontaal- ja vertikaalkoormuste jaotus ja sisejõudude leidmine	….
	• Tugevuskontrollid	….
	• Optimeerimise funktsioon	….
	• Täiendavate funktsioonide lisamine programmi	
4.	Tulemuste analüüs, ettepanekud tööprotsesside muutmiseks	23.04.2023
5.	Sissejuhatuse ja kokkuvõtte kirjutamine	30.04.2023
6.	Lõplik vormistamine	08.05.2023

	Lõputööde 95% ülevaatus, mille läbimine on kaitsmise eelduseks	**08.05.2020**

Esitlusmaterjalid kaitsmisel: Powerpoint esitlus ja jaotusmaterjalid

	Kirjeldus	Tähtaeg
1	Powerpoint esitlus	29.05.2023

Lõputöö esitamise tähtaeg: 08. mai 2023

Lõputöö ülesanne välja antud: 19.02.2021

Juhendaja: Raido Puust

Ülesande vastu võtnud: Matthias Rimm

Avalikustamise piirangu tingimused: puuduvad

SISUKORD

SISSEJUHATUS

Aina rohkem ehitatakse kõrghooneid, mille jäikus tagatakse raudbetoonist seinte süsteemidega. Jäikussüsteem võib moodustada märkimisväärse osa hoone ehitushinnast ning süsinikdioksiidi heidetest, mistõttu on ökonoomse jäikussüsteemi projekteerimine kasulik nii tellijale kui ka keskkonnale. Siiski on optimaalsete jäikusseinte laotuste loomine keeruline, kuna nende erinevate variantide arv on väga suur. Samuti on teiste hoone projekteerimisega seotud osapoolte soovid jäikusseinte paiknemisele tihti vastuolus ehitusinseneri omadega. Üldiselt tuginevad insenerid sobivate laotuste leidmiseks abistavatele reeglitele ja oma varasemale kogemusele, kuid sellisel juhul on parima variandi leidmine ebatõenäoline, sest insener ei suuda ette kujutada kõiki võimalikke lahendusi ega neid kõiki läbi proovida. Üheks põhjuseks, mis teeb jäikusseinte laotuste läbiproovimise aeganõudvaks tööks, on kasutatav tarkvara. Üldiselt projekteeritakse jäikusseinu varrasskeemidega, kasutades selle jaoks lõplike elementide meetodil põhinevat arvutustarkvara. Nendega töötamine on tihti aga ajamahukas pikkade arvutusaegade ja ebamugava kasutajaliidese tõttu. Seega võib optimaalsete laotuste leidmine takerduda ühe variandi loomiseks kuluva aja taha.

Antud töö ülesandeks on luua jäikusseinte projekteerimiseks spetsiaalne programm, mis võimaldab ehitusinseneril proovida kiiremini läbi rohkem jäikusseinte laotusi, tagades parema tulemuse leidmise vähema ajaga. Esmalt luuakse programm jäikusseinte laotuste analüüsimiseks ja dimensioneerimiseks, kasutades analüütilisi meetodeid, mis suudaks arvutused teostada kiiremini kui saadaval olevad lõplike elementide meetodi arvutustarkvarad. Seejärel kasutatakse loodud programmi koos evolutsioonilise optimeerimise algoritmiga, mis võimaldab automatiseerida jäikusseinte laotuste loomist, andes insenerile täiendavaid variante, mille peale ta ise ei võinud tulla.

Töös kasutatakse joonestamise tarkvara Rhinoceros 3D (Rhino) ja sellele loodud visuaalse programmeerimise lisandit Grasshopperit, et luua programm jäikusseinte projekteerimiseks ning erinevate laotuste genereerimiseks. Laotuste genereerimiseks kasutatakse täiendavalt Grasshopperi jaoks loodud evolutsioonilise optimeerimise tarkvara nimega Wallacei.

Antud töös antakse esmalt ülevaade varasematest uurimistöödest jäikusseinte laotuste optimeerimisest peatükis 1. Sellele järgneb töö käigus loodud jäikusseinte projekteerimiseks loodud programmi kirjeldus peatükis 2 ning kuidas seda kasutatakse optimeerimise algoritmiga jäikusseinte laotuste automaatseks loomiseks peatükis 3. Peatükis 4 viiakse laotuste genereerimise programmiga läbi katseid, milles uuritakse algoritmi efektiivsust erinevate hoonete puhul ja saadud tulemusi. Viimasena analüüsitakse peatükis 5 loodud programmi tugevusi ja nõrkusi, millistes olukordades on seda kõige otstarbekam kasutada ning kuidas seda tulevikus edasi arendada.

Võtmesõnad: Jäikusseinad, parameetriline disain, automatiseerimine, raudbetoon, magistritöö

1. KIRJANDUSE ÜLEVAADE

Selles peatükis käsitletakse jäikusseinte projekteerimist tänapäeval, sellega kaasnevaid puudusi ning kuidas neid on üritatud varasemalt lahendada. Peamiselt on varem uuritud optimeerimise algoritmide kasutamist jäikusseinte laotuse loomiseks, et leida ökonoomsemaid lahendusi kui insener selleks võimeline on. Lisaks käsitletakse töödes kasutatud arvutusmeetodite eeliseid ning kuidas on leevendatud nende puudusi.

1.1 Jäikusseinte projekteerimine tänapäeval

Hoone jäikussüsteemide projekteerimine toimub tänapäeval üldiselt manuaalselt. Insener proovib läbi erinevaid variante, muutes seinte asetust ja mõõtmeid ning vaatab nende mõju pingetele ja deformatsioonidele, kuni leiab sobiva lahenduse. Kuna lahendus peab sobima ka teistele projekteerimisega seotud osalistega, kelle nõudmised jäikusseintele on tihti vastuolus üksteisega, siis peab insener tihti looma mitmeid erinevaid variante enne kui leitakse kõigile sobiv lahendus [1]. Seetõttu on jäikusseinte projekteerimine aeganõudev ja töömahukas ning materjali kulu suhtes optimaalse variandi leidmine ebatõenäoline [2].

Jäikusseinte laotuste koostamisel on otstarbekas kasutada arvuti abi, kuna erinevate võimaluste arv on väga suur. Arvuti suudaks sama ajaga rohkem variante läbi proovida kui inimene, lühendades sellega projekteerimiseks kuluvat aega ning leides parema tulemuse kui inimene selleks võimeline on. Viimase 30 aastaga on hoogustunud raudbetoon tarindite optimeerimise uurimine, mille põhjuseks võib pidada rohkemate kõrghoonete ehitamist, mille jäikus tagatakse tihti raudbetoonist jäikusseintega. Samal ajal on arvutid muutunud võimekamaks, mistõttu on mitmete optimeerimise algoritmide kasutamine muutunud keerulisemate probleemide puhul praktiliseks [3]. Sellegipoolest ei ole tänapäeval ehituskonstruktsioonide projekteerimise valdkonnas optimeerimise algoritmide kasutamine väga levinud, sest need on ajamahukad ega suuda arvestada mitmete teguritega, mis esinevad päris elus projekteeritavates hoonetes. Optimeerimise algoritmid sisaldavad endas üldiselt ainult kandekonstruktsioonide mehaanilisi omadusi iseloomustavaid aspekte nagu pinged ja deformatsioonid, mis ei pruugi saada määravaks lõpliku jäikusseinte laotuse valikul. Juhul kui jäikusseinte asukohtade määramise ajal eksisteerib juba hoone arhitektuurne põhiplaan on kõige ökonoomsema lahenduse leidmine veel ebatõenäolisem, sest jäikusseinad saavad esineda ainult fikseeritud kohtades. Seega oleks optimeerimise algoritmidest kõige rohkem kasu kui neid kasutada juba projekteerimise esmastes staadiumites, samaaegselt arhitektuurse plaani loomisega [4].

1.2 Jäikusseinte analüüsimise ja dimensioneerimise meetodid

Betoonkonstruktsioonide projekteerimist käsitlev Euroopa standard EN-1992-1-1 [5] soovitab raudbetoonist seinte kandevõime määramisel kasutada varrasskeeme, mis tänapäeval luuakse lõplike elementide meetodit kasutavate tarkvarade abil. Loetud

artiklitest kasutavad enamik tööde autoritest jäikusseinte projekteerimiseks lõplike elementide meetodit, kuid mitmetes töödes on otsustatud kasutada analüütilisi meetodeid. Selle aluseks on lihtsustatud surutud raudbetoonist ristlõike tugevuskontroll, mida esmalt kasutas oma töös Zhang [6] ning mida kasutati hiljem mitmetes järgnevates töödes jäikusseinte tugevuskontrollide teostamiseks [7], [8]. Kahe meetodi peamiseks erinevuseks on arvutusteks kuluv aeg ning analüütilised meetodid on olulisemalt kiiremad kui lõpliku elementide meetodid. Näiteks kulus Zhou'l [9] ühe jäikusseinte laotuse arvutuste teostamiseks lõplike elementide meetodiga ligikaudu kaks ja pool minutit. Zhang [6], kes kasutas analüütilisi meetodeid, teostas ühe laotuse arvutused ligikaudu 0,025 sekundiga. Kuigi Zhang'i kasutatud jäikusseinte laotus oli oluliselt lihtsam kui Zhou, siis on arvutusteks kulunud aeg siiski kolme suurusjärgu võrra erinev. Lõplike elementide meetodiga teostatakse arvutused tervele hoonele, mistõttu hoone kõrguse kasvades suureneb oluliselt elementide arv ning seega ka arvutusteks kuluv aeg. Analüütiliste meetodid analüüsivad üldiselt ainult ühte horisontaalset lõiget hoones, mistõttu ei sõltu arvutusteks kuluv aeg hoone kõrgusest. Seega on analüütilisi meetodeid kasutades võimalik oluliselt vähendada arvutusteks kuluvat aega.

Analüütiliste meetodite puuduseks on raskused keerulisemate olukordade arvestamisel. Üheks näiteks sellisest olukorrast on seintes esinevad avad, mille mõju seina jäikusele ei ole võimalik kergelt arvestada. Sellel põhjusel ei lubanud Zhang [6] optimeerimise algoritmil luua jäikusseintest suletud ristlõikeid, sest nende kasutamiseks šahtide või trepikodadena oleks vaja avasid. Lõplike elementide meetodiga on see-eest võimalik arvestada avadega ning ka veel keerulisemate olukordadega näiteks avade kohale paigaldatud terastaladega täiendava jäikuse saamiseks.

Kuna jäikusseinte variantide läbiproovimisel on vaja korduvalt teostada arvutusi ühe hoone jaoks lõi Lou [10] arvutuste ajakulu vähendamiseks lõplike elementide meetodiga sürrogaat mudeli. Antud mudel kasutas varasemalt teostatud lõplike elementide meetodiga tehtud arvutuste tulemusi erinevate jäikusseinte laotuste puhul, et ennustada ligikaudseid tulemusi järgnevatele variantidele. Sürrogaat mudel vajas arvutuste teostamiseks oluliselt vähem aega kui lõplike elementide meetod, mis võimaldas vähendada optimeerimise algoritmil kuluvat aega kuni 44%.

1.3 Jäikusseinte paiknemise optimeerimine

Kandekonstruktsioonide optimeerimise saab üldiselt jagada kolme valdkonda: suurus, kuju ja paiknemine. Üldiselt optimeeritakse neid valdkondi eraldi, kuid kõige parema tulemuse saavutamiseks on vaja arvestada kõiki kolme korraga [4]. Näiteks jäikusseinte puhul tuleks korraga arvestada seinte paksuse, kuju ning nende paiknemisega hoones. Loetud töödest käsitlevad enamik töid ainult jäikusseinte paiknemist ja kuju [1], [2], [6]–[14] ning ainult üksikutes on arvestatud ka seinte paksusega [15]–[17]. Lisaks on ka varasemalt optimeeritud jäikusseinte armatuuri kogust ja paiknemist [17], [18]. Tõenäoliselt keskendutakse Kõige rohkem just jäikusseinte paiknemise uurimisele, sest seal on kõige rohkem võimalik võita materjali kulus. Seinte paiknemisel on äärmiselt palju erinevaid kombinatsioone, mistõttu ei suuda insenerid käsitsi enamik variante läbi proovida ega kõige optimaalsemaid variante leida. Näiteks võrdles Zhou [9] inseneri ja optimeerimise algoritmi poolt loodud jäikusseinte laotusi, kus algoritmi loodud lahendus

kasutas kuni 31% vähem materjali kui inseneri oma. Seega võib jäikusseinte kuju ja paiknemise optimeerimisel olla võit kõige suurem. Kui võtta arvesse veel täiendavaid parameetreid nagu seinte paksus või armatuuri kogus, võib probleem osutuda liiga keeruliseks, et optimeerimise algoritm suudaks anda sobivaid lahendusi mõistliku ajaga.

Optimeerimise meetoditest on enim kasutatud heuristilised, mis kasutavad stohhastilist meetodit erinevate lahenduste uurimiseks [1], [2], [6]–[16]. Selle põhjuseks on, et jäikusseinte laotuste optimeerimiseks vajalikku funktsiooni on keeruline matemaatiliselt väljendada ning neid ei ole üldiselt võimalik diferentseerida. Seetõttu ei sobi üldiselt optimeerimisel kasutatavad gradiendi meetodi põhised algoritmid [10]. Lisaks esineb jäikusseinte laotuste optimeerimisel palju lokaalseid maksimume, ehk lahendusi, mis on parimad oma lähimate naabrite seast, kuid ei pruugi olla parimad kõikidest võimalikest variantidest. Seega võivad paljud optimeerimise algoritmid jõuda esimesse leitud lokaalsesse maksimumi ega oska sealt edasi liikuda. [4]. Mitmed heuristilised algoritmid on aga loodud seda olukorda vältima.

Üks kõige levinumaid heuristilise optimeerimise algoritme ning ka loetud töödes kõige enim kasutatud on evolutsiooniline optimeerimise algoritm. Lisaks evolutsioonilistele algoritmidel on uuritud ka teiste algoritmide efektiivsust jäikusseinte laotuste optimeerimisel nagu *quantum charged system search* [8], *multi objective water cycle algorithm* [7], *tabu search* [10] ja *evolutionary structural optimization* [2]. Jäikusseinte armeeringu koguse ja paiknemise optimeerimisel võrdles Hoseini [18] veel nelja erinevat heuristilist algoritmi: *particle Swarm optimization*, *whale optimization algorithm*, *crow search algorithm*, *firefly algorithm*. Lisaks heuristlistele algoritmidele kasutas Gan [16] täiendavalt teisi algoritme lõpliku tulemuse saamiseks. Esmalt kasutas ta evolutsioonilist algoritmi sobiva jäikusseinte laotuse leidmiseks, peale mida ta dimensioneeris iga seina kasutades meetodit nimega *fully stressed design*.

Peale optimeerimise algoritmide on sobivate jäikusseinte laotuste loomiseks Liao [19] kasutanud masinõppe meetodit *generative adversarial network*. Erinevalt optimeerimise algoritmidest, kus proovitakse läbi erinevaid variante, kasutab masinõppe meetod olemasolevate hoonete jäikusseinte paiknemist, et nende põhjal luua uuele hoone sobiv laotus.

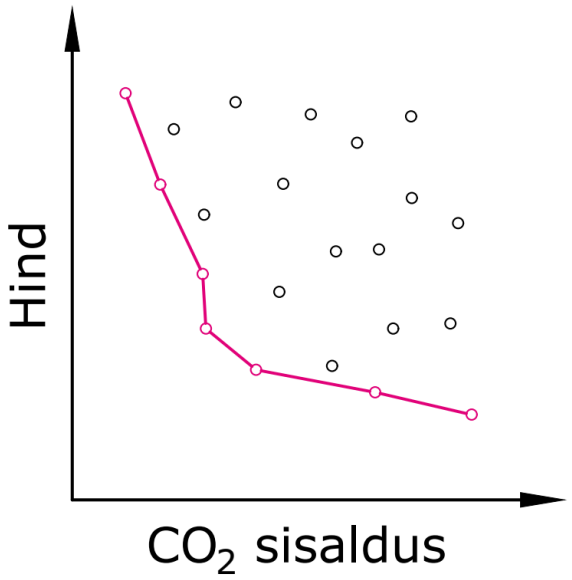
Kuna optimeerimise algoritmide efektiivsuse sõltub väga palju probleemi olemusest ning kuidas see on algoritmile defineeritud, siis ei ole võimalik algoritmide efektiivsust erinevate tööde vahel otseselt võrrelda. Lou [10] võrdles omavahel tavalist evolutsioonilise optimeerimise algoritmi ja tabu otsingu meetodid. Nende saadud lahendused olid sarnased, kuid evolutsiooniline algoritm vajas tulemuse saamiseks 7 korda rohkem iteratsioone. Kuigi selle töö põhjal ei ole evolutsioonilised algoritmid kõige efektiivsemad, siis keskendutakse antud töös detailsemalt evolutsioonilistele algoritmidele, kuna nende kohta leidub kõige rohkem tarkvaralisi lahendusi.

1.4 Jäikusseinte optimeerimine evolutsiooniliste algoritmidega

Evolutsioonilised optimeerimise algoritmid vajavad lahendusteni jõudmiseks väga palju iteratsioone, mistõttu on mõistliku ajakulu säilitamiseks vaja võimalikult palju

ebasobivaid variante juba enne optimeerimist eemaldada. Üheks selliseks viisiks on täpsustada esimesed variandid optimeerimise protsessi alguses, selle asemel et need suvaliselt luua. Sellisel juhul alustab algoritm optimeerimist, mis on lähemal soovitud tulemusele. Zhang [6] on selle jaoks piiranud optimeerimise alguses maksimaalset võimalike seinte arvu. Kuna konstruktsiooni massi suhtes optimaalne lahendus sisaldab endas pigem vähem kui rohkem seinu, siis võimaldab see välistada palju variante, mis ei oleks kindlasti optimaalsed, näiteks kui kogu hoone oleks kaetud jäikusseintega. Zhou [9] lasi esimesed variandid luua algoritmil suvaliselt ning segas iga variandi ühe inseneri poolt loodud lahendusega. Selle tulemusena olid esimesed variandid juba lähemal sobilikele lahendustele, kuid siiski üksteisest oluliselt erinevad. Ühte kõige lihtsamat lahendust on kasutanud Tafraout [14], kes nõudis, et kõik esimesed variandid oleksid ehituskonstruktsioonide tugevusnõuete poolest täidetud, välistades sellega täielikult sobimatud lahendused optimeerimise alguses.

Evolutsioonilised algoritmid kasutavad eesmärgi ja piirangu funktsioone, et hinnata variandi vastavust soovitud tulemusele. Jäikusseinte optimeerimisel on peamiselt kasutatud algoritme, mis käsitlevad ühte eesmärgi funktsiooni, milleks on minimeerida seinte mass või ruumala [15], [16]. Peale materjali kulu on sellega kaudselt seotud kaks teist parameetrit, ehitusmaksumus ning süsiniku jalajälg, mistõttu sobib jäikusseinte mass ka nende omaduste hindamiseks. Erandina teistest töödest kasutab Dehnavipour [7] mitme eesmärgiga optimeerimise algoritmi, minimeerides samaaegselt jäikusseinte massi ja hoones tekkivat väändemomenti. Kaks eesmärki võimaldas kergemini leida minimaalse väändemomendiga jäikusseinte laotusi ning parandas algoritmi toimimist ebasümmeetriliste, näiteks L-kujuliste hoonete puhul. Saadud tulemusi kuvati Pareto kõverana, kus antakse parimad lahendused erinevate eesmärkide tähtsuste puhul. Näide Pareto kõverast on toodud joonisel 1.1.



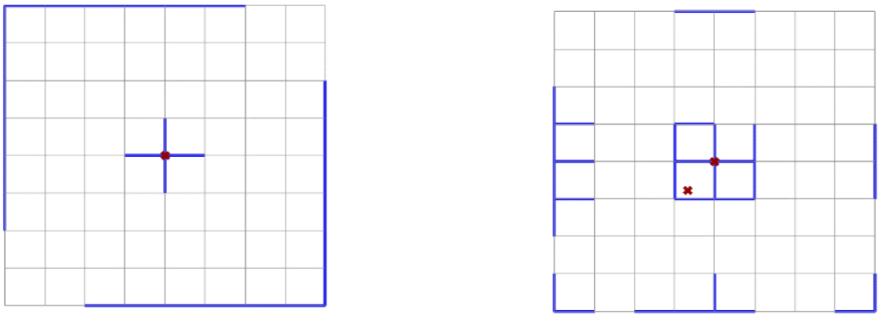
Joonis 1.1 Näide Pareto kõvera tulemustest

Mitut eesmärki on võimalik kajastada ka ühe eesmärgiliga optimeerimises, kasutades kaalutud summade meetodi. Haseeb [13] valis optimeerimise eesmärkideks

konstruktsiooni omakaalu, hinna ja süsiniku jalajälje, mille väärtused korrutati läbi koefitsiendiga vastavalt nende tähtsusele ning liideti kokku. Seega oli optimeerimise algoritmi jaoks tegu ühe eesmärgiga, kuigi see sisaldas endas mitme erineva eesmärgi osasid.

Sarnast meetodit kasutatakse ka piirangute seadmisel optimeerimise algoritmile, mis tagab lahenduste vastavuse tugevuskontrollidele ja teistele soovitud nõuetele. Eesmärgi funktsioonile lisatakse juurde tegurid, mille väärtus sõltub piirangutest ning juhul kui need on ületatud tõuseb teguri väärtus ning seeläbi ka kogu funktsiooni väärtus, muutes selle optimeerimise algoritmi jaoks ebasobivamaks. Näiteks arvestas Zhang [6] niimodi tugevuskontrolle, väändemomendi suurust ning maksimaalset siiret. Juhul kui üks nendest on ületatud, määrati tegurile väga suur väärtus, mistõttu optimeerimise algoritm seda järgmistesse generatsioonidesse ei võtnud.

Lisaks kandekonstruktsioonide tugevuse ja pingetega seotud nõutele kasutatakse kaudsemaid piiranguid, mille eesmärgiks on suunata optimeerimise algoritme sobilikema lahenduste poole. Philps [1] on oma töös kasutanud kahte täiendavat piirangut, mille eesmärk on luua arhitektuurselt sobivaid lahendusi. Esimene suunab algoritmi paigutama jäikusseinu hoone keskele või selle perimeetrisse, soodustades seega inimeste liikumise võimalust ülejäänud hoones. Teine soodustab algoritmi looma rohkem jäikusseinte gruppe, et vältida nende koondumist üheks suureks massiks hoone keskel.



Joonis 1.2 Kaudsete piirangute mõju jäikusseinte laotusele [1]

Tafraout [14] uuris oma töös kortermaja jäikusseinte laotust, kus seinad olid peamised vahelagedelt tuleva vertikaalse koormuse kandjad. Seega oli tähtis seinte ühtlane jaotus hoones, et vertikaalse koormuse vastu võtmiseks oleks vaja võimalikult vähe täiendvaid tarindeid. Selle jaoks lõi ta funktsiooni, mis hindas seinte koormuse mõjualade kattuvust ning eelistas lahendusi, kus see oleks võimalikult väike.

1.5 Kokkuvõte

Jäikusseinte projekteerimise töömahukuse vähendamiseks on varasemates töödes kasutatud peamiselt optimeerimise algoritme, millega saab automaatselt luua sobivaid jäikusseinte laotusi. Kõige enam on levinud ühe eesmärgiga evolutsioonilised optimeerimise algoritmid, mille eesmärgi funktsiooni lisatakse täiendavate teguritena piirangud, et tagada lahenduste ehituskonstruktsiooniline toimivus. Lisaks kasutatakse

täiendavaid piiranguid, et suunata optimeerimise algoritmi soovitud tulemuse poole. Kandekonstruktsioonide analüüs teostatakse nii lõplike elementide meetodiga kui ka analüütiliste meetodiga. Viimasel neist on oluliselt väiksem ajakulu, kuid selle võime toime tulla keerulisemate olukordadega on väiksem. Siiski on analüütilisi meetodeid võimalik kasutada mitte ainult optimeerimise algoritmide puhul, vaid ka tavalisemas arvutustarkvaras, millega oleks inseneril võimalik erinevaid jäikusseinte laotusi läbi proovida.

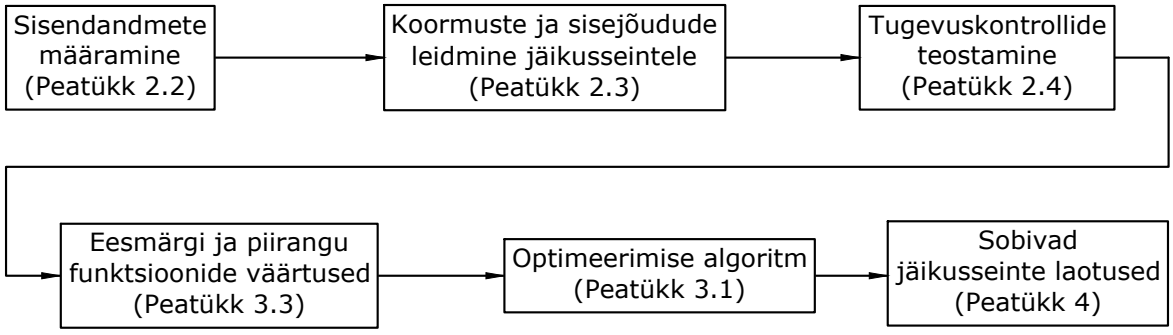
2. JÄIKUSSEINTE ARVUTAMISE PROGRAMM

2.1 Sissejuhatus

Jäikusseinte laotuste loomine on iteratiivne töö, kus on vaja läbi proovida palju erinevaid variante enne kui leitakse sobiv lahendus. Niisugune töö on väga aeganõudev, mistõttu on varasemates uurimistöödes välja pakutud luua jäikusseinte laotused optimeerimise algoritmidega. Samas on jäikusseinte laotuste loomisel üks suurimaid ajakulusid arvutuste teostamine lõplike elementide meetodiga. Seega pakub autor välja läheneda probleemile teisest suunast ning luua esmalt programm, millega oleks võimalik projekteerida jäikusseinte laotusi kiiremini kui saadaval olevate tarkvaradega. Asendades lõplike elementide meetodi analüütilisega, vähendatakse oluliselt ühe variandi arvutusteks kuluvat aega, võimaldades inseneril kiirelt teha muudatusi ning näha nende mõju konstruktsioonile.

Järgnevalt saab kasutada loodud jäikusseinte arvutamise programmi koos evolutsioonilise optimeerimise algoritmiga, et automatiseerida jäikusseinte laotuste loomine. Selleks muudetakse jäikusseinte asukohad hoones parameetriteks, mida optimeerimise algoritm saab muuta. Optimeerimise algoritmid arvestavad peamiselt ehitusinsenerile tähtsate aspektidega laotuste loomisel nagu minimaalne mass ning pinged ja deformatsioonid, mistõttu ei pruugi saadud parim lahendus sobida teistele projekteerimisega seotud osapooltele nagu tellija või arhitekt. Seetõttu peab optimeerimise algoritmi tulemusena saama mitu üksteisest erinevat jäikusseinte laotust, et nende vahel oleks hiljem võimalik valida, arvestades ka optimeerimiste protsessist välja jäänud nõuetega.

Jäikusseinte arvutamise programm luuakse joonestustarkvaraga Rhinoceros 3D ja selle visuaalse programmeerimise lisandiga Grasshopper. Antud tarkvarad on mõeldud geomeetriliste kujunditega töötamiseks, mistõttu see sobib hästi analüütiliste meetodite kasutamiseks jäikussente arvutamiseks. Järgnevates peatükkides on kirjeldatud, millistest osadest programm koosneb ning kuidas see töötab. Programmi peamised osad on toodud joonisel 2.1. Täpsemalt programmi osasid kirjeldavad diagrammid on toodud lisas 1.

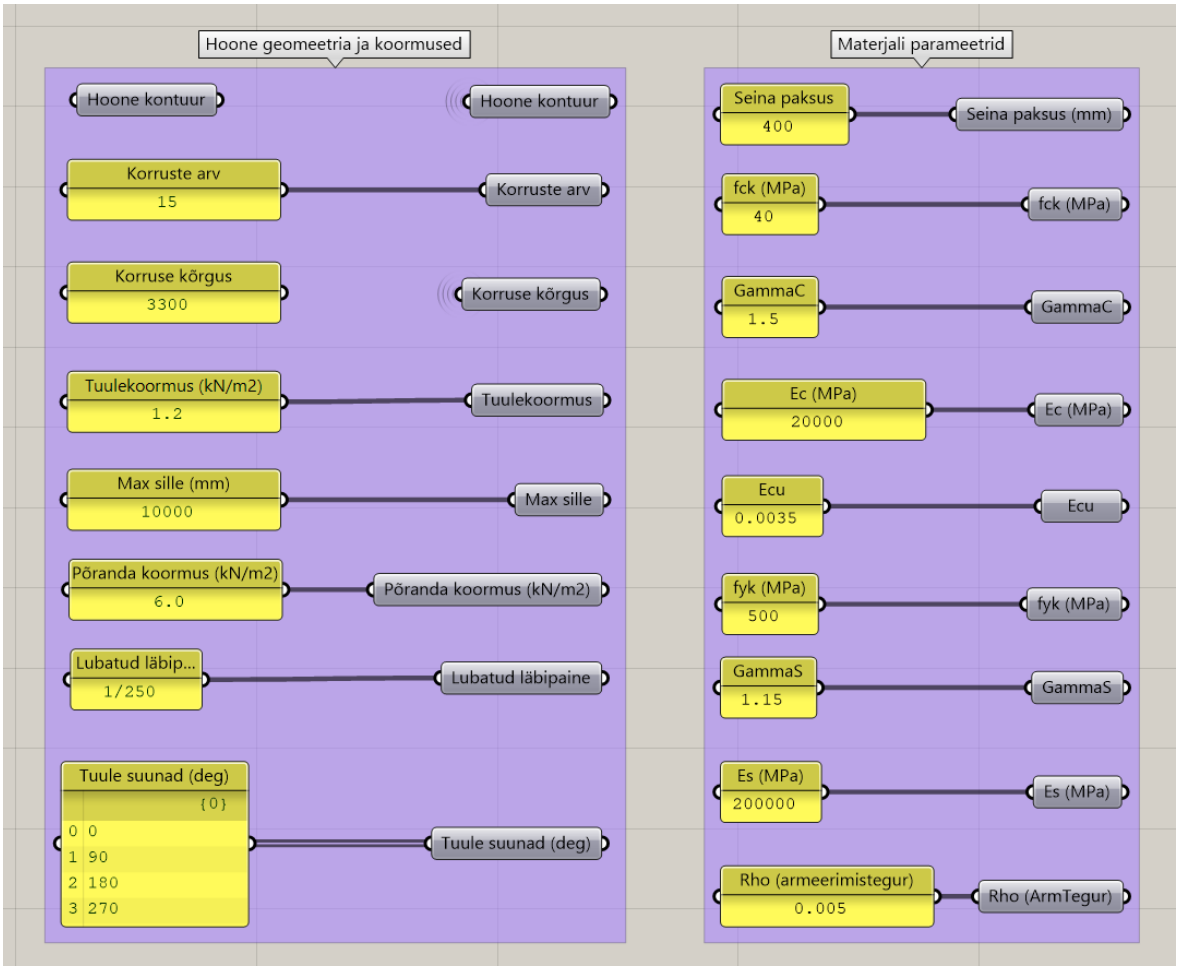


Joonis 2.1 Lähteandmete sisestamine Grasshopperis

2.2 Programmi sisendandmed

Jäikusseinte arvutamise protsess algab kasutaja poolt hoonega seotud olevate andmete sisestamisest. Kasutaja poolt sisestavad andmed Grasshopperis on näidatud joonisel 2.2. Kasutaja poolt on vaja defineerida järgnevad hoonega seotud osad:

- Hoone kuju
- Horisontaal- ja vertikaalkoormused
- Armeerimistegur
- Jäikusseinte mõõtmed ja asukohad
- Materjalide omadused



Joonis 2.2 Lähteandmete sisestamine Grasshopperis

Hoone kuju defineeritakse selle perimeetri, ühe korruse kõrguse ning korruste arvu järgi. Vastavalt sellele loob programm 3D mudeli, mida kasutatakse koormuste määramiseks hoonele ning toereaktsioonide leidmiseks jäikusseintele. Hoone perimeeter defineeritakse Rhino mudelis suletud joonena ning korruste arvu ja ühe korruse kõrguse sisestab kasutaja Grasshopperis vastavatesse lahtritesse.

Horisontaalkoormusena on ette nähtud tuule koormus, mis sisaldab endas nii

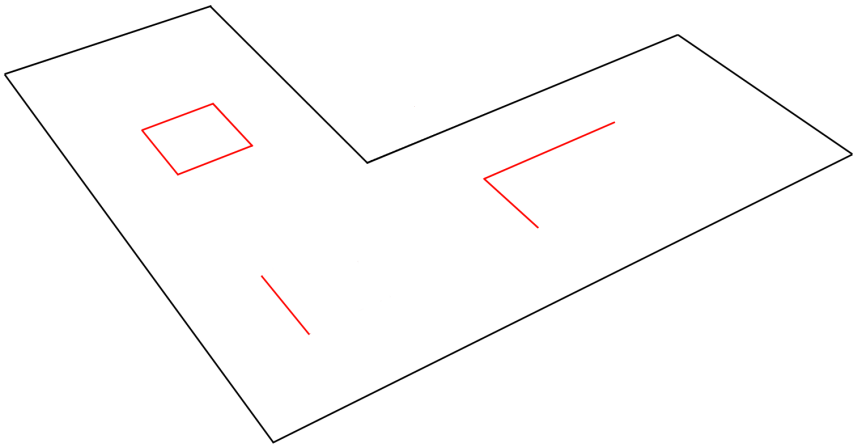
tuulekoormuse suruvat kui imevat komponenti. Antud töös kasutatakse lihtsustusena kogu fassaadil alati ühtlast tuulekoormuse väärtust, kuigi kõrgete hoonete puhul võib olla vaja arvestada muutuva väärtusega vastavalt Eurokoodile EN 1991-1-4 [20]. Vertikaalkoormuseks vahelagedel on ette nähtud ühtlane lauskoormus, mis sisaldab endas nii vahelagede omakaalu kui ka kasuskoormust. Kasutaja poolt sisestatud koormuste väärtused peavad olema juba läbi korrutatud varu- ja kombinatsiooniteguritega ning iseloomustama arvutuslikke väärtusi kasutatavas koormuskombinatsioonis.

Jäikusseinte tugevuskontrolli teostamiseks on vaja teada seintes olevate armatuurvarraste paiknemist ja kogust. Loodud arvutusprogramm ei käsitle täpset armatuuri paiknemist, vaid eeldatakse, et armatuur on seintes ühtlaselt jaotatud. Seetõttu määratakse armatuuri kogus armeerimistegurina, valemiga 2.1. Kuna seinad armeeritakse selle tasapinnas kahes kihis ühtlase sammuga armatuurvarrastega, siis sõltub armatuuri kogus jooksva meetri kohta seina paksusest. Näiteks armeerimisteguri ρ=0,005 ja 400 millimetrise seina paksuse puhul on see ligikaudu võrdne armatuurvarrastega, mille läbimõõt on 16 millimeetrit ja samm 200 millimeetrit. Samas, kui seina paksus on 200 millimeetrit, on armatuurvarraste läbimõõt ligikaudu 10 millimeetrit ja samm 150 millimeetrit. Seega sõltub armatuurikogus seina paksusest, mis vastab ka reaalsuses kasutatavatele seintele. Mida paksem on sein, seda rohkem armatuuri selles üldiselt on.

$$\rho = \frac{A_s}{A_c} \tag{2.1}$$

kus
A_s - Armatuurvarraste pindala ristlõikes
A_c - Betooni ristlõike pindala

Jäikusseinte paksuse väärtus sisetatakse Grasshopperis arvuna ning seinte asukohad määratakse kasutaja poolt Rhino mudelis, kasutades selle jaoks jooni hoone perimeetri sees. Joonis 2.3 kujutab jäikusseinte asukohtade määramist. Juhul kui kaks joont lõikuvad omavahel, siis arvestatakse need ühe seinana, mis moodustab ühe keerulisema kuju. Analoogselt talade ja postidena kasutatavatele I-, L-, T- ja ristkülikukujulistele ristlõigetele on sellised jäikusseinad oluliselt suurema inertsimomendiga kui neid moodustavate üksikute seinte jäikuste summa. Seega on jäikussüsteemi pingete ja deformatsioonide suhtes keerulisemate kujudega seinte esinemine soodne. Peale seinte asukohtade määramist teostab programm arvutused, peale mida võib kasutaja seinte paiknemist uuesti muuta ning näha missugune mõju sellel on seintes esinevatele sisejõududele ja nende kandevõimetele.

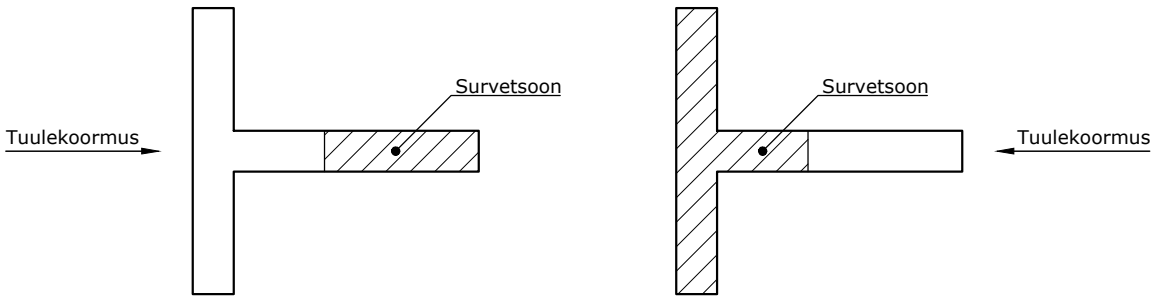


Joonis 2.3 Jäikusseinte asukohtade määramine Rhinos
punased jooned - jäikusseinad, must joon - hoone perimeeter

2.3 Koormuste jaotus ja toereaktsioonide leidmine

2.3.1 Horisontaalkoormuste jaotus

Horisontaalse koormusena on ette nähtud tuulekoormus, mis võib mõjuda hoonele neljalt küljelt, paralleelset Rhino mudeli X- või Y-teljega. Kõik arvutused teostatakse neli korda, kus iga kord mõjub tuulekoormus erinevas suunas. Ühe teljega paralleelselt tuleb teha arvutused kahes suunas, sest jäikusseinad võivad olla ebasümmeetriliste kujudega. Erineva tuule suuna puhul tekivad jäikusseintes survetsoonid eri ristlõike pooltele, mistõttu võivad nende suurused ja seega ka kandevõimed erineda. Nimetatud olukorda on illustreeritud joonisel 2.4.



Joonis 2.4 Survetsooni erinevus sõltuvalt tuule suunast

Tuulekoormus projekteeritakse hoone fassaadile, vastavalt mille pindaladele ja paiknemisele leitakse tuulekoormuse resultantjõud. Jäikusseinu vaadeldakse arvutuslikult kui konsoolseid poste, millele mõjub horisontaalne koormus. See jaotatakse jäikusseinte vahel lõpmatult jäiga vahelae põhimõttel, mistõttu jaguneb koormus proportsionaalselt üksikute seinte jäikustele. Seina jäikus on defineeritud selle tipu

siirde pöördväärtusena teatud koormuse suhtes. Seega koormates seinu võrdse jõuga, võtab kõige väiksema siirdega sein vastu kõige suurema osa tuulekoormusest. Kuna seina tipu siire sõltub peamiselt paindejäikusest ja ainult väga väiksel määral põikjõu jäikusest, siis saab lihtsustusena horisontaalkoormused jaotada seinte vahel vastavalt nende inertsimomentidele. Kui jäikussein on sirge joone kujuga ning tuule suund on sellega risti, siis ei arvestata seina horisontaalkoormuse jaotamisel, sest seina jäikus oleks liiga väike, et praktiliselt töötada.

Lisaks seinte enda jäikustele tuleb arvestada ka horisontaalkoormuse resultantjõu ja jäikustsentri erinevusest tingitud väändemomendiga. Mida suurem on nende erinevus, seda suurem on ka väändemoment. Vääne võetakse jäikusseintes vastu täiendava horisontaalse jõuna, mille suurus sõltub seina kaugusest jäikustsentrist. Seintes tekkivate horisontaalsete toereaktsioonide leidmiseks vajalikud tehted on toodud valemites 2.2 kuni 2.8. Toereaktsioonide leidmiseks vajalikke suurusi illustreerib joonis 2.5.

$$F_{xi} = I_{yi} \cdot (\frac{q_x}{I_y} - (y_i - y_j) \cdot \frac{M_q}{I_z}) \tag{2.2}$$

$$F_{yi} = I_{xi} \cdot (\frac{q_y}{I_x} + (x_i - x_j) \cdot \frac{M_q}{I_z}) \tag{2.3}$$

kus
F_i - horisontaalse reaktsiooni suurus jäikusseinas
I_i - jäikusseina inertsimoment
q - horisontaalkoormus
I - kõikide jäikusseinte inertsimomentide summa
x_i/y_i - jäikusseina raskuskeskme asukoht
x_j/y_j - jäikustsentri asukoht
M_q - väändemoment
I_z - jäikusseinte väändetugevus.

Jäikustsenter leitakse järgnevate valemitega:

$$x_j = \frac{\sum_{i=1}^{i=n} x_i I_{xi}}{I_x} \tag{2.4}$$

$$y_j = \frac{\sum_{i=1}^{i=n} y_i I_{yi}}{I_y} \tag{2.5}$$

Väändemoment leitakse järgneva valemiga:

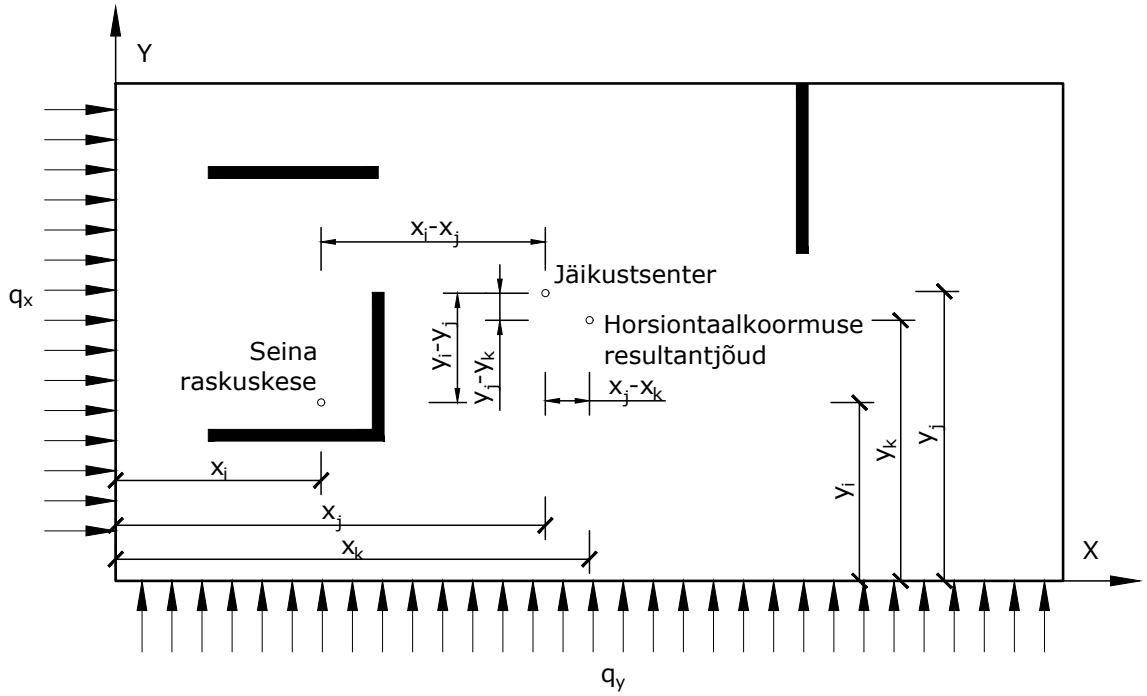
$$M_{qx} = q_x \cdot (y_k - y_j) \tag{2.6}$$

$$M_{qy} = q_y \cdot (x_k - x_j) \tag{2.7}$$

kus
x_k - horisontaalkoormuse resultantjõu kaugus x-teljest
y_k - horisontaalkoormuse resultantjõu kaugus y-teljest

Väändetugevus leitakse järgneva valemiga:

$$I_z = \sum_{i=1}^{i=n} [I_{xi} \cdot (y_i - y_j)^2 + I_{yi} \cdot (x_y - y_j)^2] \tag{2.8}$$

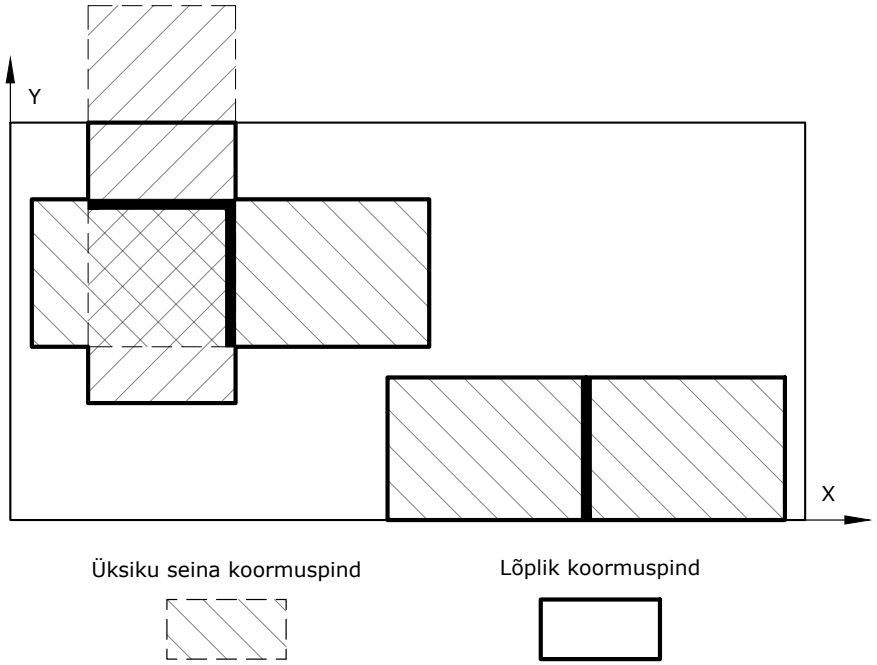


Joonis 2.5 Jäikusseinte sisejõude leidmisel kasutatavad muutujad

2.3.2 Vertikaalse koormuse jaotus

Seintele mõjuva koormuspinna leidmisel arvestatakse, et vahelagi jaotab koormust mõlemas suunas, mistõttu on kõik seinad koormatud olenemata nende suunast. Koormuspinna suuruse defineerib kasutaja maksimaalse silde pikkusena jäikusseina ja teiste vertikaalkoormust kandvate elementide vahel. Programm loob igale üksikule sirgele seinale maksimaalse suurusega koormuspinna ning summeerib need ühe terve seina ulatuses. Juhul kui koormuspind ulatub hoone perimeetrist väljapoole, siis selle

osa pindala ei arvestata. Koormuspindade leidmist illustreerib joonis 2.6. Seinale mõjuv vertikaalkoormuse väärtus leitakse koormuspinna pindala ja kasutaja poolt varasemalt defineeritud vertikaalse koormuse suuruse korrutisena.



Joonis 2.6 Koormuspindade leidmine

2.3.3 Sisejõudude leidmine

Jäikusseintes tekkivad sisejõud leitakse hoone esimesel korrusel, kus nende väärtus on kõige suurem ning tugevuskontroll seega kõige kriitilisem. Jäikusseinad käituvad kui konsoolsed postid, milles tuulekoormus tekitab põikjõu ja paindemomendi. Normaaljõud võetakse võrdseks seintele leitud vertikaalsete koormuste väärtusega. Antud töös ei arvestada seinte ebaühtlase koormamise tõttu seina tasapinnast välja tekkivat paindemomenti ega teist järku mõjusid.

2.4 Tugevuskontrollid

Jäikusseinte laotust mõjutavad piirangud, mis peavad olema sobivates lahendustes täidetud. Nendeks on tugevuskontrollid, mis tagavad konstruktsiooni toimivuse ning maksimaalse siirde kontroll, mida piiratakse kõrgetes hoonetes elanike mugavuse tõttu. Lihtsuse huvides, mõeldakse edaspidiselt tugevuskontrollide all nii tugevuskontrolle ise kui ka maksimaalse siirde kontrolli- Järgnevalt on kirjeldatud, kuidas tugevuskontrollid on teostatud.

2.4.1 Maksimaalse siirde kontroll

Hoone maksimaalne lubatud siire määratakse kasutaja poolt, kuid vaikimisi võetakse selle suuruseks 1/250 hoone kogu kõrgusest. Jäiga vahelae põhimõttel on kõikide

jäikusseinte siirded võrdsed, mistõttu on ühe seina siire võrdne ka kogu hoone siirdega. Seega leitakse läbipaine ühele jäikusseinale vastavalt sellele mõjuvale horisontaalkoormusele. Läbipainde arvutamisel käsitletakse hoonet kui püstist konsooli ning arvestatakse nii paindest kui ka põikjõust tekkiva deformatsiooniga. Hoone läbipaine leitakse valemiga 2.9.

$$\delta = \frac{q_i \cdot H^3}{E \cdot I_i \cdot 8} + \frac{q_i \cdot H}{E \cdot A_i \cdot 2/3} \tag{2.9}$$

kus
δ - hoone läbipaine
q_i - horisontaalkoormuse suurus ühel seinal
H - hoone kõrgus
E - betooni elastsusmoodul
I_i - jäikusseina inertsimoment
A_i - jäikusseina pindala

2.4.2 Põikjõu kontroll

Jäikusseinte põikjõu kontroll tehakse vastavalt betoonkonstruktsioonide Eurokoodis EN 1992-1-1 [5] toodud meetodile. Lihtsuse eesmärgil käsitletakse seinu kui arvutusliku põikarmatuurita elemente ning leitakse nende minimaalne põikjõu kandevõime valemiga 2.10. Antud valemis on Eurokoodi järgselt ristlõike kõrguse h asemel kasuskõrgus d, kuid kuna antud töös ei ole jäikusseintel kasuskõrgust määratud, kasutatakse selle asemel kogu ristlõike kõrgust. Jäikusseinte suurte mõõtmete tõttu on ristlõike kogu kõrguse ja kasuskõrguse erinevus minimaalne.

$$V_{Rd,c,min} = (v_{min} + 0,15\sigma_{cp})b_w h \tag{2.10}$$

$$v_{min} = 0,0035k^{3/2}f_{ck}^{1/2} \tag{2.11}$$

$$k = 1 + \sqrt{\frac{200}{h}} \leq 2,0 \tag{2.12}$$

kus
$V_{Rd,c,min}$ - Armeerimata ristlõike minimaalne põikjõukandevõime
v_{min} - Tegur põikjõu kandevõime määramiseks
σ_{cp} - Ristlõikele mõjuv normaaljõud
b_w - Ristlõike minimaalne laius
h - ristlõike kõrgus
k - Tegur põikjõu kandevõime määramiseks
f_{ck} - Betooni normatiivne tugevus

2.4.3 Surutud ristlõike kontroll

Üksikute sirgete jäikusseinte tugevuskontrollide teostamiseks saab varrasskeemide asemel kasutada lihtsustatud arvutusmeetodeid, kus kontrollitakse seinte otstes olevaid kõige suurema pingega lõike [21]. Kuna antud töös ei esine ainult sirgeid jäikusseinu, vaid ka keerulisemaid ristlõikeid, siis nimetatud meetod ei sobi. Seetõttu lähtutakse tugevuskontrolli teostamisel Zhang'i [6] poolt välja töötatud lihtsustatud arvutusmeetodist, mis võimaldab analüüsida ka keerulisemaid ristlõikeid.

Kasutatav lihtsustatud meetod käsitleb tervet ristlõiget korraga, kus arvestatakse järgmiste eeldustega:

- Üksikud armatuurvardad asendatakse ühtlase pinnana ning armatuuri koguse määrab armeerimistegur;
- Kandevõime arvutustes arvestatakse ainult armatuuri, mille pinge on suurem kui voolepiir;
- Ristlõige on pragunenud, mistõttu kasutatakse betoonis ristkülikukujulist pingejaotust;
- Betooni maksimaalne deformatsioon on ϵ_{cu3}=0,0035.

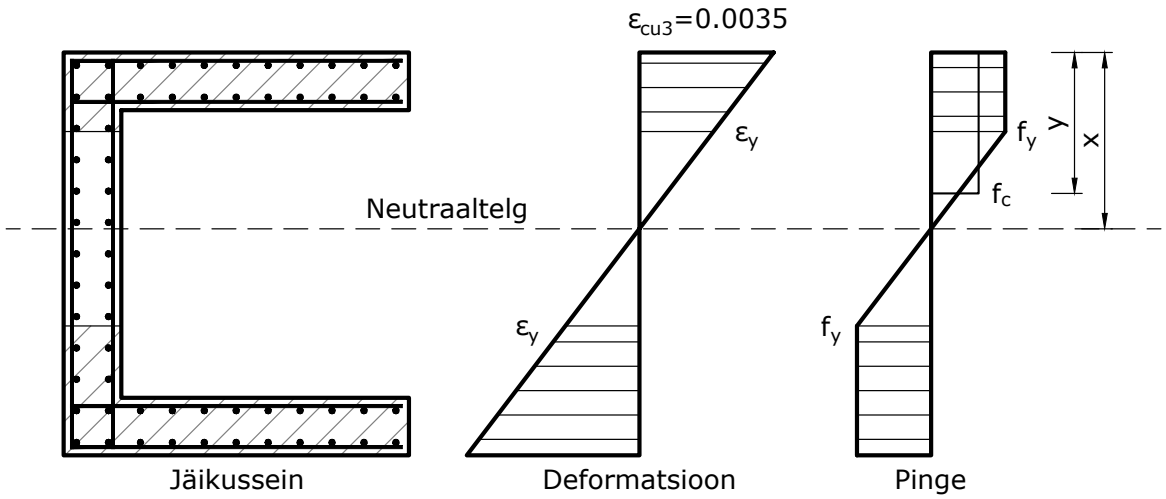
Ristlõike normaal- ja paindekandevõime arvutamiseks valitakse neutraaltelje asukoht ning koos lineaarse deformatsiooni diagrammiga on võimalik leida voolava armatuuriga ristlõike pinnad ja survetsooni pind. Joonisel 2.7 on vastavad alad tähistatud viirutusega. Vastavalt ristlõike voolava armatuuriga pindadele ja survetsooni pinnale on võimalik leida normaaljõu- ning paindekandevõime. Ristlõike tugevus leitakse ümber ristlõike plastse raskuskeskme ning armeeringu jõuõlgadeks võetakse voolava armeeringuga pindala raskuskeskme kaugus ristlõike plastsest raskuskeskmest. Normaaljõu- ja paindekandevõime leitakse valemitega 2.13 ja 2.14.

$$N_{Rd} = F_{cc} + F_{sc} + F_{st} = A_{cc} \cdot f_{cd} + A_{sc} \cdot f_{ycd} + A_{ss} \cdot f_{yd} \tag{2.13}$$

$$M_{Rd} = F_{cc} \cdot (y/2 - h_{geo}) + F_{sc} \cdot (h_{geo} - h_{sc}) + F_{ss} \cdot (h_{geo} - h_{ss}) \tag{2.14}$$

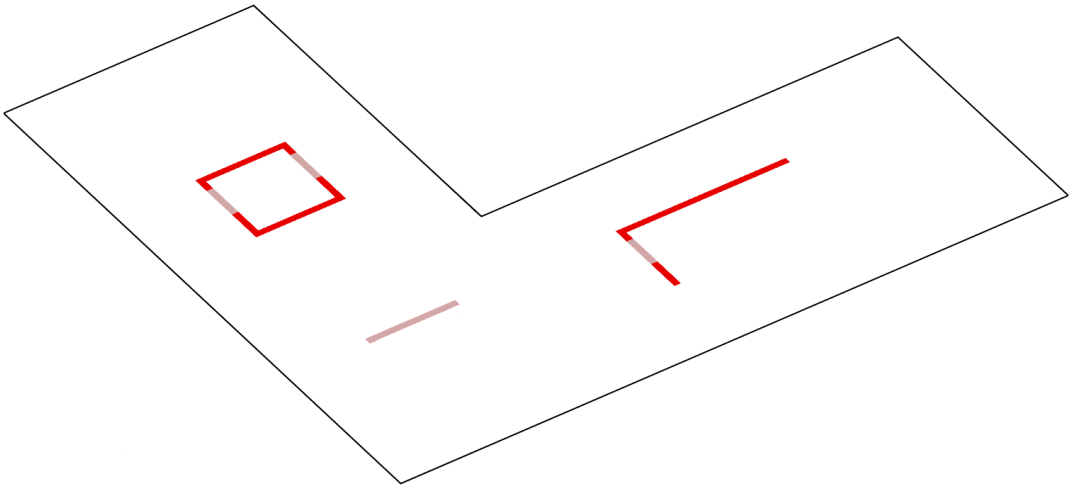
kus
N_{Rd} - ristlõike normaaljõu kandevõime
M_{Rd} - ristlõike paindekandevõime
F_{cc} - jõud betooni survetsoonis
F_{sc} - jõud surutud armatuuris
F_{st} - jõud tõmmatud armatuuris
A_{cc} - surutud betooni pindala
A_{sc} - surutud armatuuri pindala
A_{ss} - tõmmatud armatuuri pindala
f_{cd} - betooni arvutuslik survetugevus

f_{yd} - armatuuri arvutuslik voolepiir tõmbel
f_{ycd} - armatuuri arvutuslik voolepiir survel
y - betooni survetsooni arvutuslik kõrgus
h_{geo} - ristlõike plastse raskuskeskme kaugus
h_{sc} - surutud voolava armatuuri raskuskeskme kaugus
h_{ss} - tõmmatud voolava armatuuri raskuskeskme kaugus



Joonis 2.7 Surutud ristlõike kontroll

Tugevuskontrolli läbiviimiseks võetakse ristlõikel kümme erinevat neutraaltelje asukohta, millest esimene asetseb ristlõike ülemises servas ning viimane alumises servas. Esimene käsitleb olukorda, kus kogu ristlõige on tõmmatud ning viimane, kus kogu ristlõige on surutud. Täiendavalt võetakse veel üks neutraaltelje asukoht, kus selle kaugus ülemisest servast on lõpmata kaugel, mis juhul on terves ristlõikes ühtlane survepinge. Antud normaaltelje asukohtadele leitakse ristlõike normaaljõu- ning paindekandevõimed, mille alusel luuakse normaallõike kandevõime diagrammid. Joonisel 2.8 on toodud näide jäikusseintest, milles on voolava armatuuriga alad. Joonisel on toodud üks sirge joone kujuline jäikussein, millel puuduvad voolava armatuuriga alad. Nagu varasemalt mainitud on põhjuseks, et antud jäikusseina ei arvestata kui tuule suund on sellega risti.



Joonis 2.8 Jäikusseinte ristlõigete surve- ja voolava armatuuriga tsoonid
punane - voolava armatuuriga alaga, helepunane - mitte voolava armatuuriga ala

Tugevuskontrolli teostamisel võrreldakse esmalt mõjuvat normaaljõudu ja leitakse selle asukoht kandevõime diagrammil. Vastavalt normaaljõule leitakse ristlõike paindekandevõime ning võrreldakse seda mõjuva paindemomendiga. Juhul kui paindekandevõime jääb diagrammil kahe eelnevalt leitud paindekandevõime vahele, siis tulemus interpoleeritakse. Tugevuskontrolli tulemus antakse kasutajale protsentuaalselt, ehk kui suur osa kandevõimest on kasutatud.

2.5 Kokkuvõte

Loodud jäikusseinte arvutamise programm võimaldab inseneril kiiresti proovida läbi erinevaid jäikusseinte laotusi ning leida, millised neist võiksid sobivad olla. Tänu automaatsele koormuste määramisele ja lihtsustatud tugevuskontrollidele saab programm teostada arvutused paari sekundiga, mis võimaldab kasutajal teha väga palju muudatusi lühikese ajaga ning peaaegu koheselt näha nende mõju.

3. JÄIKUSSEINTE LAOTUSTE GENEREERIMINE

Antud peatükis kasutatakse eelnevalt loodud jäikusseinte arvutamise programmi, et automatiseerida jäikusseinte laotuste loomine. Selle jaoks kasutatakse evolutsioonilist optimeerimise algoritmi, mis proovib läbi suure koguse erinevaid laotusi ning leiab neist parimad.

3.1 Optimeerimise algoritm

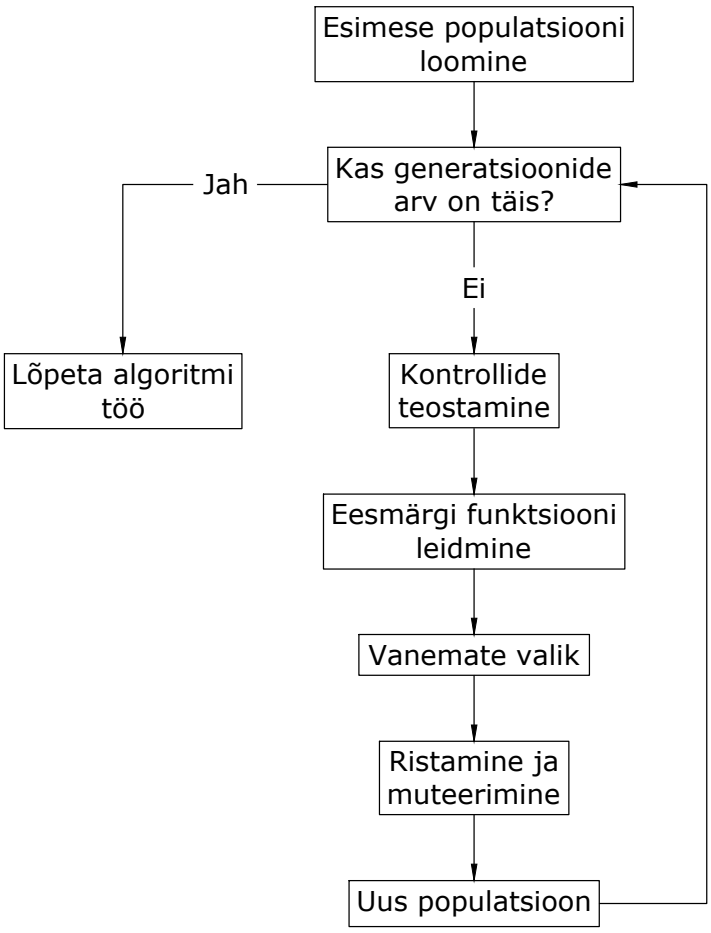
Antud töös kasutati jäikusseinte laotuste genereerimiseks evolutsioonilist optimeerimise algoritmi. Algoritmi üheks eeliseks on, et selle kasutamiseks ei pea olema optimeeritav funktsioon diferentseeritav. See võimaldab algoritmi kasutada ka väga keeruliste probleemide puhul. Loodud jäikusseinte programmis peab optimeerimise algoritm arvestama nii seinte paiknemise kui ka tugevuskontrollidega, mistõttu oleks ühe olukorda kirjeldava funktsiooni loomine ja selle tuletise leidmine äärmiselt raske kui mitte võimatu. Evolutsioonilised optimeerimise algoritmid vajavad see-eest ainult funktsiooni, mis kirjeldab jäikusseinte laotuse vastavust soovitud tulemusele, mida nimetatakse eesmärgi funktsiooniks. Lisades peatükis 2 kirjeldatud jäikusseinte arvutamise programmile laotuse sobivust kirjeldava funktsiooni, saab seda kasutada optimeerimise algoritmis ilma täiendavate muudatuste tegemiseta.

Evolutsioonilised optimeerijad põhinevad stohhastilisel meetodil, kus kasutatakse suvaliselt loodud variante, et jõuda optimaalse lahenduseni. Algoritm töötab sammude ehk generatsioonide kaupa, kus igas generatsioonis luuakse kogumik erinevaid variante, ehk populatsiooni. Kõigile populatsiooni liikmetele määratakse nende vastavus soovitud tulemusele, kasutades selleks eesmärgi funktsiooni, mille tulemust väljendatakse numbrilise väärtusega. Käesolevas töös vastab madalam number paremale laotusele. Parimaid variante kasutatakse järgmise generatsiooni loomisel, peale mida kordub protsess uuesti ning iga generatsiooniga koondub algoritm paremate tulemuste suunas.

Uute variantide loomiseks kasutatakse ristamist, kus kaks varianti segatakse omavahel, saades uue. Mida rohkem vastab variant eesmärgi funktsioonile, seda suurem on tõenäosus, et see valitakse ristamiseks, tagades variantide paranemise iga generatsiooniga. Lisaks ristamisele kasutatakse ka muteerumist, mille puhul võetakse üks variant ning muudetakse osa sellest suvaliselt. Selle eesmärgiks on tõsta variantide mitmekesisust, et uurida erinevamaid variante ning vältida algoritmi koondumist ühte lokaalsesse maksimumi. Algoritm võib töötada igavesti, kuid üldiselt koondub see lõpuks kas ühe või mitme variandini, millest järgmiste generatsioonide loomisel tulemused enam edasi ei parane.

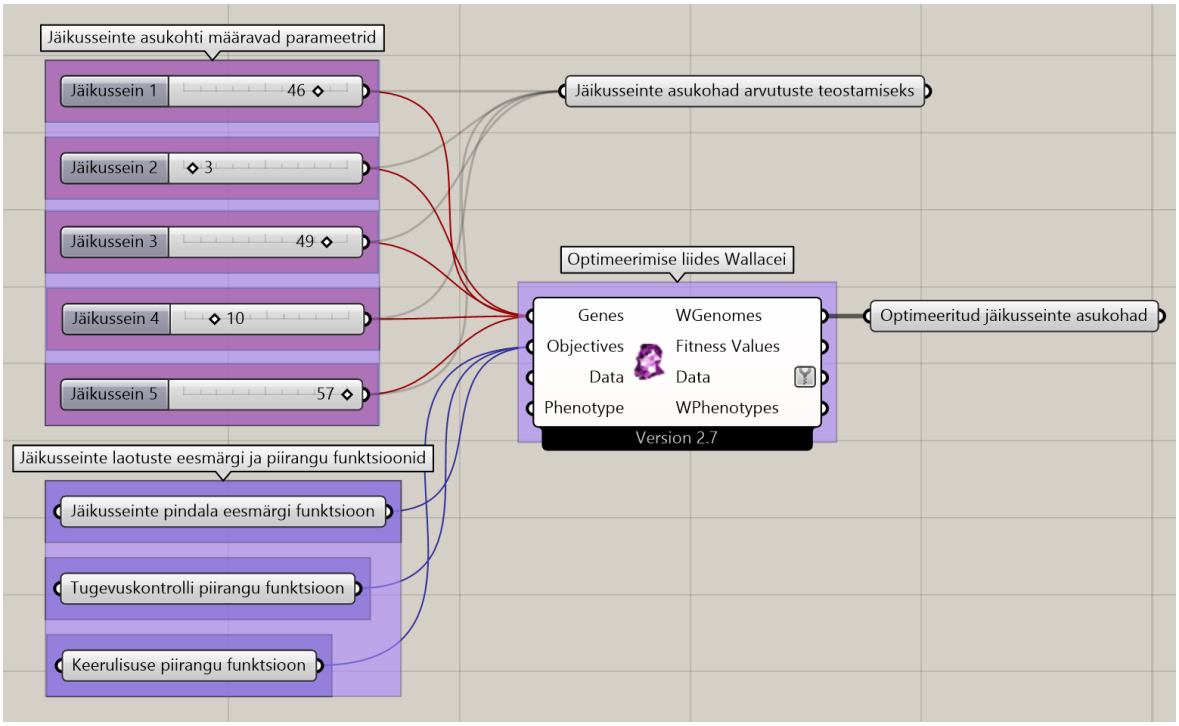
Joonisel 3.1 on toodud diagramm optimeerimise algoritmi töö protsessist. Esmalt loob algoritm esimese generatsiooni populatsiooni, kus iga variant populatsioonis kujutab endast ühte jäikusseinte laotust. Järgnevalt teostatakse igale laotusele tugevuskontrollid ning leitakse nende eesmärgi funktsiooni väärtus. Seejärel valib algoritm välja väiksema

eesmärgi funktsiooniga variandid, millest luuakse ristamise ja muteerumisega uus generatsioon. Protsess kordub kuni on täidetud ette antud fikseeritud arv generatsioone.



Joonis 3.1 Optimeerimise algoritmi tööpõhimõte

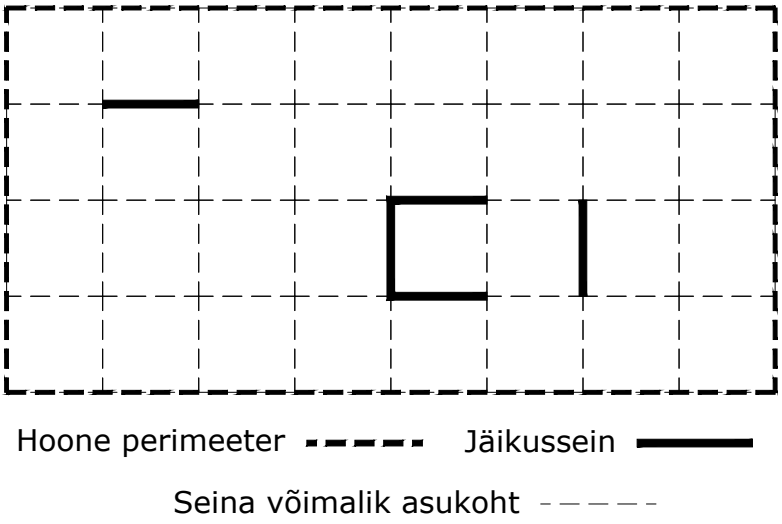
Antud töös kasutatakse jäikusseinte laotuste loomiseks Rhino Grasshopperi jaoks loodud täiendavat tarkvara nimega Wallacei [22]. See on evolutsiooniline optimeerija, mis kasutab ühte enim levinud algoritmi NGSA-II [23]. Üheks algoritmi eristavaks omaduseks on, et see suudab korraga optimeerida mitut eesmärgi funktsiooni. Grasshopperi keskkonnas töötab Wallacei järgnevalt. Algoritmile antakse parameetrid, millega saab muuta optimeeritavat objekti. Antud juhul on parameetriteks numbrid, millest igaüks tähistab ühe jäikusseina asukohta hoones. Kui algoritm valib antud numbri, siis seal kohas eksisteerib jäikussein. Samuti antakse algoritmile iga variandi eesmärgi funktsioonide väärtused. Teades eesmärgi funktsiooni väärtusi ja millistele parameetritele need vastavad, proovib algoritm läbi erinevaid variante ning liigub paremate suunas. Joonisel 3.2 on toodud pilt Wallacei komponendist Grasshopperis ning selle sisendeid ja väljundeid, mida töös kasutatakse.



Joonis 3.2 Wallacei ja selle sisendid ning väljundid

3.2 Jäikusseinte paiknemine

Optimeerimise algoritmi jaoks on vaja määrata, kus kohas võivad jäikusseinad paikneda. Sarnaselt varasemalt tehtud töödele jaotatakse ka antud töös hoone põhiplaan ruudustikuks, mille iga serv on potentsiaalne jäikussein. Ruudustik on võimalik luua automaatselt, täpsustades mitmest ruudust peaks ligikaudu põhiplaan koosnema. Keerulisemate plaanide puhul ei pruugi automaatselt loodud ruudustik ühtida kasutaja soovidega, mistõttu on võimalus ka ruudustiku küljed määrata käsitsi, joonestades need Rhino keskkkonnas. Näide hoone põhiplaanist, mis on jaotatud ruudustikuks, on toodud joonisel 3.3. Võrreldes olukorraga, kus jäikusseinad võivad asuda hoone põhiplaanil igas kohas on ruudustiku kasutamisel nii eeliseid kui puudusi. Ruudustiku kasutamisel on saadud jäikusseinte laotused ligikaudselt optimaalsed, kuna on ebatõenäoline et parimas laotuses kattuksid seinte asukohad täpselt ruudustikuga. See-eest esineb ruudustiku kasutamisel optimeerimisel rohkem omavahel lõikuvaid üksikud jäikusseinu, mille tulemusena tekib keerulisem ristlõige. Kuna evolutsiooniliste optimeerimise algoritmide kasutamine vajab juba niigi väga paljude variantide läbi proovimist, siis võib ilma ruudustikuta kuluda algoritmil ebapraktiliselt palju aega sobivate lahenduste leidmiseks.



Joonis 3.3 Näide hoone jaotamisest ruudustikuks

Ruudustiku loomisel tuleb arvestada selle mõjuga algoritmil vaja minevale ajale. Tiheda ruudustiku puhul on algoritmil rohkem võimalusi, mistõttu võidakse saada täpsemaid tulemusi, kuid see-eest kasvab võimalike laotuste arv. Võimalike jäikusseinte laotuste arvu kirjeldab valem x^n, kus x on iga asukoha võimalik väärtus ning n on asukohtade arv. Näiteks kui seinte asukohtadel on kaks võimalust, kas seal eksisteerib sein või mitte ning hoones on 30 seina asukohta, on võimalike kombinatsioonide arv $2^{30}=1{,}07 \cdot 10^9$. Samas kui seinte asukohtade arv on 40, siis on võimalike kombinatsioonide arv juba $2^{40}=1{,}10 \cdot 10^{12}$, mis on ligikaudu 1000 korda rohkem. Rohkemate võimaluste korral pikeneb optimeerimise algoritmil sobivate lahenduste leidmiseks kuluv aeg. Samuti tekib tiheda võrgustikuga oht, et algoritm loob väga keerulisi lahendusi, mis ei ole päris elus praktilised. Näiteks kui võrgustiku samm on viis meetrit saab algoritm paigutada ühe viie meetri pikkuse seina, kuid kui võrgustiku samm on 2,5 meetrit, siis on vaja samasuguse seina jaoks paigutada kaks seina järjestikku ühele joonele. Samal ajal esineb ka võimalus, et algoritm paigutab teise seina esimesega risti ning saadakse L-kujuline sein. Väga hõreda ruudustikuga ei pruugi see-eest saadud lahendused olla piisavalt detailsed, et sarnaneda päris elus kasutatavatele jäikusseinte laotustele. Seega tuleb leida tasakaal hõreda ja tiheda võrgustiku vahel. Autori arvates peaks ruudustiku ühe külje pikkus olema ligikaudu võrdne ühe üksiku sirge seina pikkusega, mis võiks hoones esineda.

Wallacei kasutab seinte asukohtade valmimiseks arvulisi parameetreid, mille iga väärtus vastab ühele jäikusseina asukohale. Grasshopperi omapärade tõttu ei ole võimalik täpsustada, et kõik väärtused võivad esineda ainult üks kord. Seetõttu võib tekkida olukord, kus kaks seina asetsevad ühes kohas. Selle vältimiseks, määratakse kahe võrdse parameetri korral ühe seina asukohaks lähim vaba seina asukoht.

3.3 Eesmärgi ja piirangu funktsioonid

Sobilikud jäikusseinte laotused peavad tagama hoone kandekonstruktsiooni toimivuse, kuid kasutama samal ajal võimalikult vähe materjali. Antud töös on seetõttu peamiseks

eesmärgi funktsiooniks võetud jäikusseinte põhja pindala minimeerimine, mis on otseselt seotud seinte ruumalaga. Lisaks eesmärgi funktsioonile lisatakse piirangu funktsioonid, mille eesmärgiks on tagada lahenduste sobivus. Ilma piiranguteta oleks kõige optimaalsem lahendus ilma ühegi jäikusseinata hoone, sest sellisel juhul on nende põhjapindala võrdne nulliga. Sellisel laotusel ei ole aga praktilist väärtust. Piirangu funktsioonid kirjeldavad, missugused tingimused peavad lahenduses täidetud olema. Kui need ei ole täidetud, siis loetakse lahendus ebasobivaks.

Enamik optimeerimise algoritme suudavad käsitleda ainult ühte eesmärki, mistõttu tuleb eesmärgi ja piirangu funktsioonid kokku panna ning esitada ühe funktsioonina. Selle jaoks kasutatakse penalteid, ehk kui piirang on ületatud, siis suurendatakse funktsiooni väärtust teatud suuruse võrra, et muuta see optimeerimise algoritmi jaoks ebasobivamaks. Meetodi raske osa seisneb penaltite suuruste määramisel, mis tuleb üldiselt leida katse eksitus meetodil [24]. Lisaks võivad penaltite sobivad suurused erineda ülesande olemusest, ehk antud töö puhul hoone kujust. Kuna penalti suurused mõjutavad oluliselt algoritmi toimivust ning nende määramine ei ole täpne töö, siis kasutatakse käesolevas töös optimeerimise algoritmi, mis suudab käsitleda mitut eesmärki korraga. Kuigi algoritmi otsene eesmärk on minimeerida seinte pindala, siis saab ühe eesmärgiga optimeerimise muuta mitme eesmärgiga optimeerimiseks, lisades piirangu funktsioonid iseseisvateks eesmärkideks, vältides sellega penalti suuruste määramist [25].

Antud töös kasutatakse järgnevaid eesmärgi ja piirangu funktsioone, mida algoritm proovib minimeerida:

- f_s - Jäikusseinte pindala
- f_p - Tugevuskontrollidega seotud piirangud
- f_k - Jäikusseinte keerulisuse piirang

Nagu varasemalt mainitud on peamine eesmärgi funktsioon jäikusseinte pindala, mis on otseselt seotud nende ruumala ja massiga. Selle minimeerimisega leitakse materjali kasutuse poolest kõige ökonoomsem lahendus. Tugevuskontrollide piirang tagab hoone konstruktsiooni vastavuse nõuetele. Eraldiseisva täiendava piiranguna on lisatud veel jäikusseinte keerulisust kirjeldav funktsioon, mille eesmärgiks on vältida üleliigselt keeruliste jäikusseinte tekkimist. Seega kasutatakse optimeermisel kokku kolme eesmärki.

Tugevuskontrolli piirangu moodustavad seina paindekandevõime kontroll, maksimaalse siirde kontroll ja horisontaalkoormuse resultantjõu ja jäikustsentrite erinevus. Jäikusseinte paindetugevuse kontrolli kirjeldab valem 3.1. Olukorras kus seina normaaljõu tugevus on ületatud, on ka paindetugevus ületatud, mistõttu ei ole vaja täiendavat normaaljõu kontrolli teha. Valem arvestab kõikide seinte tugevuskontrolle kõigis neljas suunas ning valib neist maksimaalse. Kuna jäikusseintele jaotub koormus vastavalt nende inertsimomentidele, mis määrab ka paindetugevuse, siis ei tohiks jäikusseinte tugevuskontrollid liialt üksteisest erineda. Suured erinevused võivad tekkida erineva tuule suundade puhul. Näiteks võib jäikussüsteem olla ühes suunas väga tugev, kuid teises suunas nõrk. Maksimaalse tugevuskontrolli väärtuse võtmine takistab selliste

laotuste tekkimist, kuna määravaks saab alati kõige nõrgem olukord.

$$f_{tugevus} = max(\frac{M_{Ed,i}}{M_{Rd,i}}) \tag{3.1}$$

Maksimaalse siirde kontroll töötab analoogselt paindetugevuse kontrollile, erinevuseks on, et kõikide jäikusseinte siire on hoones võrdne, mistõttu erineb siire ainult tuule suuna tõttu. Siirde kontroll on toodud valemis 3.2.

$$f_{siire} = max(\frac{\delta_i}{\delta_{max}}) \tag{3.2}$$

Nii paindetugevuse kui ka siirde piirangu väärtused võivad väheneda kuni nende väärtus on null, kuigi kõik variandid, mille kontrollide väärtus on alla ühe, on juba sobivad. Selle eesmärgiks on soosida üksikute seinte grupeerumist, mille tulemusena saadakse oluliselt suurema inertsimomendiga sein ja tugevuskontrollid võivad väheneda oluliselt alla ühe. Sellisel piirangu defineerimisel on väiksema tugevuse ja siirde kontrolliga variandid alati paremad kui suuremaga. See kehtib ka juhul kui mõlemate kontrollide väärtused on alla ühe.

Täiendava piiranguna on lisatud veel koormus- ja jäikustsentrite erinevus, et suunata algoritmi looma väiksema väändemomendiga jäikusseinte laotusi. Väändemoment tekitab seintes täiendavaid sisejõude, mille tõttu peaks algoritm eelistama väiksema väändemomendiga lahendusi, kuid need ei ole piisavalt suured, et algoritmile olulist mõju avaldada. Ilma piiranguta on saadud lahendused tihti väga suure väändemomendiga. Piirangut kirjeldab valem 3.3. Juhul kui antud piirang on suurem maksimaalsest lubatud väärtusest, siis on funktsiooni väärtus võrdne kasutusprotsendiga. Vastasel juhul on funktsiooni väärtus null. Piirangu eesmärgiks on hoida jäikustsentri ja resultantjõu kaugust üksteisest teatud piiris, mistõttu on funktsioon sätestatud, et piirangu vähendamine alla maksimumi ei tooks enam täiendavat kasu.

$$f_{tsenter} = \begin{cases} \frac{l_t}{l_{t.max}} < 1;\ 0 \\ \frac{l_t}{l_{t.max}} \geq 1;\ \frac{l_t}{l_{t.max}} \end{cases} \tag{3.3}$$

kus
l_t - Jäikustsentri ja horisontaalse resultantjõu kauguste erinevus
$l_{t.max}$ - Maksimaalne lubatud kauguste erinevus

Piirangu funktsiooni väärtuseks võetakse selle üksikosade maksimaalne väärtus. Kuna piirangu funktsioon on esitatud eraldi eesmärgina, siis eelistab algorim alati lahendusi, kus tugevuskontroll on väiksem. Algoritm eristab, kas tugevuskontroll on ületatud 150% või 200% võrra ning eelistab vähem piirangut ületanud varianti. Piirangu funktsioon on esitatud valemis 3.4.

$$f_p = max(f_{tugevus}, f_{siire}, f_{tsenter}) \tag{3.4}$$

Viimaseks on iseseisva eesmärgina võetud jäikusseinte keerulisuse funktsioon, mis kirjeldab üksikute jäikusseinte ühinemisel tekkivat ristlõike kuju. Funktsioon on kirjeldatud valemiga 3.5. Keerulisuse astme määramine ja selle loomise põhjus on kirjeldatud alapunktis 3.7.

$$f_k = max(k_i) \tag{3.5}$$

kus
k_i - Seina keerulisuse aste

3.4 Optimeerimiseks kuluv aeg

Evolutsioonilised algoritmid vajavad lahenduste koondumiseks väga palju variante, mistõttu võib halvasti defineeritud ülesande korral kuluda lahenduse leidmiseks väga palju aega. Optimeerimiseks kuluv aeg sõltub peamiselt kahest tegurist: ühe variandi arvutusteks kuluv aeg ning vajalike variantide arv. Vähendades üheks variandiks kuluvat aega, saab sama ajaga proovida läbi rohkem variante. Antud töös kasutatakse arvutit, millel on nelja tuumaline protsessor Intel Core i5-10300H. Arvutuste teostamiseks kasutatakse ainult protsessorit, mitte videokaarti ning need optimeerimise algoritm kasutab kõiki tuumasid. Siiski kasutab Wallacei ligikaudu ainult 50% kogu võimsusest. Ühe variandi arvutuseks kuluv aeg sõltub jäikusseinte arvust, kuid see jääb üldiselt vahemikku üks kuni kaks sekundit.

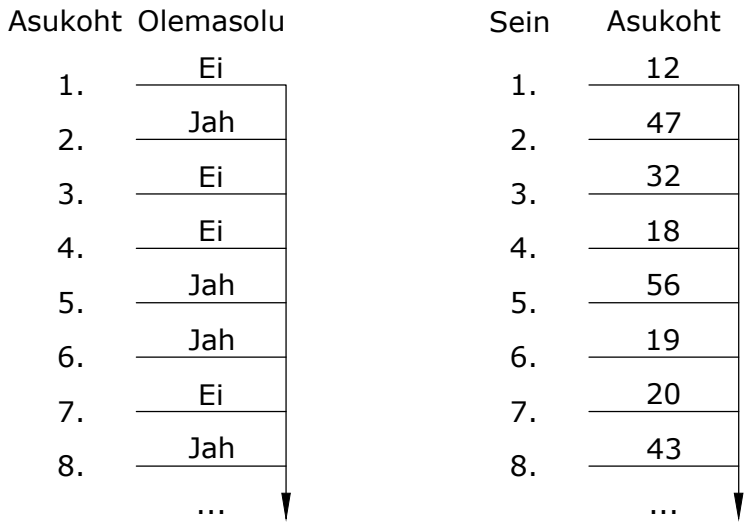
Antud töös kulub ühe variandi arvutamiseks üks kuni kaks sekundit olenevalt seinte arvust. Siiski ei ole võimalik ühe variandi arvutamiseks kuluvat aega lõputult vähendada, mistõttu on ka tähtis roll vajalike variantide vähendamisel.

Antud töös on jaotatud hoone ruudustikuks, mille iga serv võib olla jäikussein. Peatükis 3.2 kirjutati, et ruudustiku tihedusest sõltub jäikusseinte paiknemise võimalike kombinatsioonide arv ning hõredama ruudustiku kasutamisel on see väiksem. Siiski on isegi suure sammuga ruudustiku kasutamisel kombinatsioonide arv väga suur, mistõttu tuleks kombinatsioone veelgi piirata ning eemaldada variandid, mis kindlasti ei ole sobilikud. Üheks võimaluseks selle jaoks on seada piirangud jäikusseinte maksimaalsele arvule. Kuna optimaalses lahenduses on pigem vähem kui rohkem jäikusseinu, siis tasub piirata seinte maksimaalset arvu ning seega jätta välja variandid, kus seinu on liiga palju. Näiteks ei sobi kindlasti variant, kus enamik põranda pinnast on kaetud seintega. Kombinatsioonide arvu vähendamisega jõuab algoritm vähemate generatsioonidega sobivate lahendusteni, mistõttu väheneb ka nendeni jõudmiseks kuluv aeg.

3.5 Esmase populatsiooni loomine

Wallacei üheks puuduseks on, et sellel puudub võimalus täpsustada esimest populatsiooni ning see luuakse täiesti suvaliselt. Üldiselt määratakse evolutsiooniliste optimeerimise algoritmide puhul esimese populatsiooni variantideks juba kaudselt sobivad lahendused. Sellisel juhul ei pea algoritm alustama lahenduste otsimist täiesti suvaliselt, mis juhul võib esimestes generatsioonides esineda väga palju täiesti sobimatuid lahendusi. Kuna antud probleemi kombinatsioonide arv on äärmiselt suur, siis võib esmase populatsiooni kirjeldamata jätmine oluliselt pikendada optimeerimiseks kuluvat aega. Kuigi esmaseid lahendusi ei ole võimalik täpsustada, saab sellegipoolest Grasshopperis optimeerimise algoritmi poolt muudetavate parameetrite defineerimisega piirata algoritmi võimalusi suvalise populatsiooni loomisel.

Üheks peamiseks eesmärgiks esmase populatsiooni täpsustamisel on seinte arvu piiramine. Kuna hoone põhiplaan jagatakse ruudustikuks on seinte võimalike asukohtade arv suur, samas kui ainult üksikud nendest eksisteerivad optimaalsetes variantides. Suvaliselt variante genereerides on tõenäoliselt enamikes variantides liiga palju seinu, mistõttu võib kuluda mitmeid generatsioone enne kui üldse jõutakse sobivate laotuste piirkonda. Seinte maksimaalne arv määratakse Wallacei parameetritega, mis määravad seinte asukohad. Lihtsaim variant oleks iga seina asukoha jaoks luua parameeter, mille väärtus kirjeldab, kas seal eksisteerib sein või mitte. Sellisel juhul aga ei ole võimalik seada seinte maksimaalset arvu, mistõttu kasutatakse varianti, kus iga parameetri väärtus kirjeldab ühe seina asukohta hoones. Sellisel juhul on seinte maksimaalne arv võrdne parameetrite arvuga. Meetodi puuduseks on, et iga hoone jaoks peab parameetrite arvu käsitsi määrama. Parameetrite määramise variantide erinevust kirjeldab joonis 3.4



Joonis 3.4 Jäikusseinte muutujate määramine

Lisaks tuleks määrata ka seinte minimaalne arv. Kuna optimeerimise algoritm üritab leida kõiki Pareto kõveral olevaid lahendusi, siis mitmetes neist on väga väike seinte

arv väikese massiga, millega ei ole võimalik tugevuskontrolle täita. Algoritm võib kulutada liialt palju ressursse selliste lahenduste peale, mistõttu piiratakse seinte minimaalset arvu. Seinte minimaalse arvu määramiseks antakse osadele seina asukoha parameetritele täiendav väärtus, mille puhul seda seina ei eksisteeri. Lisa väärtus antakse nii mitmele parameetrile, et kui need on kõik kasutuses jääb hoones alles minimaalne seinte arv.

3.6 Seinte esmane dimensioneerimine

Minimaalne ja maksimaalne seinte arv määratakse vastavalt kaudsetele reeglitele. Saadud väärtused ei pea olema täpsed, vaid ligikaudselt määrama vahemikku, kuhu seinte arv peaks jääma. Kirjanduses on antud, et raudbetoonist jäikusseinte korral peaks seina kõrguse ning laiuse suhe jääma vahemikku 7...10 [26]. Samuti on Jinjie [11] väitnud, et seina laiuse ja paksuse suhe peaks olema vähemalt 8. Nende kahe reegli põhjal leitakse laotuste genereerimisel kasutatav seinte laius ja paksus. Näiteks 40 meetri kõrguse hoone puhul peaks seina laius olema 40/10=4 meetrit ning paksus maksimaalselt 4/8=0,5 meetrit. Saadud seinte laius võetakse seinte asukohti kirjeldava ruudustiku külje pikkuseks.

Teades seinte ligikaudseid dimensioone, tuleb määrata nende maksimaalne ja minimaalne arv. Maksimaalse seinte arvu saab leida kasutades hoone läbipainde piirangut. Kuna hoone töötab konsoolselt, siis on valemist kerge tuletada vajalik inertsimoment piirangu tagamiseks. Vajaliku inertsimomendi võib jagada eelnevalt leitud ühe seina inertsimomendiga, mille tulemusena saadakse vajalik seinte arv ühes suunas. Kuna optimaalsetes variantides on seintel keerulisemad kujud kui sirgjooned ning seega ka suuremad inertsimomendid, siis ei ole nii palju seinu vaja, mistõttu sobib see seinte maksimaalseks arvuks. Seinte arvu määramisel tuleb arvestada, et läbipaine peab olema tagatud mõlemas suunas, mistõttu võib ühes suunas vajalike seinte arv olla suurem kui teises suunas. Maksimaalne seinte arv saadakse summeerides vajalike seinte arv kahes ristuvas suunas. Seinte minimaalse arvu leidmiseks puuduvad täpsed meetodid ning see võetakse kogemuse järgi. Antud töös on minimaalseks seinte arvuks võetud ligikaudu 50% maksimaalsest arvust.

3.7 Suurte ristlõigete piiramine

Jäikusseinte laotuste optimeerimisel tekivad tihti väga keerulise kujuga jäikusseinte grupid, mille kasutamine reaalselt võib osutuda ebapraktiliseks. Kuna suurematel ristlõigetel on oluliselt suurem inertsimoment on ka nende tugevuskontrollide tulemused paremad, mistõttu eelistab algoritm lahendusi, kus hoones on üks suur jäikussein. Kuigi hoone keskel asuv jäikussüdamik on väga levinud lahendus ei ole algoritmi poolt loodud variantide kujud üldiselt korrapärased nagu I-, U- või ristkülikukujulised. Seetõttu oleks vaja piirata kujusid, mida algoritm võib jäikusseintest luua.

Sarnase probleemiga tegeles ka Philps [1], kelle algoritm kaldus looma ühte väga keerulise kujuga jäikusseinte massi hoone keskele. Selle vältimiseks loodi piirang seinte gruppide arvule, mis soodustas algoritmi kasutama rohkem kui ühte gruppi. Siiski ei

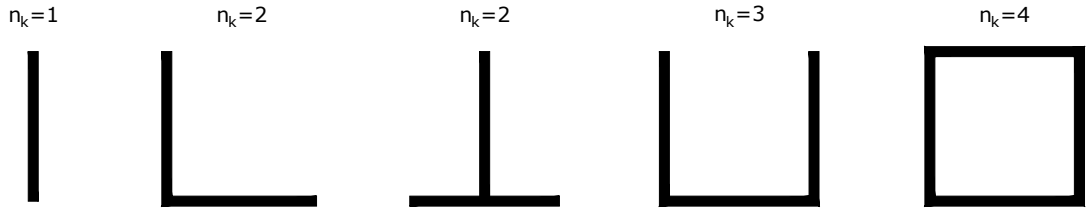
mõjuta see otseselt gruppide enda kuju ning suure seinte arvu puhul osutuvad grupid sellegipoolest väga keerulisteks.

Autor pakub välja lahenduse mõõta jäikusseinte keerulisust ning luua piirang, mis soodustaks kasutama vähem keerulisi jäikusseinu. Selle jaoks leitakse iga seina keerulisuse aste n_k, mis kirjeldab mitmest sirgjoonelisest lõigust jäikussein koosneb. Näiteks kahest üksikust seinast koosneval sirgel seinal on keerulisuse aste 1, samas kui L-kujulisel seinal oleks astme väärtus 2, U-kujulisel 3 ning ristkülikukujulisel 4. Keerulisuse astmed on illustreeritud joonisel 3.5. Kuna üldiselt ristkülikukujulisest keerulisemaid jäikusseinu ei kasutata, siis määratakse maksimaalseks keerulisuse astmeks 4. Kui laotusel on vähemalt üks kõrgema astmega südamik, siis piirangu funktsiooni väärtus kasvab. Funktsioon on antud valemis 3.6.

$$f_k = max(n_k) - n_{max} \tag{3.6}$$

kus
n_k - Seina grupi keerulisuse aste
n_{max} - Maksimaalne lubatud seina grupi keerulisuse aste

Meetodi eelis on, et see võimaldab jäikusseintel koosneda mitmest üksikust seinast, kuid siiski säilitada madal astme arv lihtsa kuju korral. Näiteks võib L-kujulise grupi üks külg koosneda kolmest seinast, kuid sellel on siiski sama keerulisuse aste kui külg koosneks ainult ühest seinast. Seetõttu ei olene südamike keerulisus selle suurusest, vaid ainult kujust.



Joonis 3.5 Keerulisuse aste

3.8 Optimeerimise algoritmi sätted

Evolutsioonilistes optimeerimise algoritmides on sätted, mille väärtused tuleb määrata enne töö algust. Nendest kaks kõige tähtsamat on populatsiooni suurus ja generatsioonide arv. Suurem populatsioon võimaldab katta suuremat osa võimalikest variantidest, mistõttu on üksteisest erinevamate ja paremate variantide leidmise tõenäosus suurem. Suurem generatsioonide arv võimaldab algoritmil rohkem koonduda ning jõuda lähemale olukorrale, kus algoritm ei oska enam paremaid variante luua. Kuna optimeerimiseks vajalik variantide arv on võrdne generatsioonide arvu ja populatsiooni suuruse korrutisega, siis mõlema suure väärtuse puhul muutub arvutatavate variantide

arv äärmiselt suureks, mistõttu kulub arvutustele rohkem aega. Seega tuleb leida tasakaal nende kahe parameetri vahel. Ülejäänud parameetrite väärtusteks võetakse Wallaceis vaikimisi antud väärtused, mis jäävad kirjanduses soovitatud vahemikku [12]. Antud parameetrid ja nende väärtused on järgnevad:

- *Crossover probability* = 0,9 - Lahenduste hulk, mis moodustavad järgmise generatsiooni.
- *Mutation probability* = 1/n - Muteeruvate isendite hulk generatsioonis, kus n on muudetavate parameetrite arv.
- *Crossover and Mutation Distribution Index* = 20 - Suurem väärtus tõstab tõenäosust, et uued isendid on sarnased vanematele, väike väärtus lubab neil olla erinevam.

3.9 Kokkuvõte

Antud peatükis loodi võimalus kasutada varasemalt loodud jäikusseinte arvutamise programmi optimeerimise algoritmil, et automaatselt luua jäikusseinte laotusi. Selle jaoks jaotati hoone põhiplaan ruudustikuks, mille iga serv on potentsiaalne jäikussein. Optimeerimise algoritmile defineeriti funktsioonid, millega hinnata jäikusseinte vastavust soovitud tulemustele ning lisaks tugevuskontrollidele lisati ka täiendavad piirangud, et saadud jäikusseinte laotused oleks praktilisemad.

4. JUHTUMIUURINGUD

Käesolevas peatükis kasutatakse evolutsioonilist optimeerimise algoritmi, et luua jäikusseinte laotusi erineva kujuga hoonetele. Katsete eesmärgis on leida kui efektiivsed on algoritmi loodud laotused ning millistes olukordades töötab programm hästi või halvasti.

4.1 Üldised katsete andmed

Järgnevalt on toodud andmed, mis on kõikides katsetes ühised. Kasutatava raudbetooni andmed on järgnevad:

- Betooni normatiivne survetugevus - 40 MPa
- Betooni varutegur - 1,5
- Betooni elastsusmoodul - 20 000 MPa
- Armatuuri normatiivne voolepiir - 500 MPa
- Armatuuri varutegur - 1,15
- Armatuuri elastsusmoodul - 200 000 MPa
- Armeerimistegur - 0,005

Hoone kujuga seotud andmed on järgmised:

- Hoone kõrgus - 49,5 meetrit
- Korruste arv - 15 tk
- Korruse kõrgus - 3,3 meetrit

Koormustega seotud andmed on järgmised:

- Tuulekoormus - 1,2 kN/m^2
- Vahelaekoormus - 6 kN/m^2
- Vahelae koormuse ala pikkus - 5 meetrit

Optimeerimise algoritmiga seotud andmed on järgmised:

- Horsiontaalkoormuse resultantjõu ja jäikustsentri lubatud erinevus - 1,5 meetrit
- Jäikusseina lubatud keerulisuse aste - 4

Katsed teostatakse andmetega, mis peaksid sarnanema päris elus esinevatele olukordadele. Koormuskombinatsioonina vaadeldakse olukorda, kus tuul on domineeriv muutuvkoormus ning vahelae koormus mõjub soodsalt. Üldiselt ei ole vahelagedelt mõjuv vertikaalne koormus piisavalt suur, et betoonist ristlõikes toimuks survepurunemine, mistõttu mõjub normaaljõud soodsalt ristlõike kandevõimele. Kasutatav koormuskombinatsioon on järgnev:

$$1,0 \cdot G + 1,5 \cdot W$$

kus G - alaliskoormus W - tuulekoormus

Vahelae koormuseks on arvestatud 6 kN/m2, mis sisaldab endas 265 mm kõrgust õõnespaneeli koormusega 3,8 kN/m2, 80 mm paksust betoonist pealevalu koormusega 2,0 kN/m2 ja 30 mm paksust sammumüra villa koormusega 0,2 kN/m2. Tuulekoormusena on arvestatud suruvast ja imevast jõust tekkivat resultantjõudu ning on juba läbi korrutatud varuteguriga.

4.2 Populatsiooni suuruse määramine

Töö eesmärk ei ole leida ainult üks hea jäikusseinte laotus vaid mitu, sest paljud saadud lahendustest võivad osutuda ebasobivateks muudel põhjustel, millega programm ei ole arvestanud. Seetõttu on tähtis seadistada optimeerimise algoritm nii, et tulemuste koondudes oleks mitmeid üksteisest oluliselt erinevaid lahendusi. Lahenduste arvu määrab suures osas populatsiooni suurus, kuna sellest sõltub kui palju erinevaid variante algoritm uurib. Varasemates loetud töödes, mis kasutasid evolutsioonilisi optimeerimise algoritme, varieerus populatsiooni suurus äärmiselt palju (16...2000) ning enamikus töödes esitati algoritmi tulemusena ainult üks lahendus. Kuna populatsiooni suuruse mõju lahenduste mitmekesisusele on teadmata, viidi läbi katsed erinevate populatsiooni suurustega, et leida antud töö jaoks sobiv väärtus. Ühelt poolt peab populatsiooni suurus olema piisavalt suur, et tulemustes oleks mitu üksteisest erinevat lahendust. Samas ei tohi populatsioon olla liiga suur, et algoritmi tööaeg muutuks ebapraktiliselt pikaks. Antud töös arvestatakse, et sobiv ajakulu algoritmil töötamiseks on kuni 15 tundi, sest sellisel juhul on võimalik programm panna tööle tööpäeva lõpus ning hommikul saada tulemused kätte.

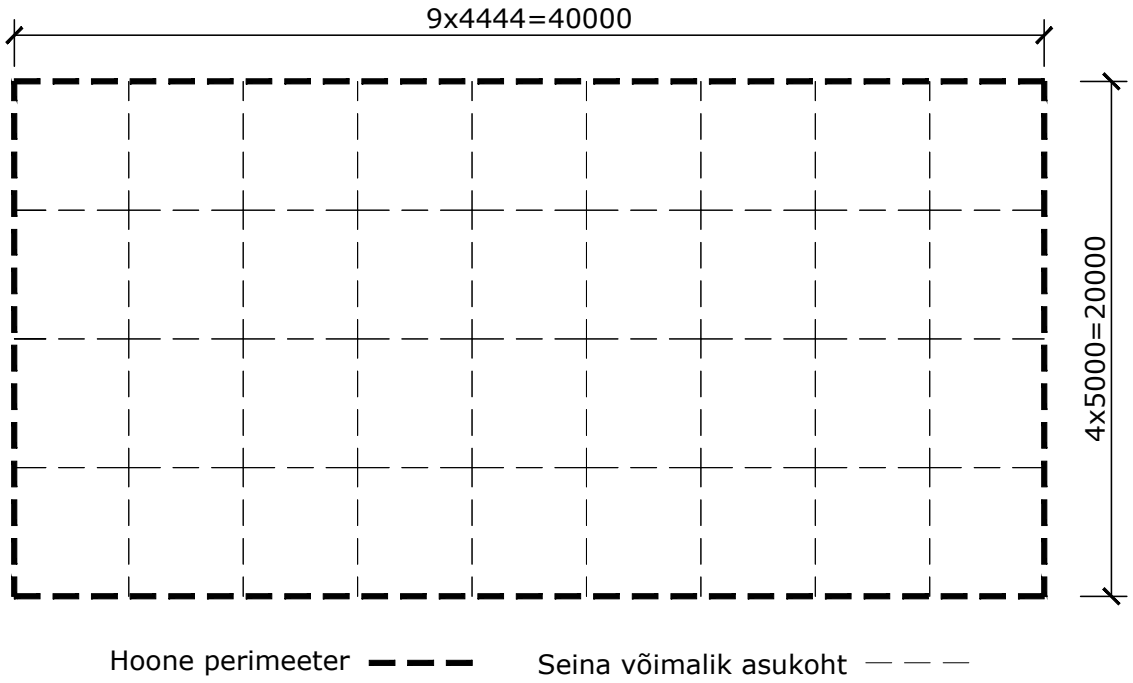
Katsete käigus leiti mitu sarnaste lahenduste gruppi on generatsioonis, kus jäikusseinte pindala eesmärgi funktsioon lõpetab kahanemise. Selles generatsioonis peetakse algoritmi tulemust koondunuks. Lahendused on määratletud sarnasteks kui need ei erine üksteisest rohkem kui 51%. Ehk kui kahel variandil on rohkem kui 51% ühiseid üksikuid seinu, siis need loetakse sarnasteks lahenduseks. Väärtus 51% on võetud olukorraks kui variantidel on paarisarv seinu, sest siis peavad rohkem kui pooled üksikud seinad olema samades asukohtades, et tulemused oleksid sarnased. Näiteks kaheksa üksiku seina puhul peab olema viis ühesuguse asukohaga seina. Lisaks loetakse lahendused sarnasteks kui need ei ole omavahel sarnased, kuid neil on ühine sarnane lahendus. Näitena tuuakse kolm kolmest numbrist koosnevat rida.

1. 1-2-7
2. 2-5-7
3. 3-5-7

Esimene ja teine numbrite rida on sarnased, kuna nendes mõlemas on numbrid 2 ja 7. Samuti on teine ja kolmas sarnased, sest neil on ühised numbrid 5 ja 7. Esimene ja

kolmas rida omavahel ei ole sarnased, kuna neil on ainult üks ühine number 7. Siiski, kuna kõik arvu read on omavahel sarnaste ridadega seotud, loetakse kõik arvu read omavahel sarnasteks.

Katse viidi läbi ristkülikulise hoone puhul, mis on toodud joonisel 4.1.



Joonis 4.1 Populatsiooni suuruse määramiseks kasutatud hoone põhiplaan

Katse tulemustest filtreeritakse välja kõik ebasobivad lahendused, ehk mille tugevuskontrolli piirang on suurem kui üks või mille seinte keerulisuse aste on suurem kui 4. Katsed viidi läbi erinevate populatsiooni suuruste korral kuni algoritm koondus. Saadud tulemustest määratakse sarnaste variantide arv. Katsed tehti populatsiooni suurustele 50, 100, 250, 500 ja 1000. Katse tulemused on antud tabelis 4.2.

Katse	**n_{pop}**	**Koondumise generatsioon**	**Variantide arv**	**Ajakulu [h]**	**Sobivate lahenduste arv**	
					Kokku	**Sarnased grupid**
1	50	30	1500	0,5	8	2
2	100	30	3000	1,0	16	2
3	250	25	6250	2,1	46	8
4	500	25	12500	4,2	100	7
5	1000	21	21000	7,0	213	11

Joonis 4.2 Populatsiooni suuruse määramise katsete tulemused

Katsete tulemused kinnitavad väidet, et mida suurem on populatsioon, seda suurem on ka üksteisest erinevate lahenduste arv. Suurema populatsiooni puhul on ka sobivate lahenduste arv suurem, mistõttu esineb grupi siseselt rohkem varieeruvust. Suurema

populatsiooni arvu puhul kulub vähem generatsioone tulemuse koondumiseni, kuid see ei ole siiski piisavalt suur, et hoida arvutusteks vajalike variantide arv konstantsena. Seega kaasneb suurema populatsiooni kasutamisega siiski oluliselt suurem ajakulu. Antud katses kulus ühe variandi arvutamiseks ligikaudu 1,2 sekundit, mistõttu on näiteks populatsiooni suuruse 1000 juures optimeerimiseks kuluv aeg 7 tundi. Antud töös kasutatakse järgnevate juhtumiuuringute puhul populatsiooni suurust 1000, sest see võimaldab saada kõige rohkem üksteisest erinevaid tulemusi, samas kui ajakulu jääb siiski sobivatesse piiridesse. Generatsioonide arvuks võetakse 30, et tagada tulemuste koondumine, mis võib erinevate hoonete puhul rohkem aega võtta. Tulemused võetakse siiski kõige esimesest generatsioonist, mida võib pidada koondunuks.

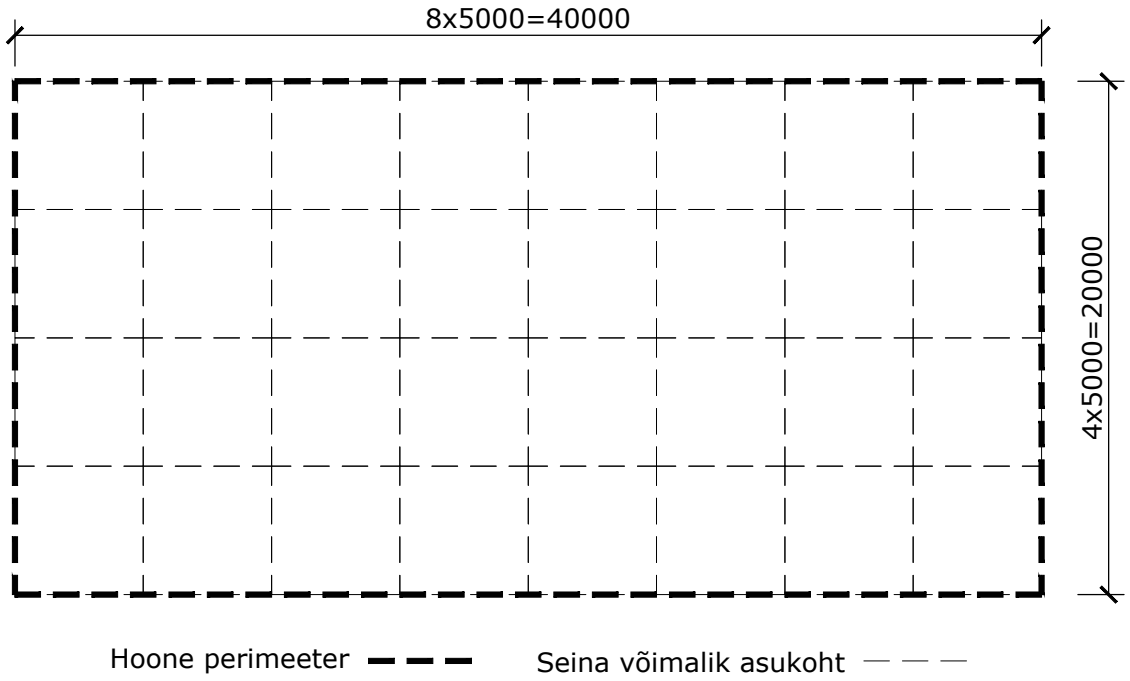
4.3 Jäikusseinte laotuste genereerimine

Välja töötatud jäikusseinte automaatset laotuste genereerimise programmi rakendati mitme erineva põhiplaaniga hoone puhul, et uurida selle efektiivsust erinevates olukordades. Esimene katse käsitleb ristkülikukujulist hoonet, mille üks külg on oluliselt pikem kui teine, mistõttu on vajalik jäikus ühes suunas suurem kui teises. Teine katse käsitleb L-kujulist hoonet, mille horisontaalkoormuse resultantjõud jääb hoonest väljaspoole, mistõttu ei ole võimalik kasutada ühte keskset jäikussüdamiku ilma suurte väändemomentide tekkimiseta. Viimasena vaadeldakse olukorda olemasoleva arhitektuurse plaaniga, kus jäikusseinte võimalikud asukohad kattuvad arhitektuurse plaani seintega.

4.3.1 Ristkülikukujuline hoone

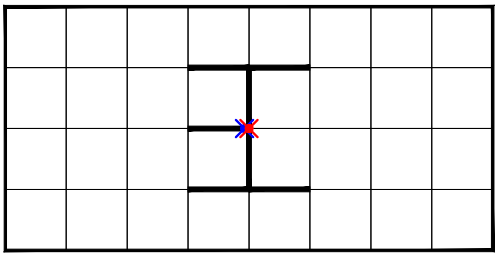
Esimeses katses käsitletakse ristkülikukujulist hoonet. Üks hoone külgedest on pikem kui teine, mistõttu on ka vajalik jäikus ühes suunas suurem kui teises. Seetõttu peaks see kajastuma ka algoritmi loodud laotustes. Kuna hoone on kahes suunas sümmeetriline, siis on sobiv lahendus kasutada jäikusseinu hoone keskel, mis tagab horisontaalkoormuse resultantjõu ja jäikustsentri kattuvuse.

Katses kasutatud hoone külje pikkused on 40 ja 20 meetrit ning põhiplaan on jagatud ruudustikuks külje pikkusega viis meetrit. Iga külg on võimalik jäikusseina asukoht välja arvatud hoone perimeeter. Katses kasutatud hoone põhiplaan on toodud joonisel 4.3. Kuna hoone keskele on võimalik paigutada jäikusseinad, siis oleks lihtsaim lahendus kasutada I- või U-kujulist jäikussüdamikku hoone keskel. Juhul kui hoone keskel asuks ruut, oleks sobilik lahendus ristkülikukujuline jäikussein. Vastavalt hoone lubatud siirde tagamiseks võeti maksimaalseks seinte arvuks kaksteist ja minimaalseks kuus.

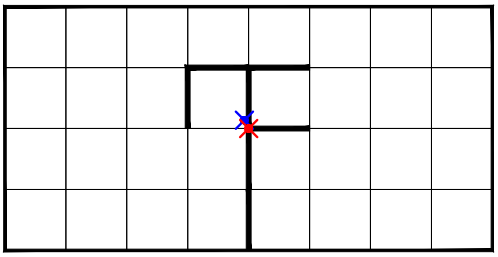


Joonis 4.3 Ristkülikuline hoone

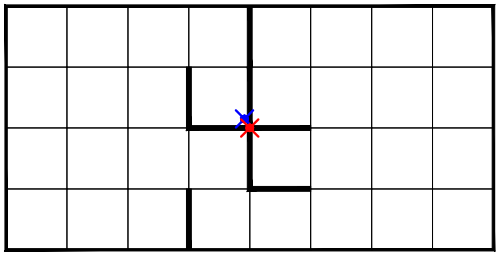
Programmi loodud variantidest filtreeriti välja kõik, mille tugevuse piirangu funktsioon oli suurem kui üks ning mille keerulisuse aste oli suurem kui neli. Antud variantidest valiti välja kolm iseloomulikku lahendust. Variandid on esitatud joonisel 4.4.



Jäikusseinte arv - 7
Tsentrite erinevus - 362 mm



Jäikusseinte arv - 7
Tsentrite erinevus - 766 mm



Jäikusseinte arv - 8
Tsentrite erinevus - 862 mm

Joonis 4.4 Ristkülikulisele hoonele loodud jäikusseinte laotused:
punane punkt - horisontaalkoormuse resultantjõu asukoht, sinine punkt - jäikustsentri asukoht

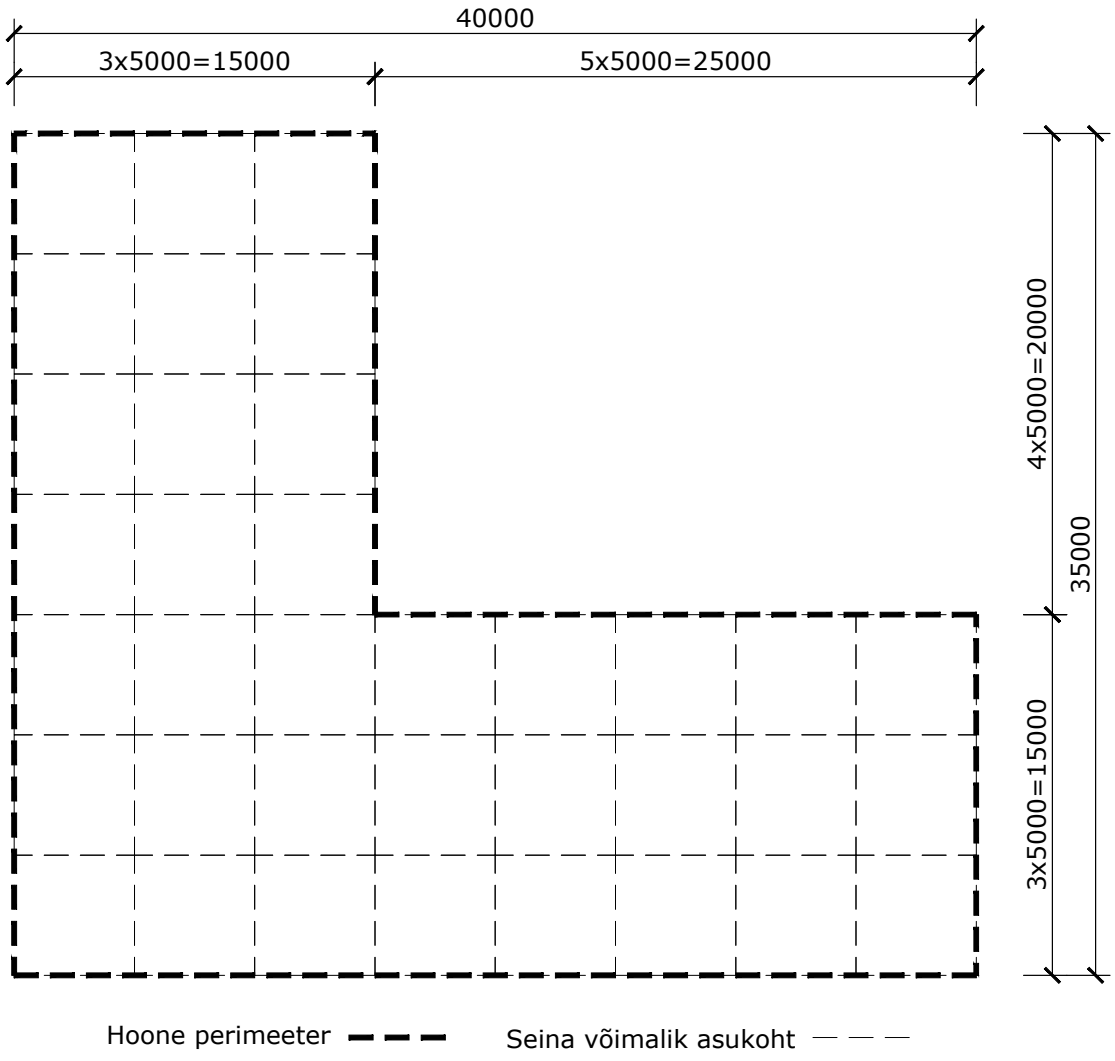
Välja toodud lahendustest on näha, et algoritm kaldub looma jäikusseinu hoone keskele. Selle põhjuseks on jäikustsentri osa tugevuskontrolli piirangu funktsioonis, mis tagab horisontaalkoormuse resultantjõu ja jäikustsentri väikese erinevuse. Samasuguse lahenduse saaks saavutada ka kahe jäikusseinaga, mis asuvad üksteisest eraldi kummalgi pool hoonet, kuid ühtegi sellist lahendust algoritm ei leidnud.

Algoritmi loodud jäikusseinte ristlõiked ei ole enamasti konstruktsioonilises mõttes efektiivsed. Nagu mainitud alapeatükis 2.3.1 on sümmeetrilised ristlõiked kõige efektiivsemad, kuna nende tugevused on võrdsed, olenemata tuule suunast. Lisaks peaksid ristlõiked koosnema tuule suunaga paralleelsetest seintest ristlõike keskel ja nendega ristuvatest seintest ristlõike servades, et saada võimalikult suur inertsimoment. Kõige ebapraktilisemad on seinad ristlõike keskel, kuna need töötavad peamiselt ainult ühe tuule suuna puhul. Efektiivse ristlõike kasutust on näha ainult esimeses variandis, kus hoone jäikusseinad moodustavad peaaegu I-kuju, kuid samas esineb seal ristlõike keskel üleliigne sein. Ilma nimetatud seinata oleks samuti jäikusseinte tugevuskontroll tagatud, kuid algoritm seda lahendust ei leidnud. Ülejäänud kahes variandis on algoritm loonud seintest suure ristlõike, et saavutada suurem inertsimoment, kuid need on pigem T- kui I-kujulised, mistõttu on need pigem ebaefektiivsed.

Lihtsa kujuga hoone puhul nagu antud ristkülikukujuline hoone ei ole automaatselt loodud jäikusseinte laotused paremad ega varieeruvamad kui insener suudaks ise vähese vaevaga luua. Kuigi algoritmi loodud esimest varianti võib pidada sobivaks, siis ei ole saadud tulemus väärt aega, mis kulus selle genereerimiseks. Seega ei ole antud katse põhjal lihtsate hoonete puhul jäikusseinte laotuste loomine automaatselt praktiline.

4.3.2 L-kujuline hoone

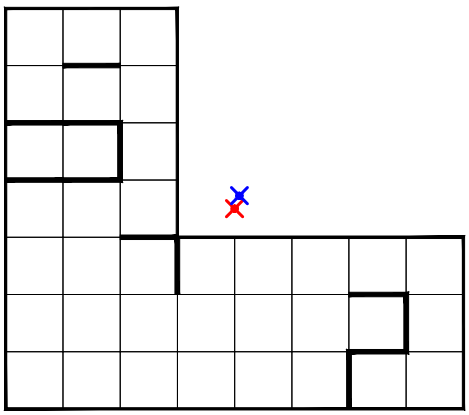
L-kujulise hoone puhul uuritakse programmi efektiivust keerulise kujuga põhiplaanide puhul, mis erinevad oluliselt ristkülikulisest kujust. Antud hoone puhul jääb horsiontaalse koormuse resultantjõu asukoht väljaspoole hoonet, mistõttu ei ole võimalik efektiivselt kasutada ühte jäikusseinte gruppi. Et hoida resultantjõu ja jäikustsentrite erinevus väiksena, peab lahendus koosnema mitmest jäikusseinte grupist. Katses kasutatakse hoonet, mille põhiplaan koosneb kahest ristkülikust, üks suurusega 40x15 meetrit ja teine 15x20 meetrit. Hoone ruudustiku sammuks on seatud viis meetrit ning illustreeriv skeem on toodud joonisel 4.5. Hoone maksimaalseks seinte arvuks võeti 16 ja minimaalseks 11.



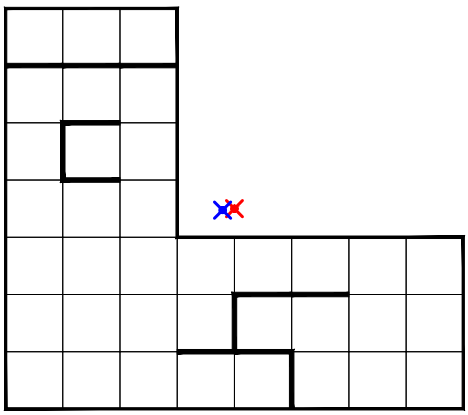
Joonis 4.5 L-kujuline hoone

Algoritmi poolt loodud variantidest on esitatud kolm tükki joonisel 4.6. Sarnaselt ristkülikukujulisele hoonele on ka siin horisontaalkoormuse resultantjõu ja jäikustsentrite vahe väike, kuid selle jaoks kasutas algoritm L-kujulise hoone puhul kahte suuremat jäikusseinte gruppi, mis asuvad hoone otste suunas. Antud lahendus on autori arvates sobilik.

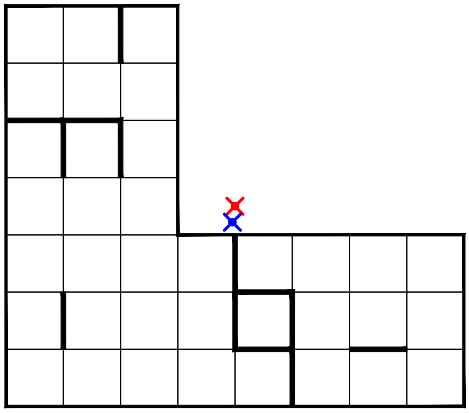
Programm on kasutanud jäikusseintena rohkem kompaktsemaid gruppe nagu U- ja ristkülikukujulised. Selle põhjuseks on ruudustiku paiknemine, kus jäikusseinte asukohad ei paikne hoone kesktelgedel. Enamik loodud jäikusseinte gruppidest ei ole efektiivsed, kuna U-kujulised jäikusseinad ei ole sama tugevad mõlema tuule suuna puhul nagu kirjeldatud joonisel 2.4.



Jäikusseinte arv - 12
Tsentrite erinevus - 1204 mm



Jäikusseinte arv - 12
Tsentrite erinevus - 1044 mm



Jäikusseinte arv - 13
Tsentrite erinevus - 1425 mm

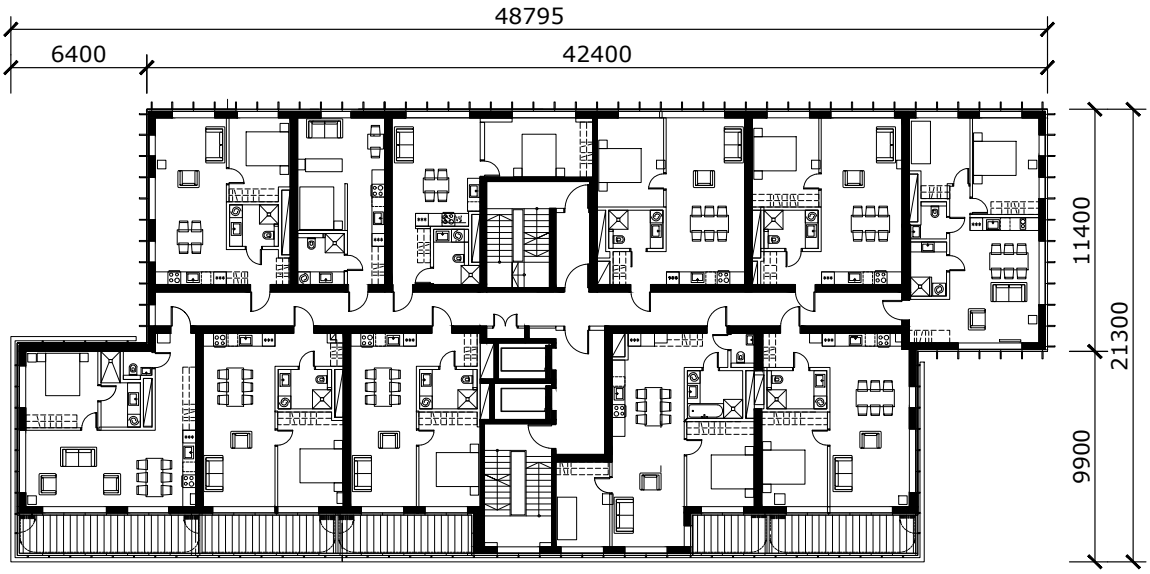
Joonis 4.6 L-kujulise hoone jäikusseinte laotused:
punane punkt - horisontaalkoormuse resultantjõu asukoht, sinine punkt - jäikustsentri asukoht

L-kujulise hoone puhul on programmi poolt pakutud lahendused sobivas piirkonnas ning see suudab paigutada seinad nii et ei teki suurt väändemomenti. Samas on loodud lahendused siiski kaootilised ning sisaldavad endas mitmeid üksikuid jäikusseinu ja ebaefektiivse kujuga jäikusseinte gruppe. Seega suudab programm leida ligikaudselt häid lahendusi, kuidas jäikusseinad peaksid paiknema, kuid need vajavad palju täiendavat muutmist programmi kasutaja poolt, enne kui neid võib nimetada praktilisteks. Keeruliste hoonete puhul, kus insener ei oska kohe head jäikusseinte laotust välja pakkuda, võib automaatsete laotuse loomisest abi olla, et anda insenerile ideid, milliseid variante täpsemalt läbi proovida.

4.3.3 Arhitektuurse põhiplaaniga hoone

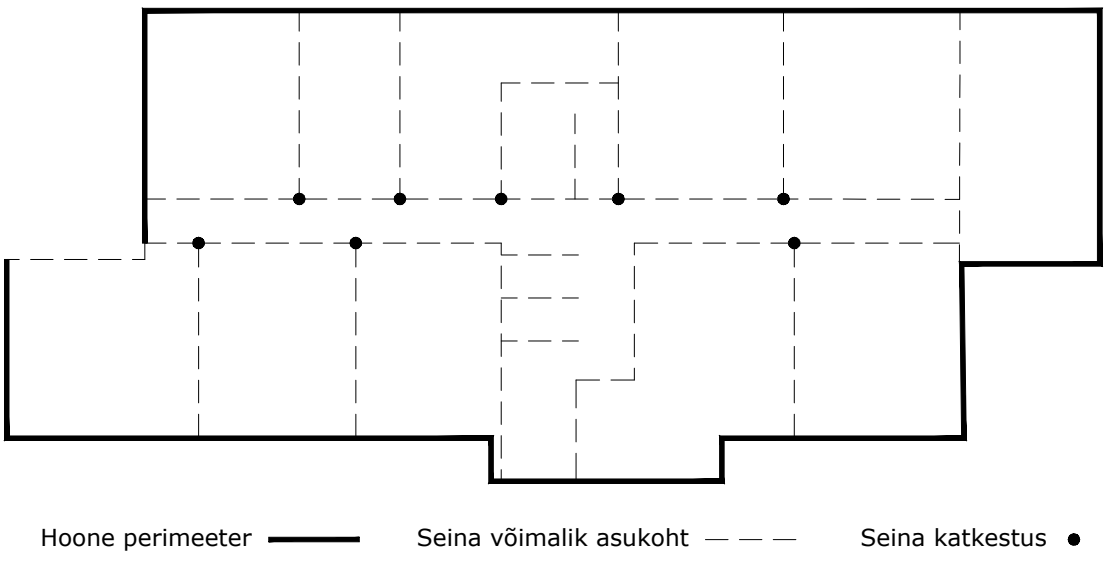
Antud katses proovitakse jäikusseinte laotuste genereerimise programmi sobivust olukorras, kus hoonel on juba arhitektuurne põhiplaan olemas. Valitud hoone on Tallinna kesklinnas valmiva Arteri büroo- ja elumajakompleksi üks tornidest, mille põhiplaan

on toodud joonisel 4.7. Hoone põhiplaani keskel asuvad kaks trepikoda ja liftišahtid, mis tuleohutusnõuete pärast rajatakse üldiselt betoonist. Kuigi programm ei arvesta arhitektuursete nõuetega, siis eeldatakse et ruumide keskse asetuse tõttu paigutab programm sellegipoolest sinna jäikusseinad.



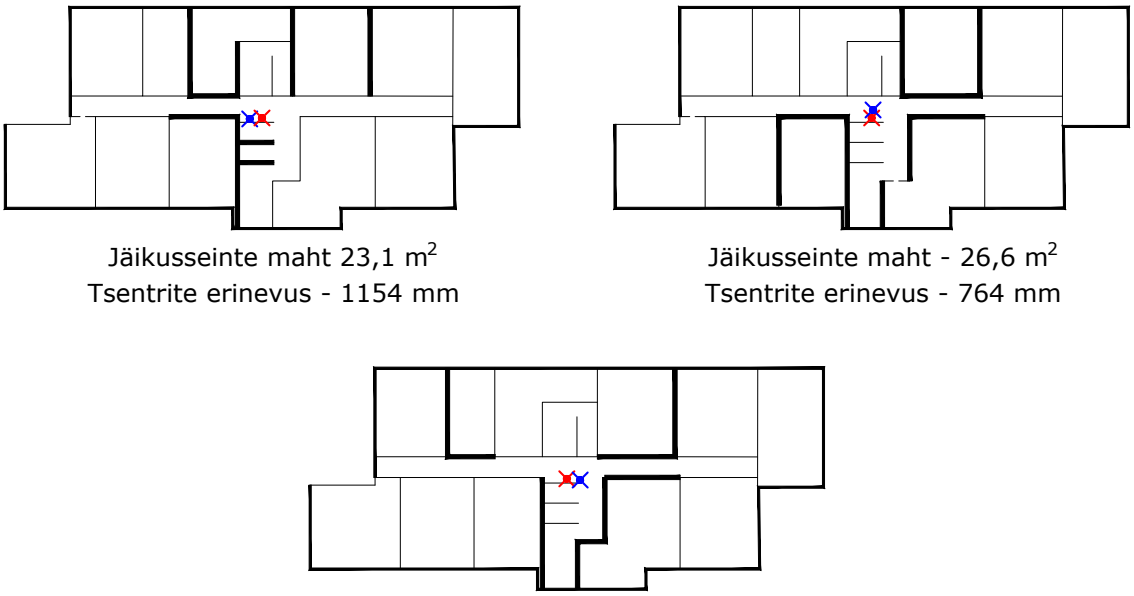
Joonis 4.7 Hoone põhiplaan

Hoone võimalike jäikusseinte asukohad on toodud joonisel 4.8. Arhitektuurse põhiplaani seinad jagati üksikuteks lõikudeks, kus võivad asuda jäikusseinad. Pikad sirged seinad on jaotatud lõikudeks kohtades, kus teised seinad ristuvad nendega. Antud kohad on tähistatud joonisel mustade täppidega. Maksimaalseks seinte arvuks võeti kümme ning minimaalseks viis. Kuna jäikusseinte pikkused on hoones erinevad, siis on tulemustena kuvatud jäikusseinte kogust nende pindala järgi.



Joonis 4.8 Hoone jäikusseinte skeem

Algoritmi poolt loodud jäikusseinte laotused on toodud joonisel 4.9. Jäikusseinad esinevad osaliselt trepikodade ja liftišahtide seintena, kuid on tihtipeale nende läheduses olevad teised seinad. Selle põhjuseks võib olla, et trepikoja ja liftišahti ümbruses olevad seinad on palju pikemad, mistõttu on nende kasutamisel lihtsam saada vajalik jäikus kui mitmete lühemate seinte kasutamisel. Programmi loodud jäikusseinte laotustel on jäikusseinad asetatud pigem kaootiliselt ning kuigi nende puhul on tagatud tugevuskontrollid ja väike väändemoment, ei ole need kõige praktilisemad ehitamiseks. Arhitektuurse põhiplaaniga olukorras oleks seetõttu mõistlik esmalt juba paigutada jäikusseinad loogilistesse kohtadesse nagu liftišahtid ja trepikojad. Juhul kui nendest jääb puudu, võib kasutada algoritmi, et leida sobivad kohad täiendavatele jäikusseintele.



Joonis 4.9 Arhitektuurse põhiplaaniga hoone jäikusseinte laotused :
punane punkt - horisontaalkoormuse resultantjõu asukoht, sinine punkt - jäikustsentri asukoht

4.4 Kokkuvõte

Antud peatükis uuriti automaatset jäikusseinte laotuste loomist, kasutades selleks evolutsioonilist optimeerimise algoritmi. Loodud jäikusseinte laotused olid sarnased inseneri poolt kasutatavatele laotustele, kuid nendes sisalduvad jäikusseinad olid ebaefektiivsete ristlõigetega. Jäikusseinte laotused arhitektuurse alusplaaniga hoones olid kõige ebasobivamad ega asetsenud üldiselt arhitektuurselt sobivate ruumide ümber.

5. TÖÖ ANALÜÜS

Antud töös loodi programm jäikusseinte laotuste analüüsimiseks ja dimensioneerimiseks ning lahenduste loomiseks automaatselt. Järgnevalt on toodud loodud programmi analüüs, millised on selle võimalikud kasutuskohad ning võimalused edasiarenduseks.

5.1 Loodud programm võrreldes teiste arvutustarkvaradega

Loodud programmi suurimaks eeliseks on selle võimekus kiirelt läbi proovida erinevaid jäikusseinte laotusi. Selle eelised võib kokku võtta kolme omadusega, milleks on kiirus, kasutajasõbralikkus ja universaalsus. Järgnevalt võrreldakse seda kahe projekteerimise valdkonnas levinud arvutustarkvaraga, millega oleks võimalik sarnaseid arvutusi teostada. Nendeks on Microsofti tabelarvutus programm Excel [27] ja Autodeski lõplike elementide meetodi programm Robot Structural Analysis Professional (ARSA) [28].

Töös kasutati jäikusseinte sisejõudude leidmiseks ja tugevusarvutuste teostamiseks analüütilisi meetodeid, mistõttu kulus ühe variandi arvutuste teostamiseks üks kuni kaks sekundit. Sarnase arvutuse teostamine Excelis, mis kasutaks samuti analüütilisi meetodeid, oleks peaaegu silmapilkne. ARSA puhul, mis kasutab lõplike elementide meetodit oleks ajakulu märgatavalt suurem. Väga väike arvutustele kuluv aeg võimaldab kasutajal liigutada hoone plaani jäikusseina asukohta ning kohe näha sellega kaasnevat mõju. Suure ajakulu puhul ei oleks autori arvates sellise programmiga enam mugav tööd teha, kuna pidevad pausid segavad sujuvat töövoogu ning kokkuvõttes võib ajakulu muutuda väga suureks.

Jäikusseinte laotuste genereerimisel evolutsiooniliste algoritmidega oleks lõplike elementide meetodiga arvutuste teostamine ebapraktiline, sest algoritm vajab lahenduste leidmiseks palju iteratsioone, mis lõplike elementide meetodite puhul vajaks nädalaid. Kui aja kokkuhoiu eesmärgil vähendada populatsiooni suurust, uuritakse palju väiksemat võimalike lahenduste hulka, mistõttu algoritm koondub tõenäoliselt esimesse leitud maksimumi. Seetõttu võiks sellistes olukordades kasutada teisi optimeerimise algoritme, mis sobivad paremini ühe lokaalse lahenduseni jõudmiseks ja suudaks teha seda kiiremini.

Lisaks kiirusele on sujuva töövoo jaoks vajalik ka mugav kasutajaliides, et vältida täiendavat ajakulu ja ebamugavust lihtsate käskude täitmisel. Võrreldes Exceli ja ARSA'ga on Rhinol ja Grasshopperil suur eelis, kuna tegemist on joonestamise programmiga, mille jaoks on käskude kiire sisestamine esmatähtis. Excelis muudetakse arvutuste sisendandmeid nagu seinte arvu, kuju ja paiknemist üldiselt numbriliste väärtuste muutmisega, mis ei ole intuitiivne. Lisaks sellele on ilma visuaalse väljundita suurem vigade tekkimise oht. Robotis ja Rhinos teostatakse muudatused graafiliselt, kuid Robotis on nende teostamine äärmiselt tülikas, samas kui Rhinos suudab kogenud kasutaja teha seda paari nupuvajutusega.

Viimasena vaadeldakse universaalsust, ehk kui paljude erinevate keeruliste olukordadega suudab programm toime tulla. Selles osas on eelis ARSA'l, kuna lõplike elementide meetod ei sõltu seinte geomeetriast ning suudab peaaegu igasuguste olukordadega toime tulla. Excel on selles see-eest äärmiselt nõrk, sest arvutused teostatakse valemitega, mis üldiselt kehtivad ainult kindla geomeetria puhul. Keerulisemate olukordade, näiteks T-kujuliste seinte puhul, tuleb kirjutada nendega arvestavad täiendavad valemid. Niisugune meetod võib muutuda kiiresti liiga mahukaks kui igale erinevale olukorrale on vaja luua eraldi lahendus. Rhino koos Grasshopperiga jääb kahe eelmise programmi vahele. Arvutused teostatakse samuti valemitega, kuid kuna Grasshopper on loodud geomeetria manipuleerimisele, siis on ühe üldise valemiga võimalik käsitleda väga erinevaid geomeetriad. Näiteks antud töös kasutati surutud ristlõike kontrolli valemis survetsooni pindala, mille Grasshopper suudab leida iga kujuga ristlõike jaoks. Seega suudab loodud programm kasutada analüütilisi meetodeid väga erinevate olukordade puhul sarnaselt lõplike elementide programmidele.

Kokkuvõttes on loodud programmil eelised nii Exceli kui ka ARSA üle. Ühest küljest on see oluliselt kiirem ja kasutajasõbralikum kui ARSA, kuid teiselt küljelt on sellega võimalik teostada arvutusi, mis Excelis muutuksid liiga keeruliseks, sarnanedes oma võimekuselt rohkem lõplike elementide programmile.

5.2 Jäikusseinte laotuste loomine automaatselt

Töös kasutati jäikusseinte laotuste loomise automatiseerimiseks evolutsioonilist optimeerimise algoritmi, mis proovib läbi tuhandeid variante ning leiab nendest parimad. Algoritmi loodud lahendused sarnanesid ehitusinseneri poolt loodud lahendustele, kuid need ei olnud efektiivsed ehituskonstruktsioonide seisukohast ega sarnanenud alati levinud jäikussüdamike kujudele. Kuigi algoritm suudab leida ligikaudselt sobilikke lahendusi, ei ole need üldiselt sobilikud kasutamiseks ilma suurte muudatusteta. Üheks põhjuseks on lihtsustatud arvutused, mis ei arvesta näiteks teist järku mõjudega. Kompaktsematel jäikussüdamikel nagu ristkülikukujulistel oleks sellisel juhul eelis, kuna ristuvad seinad stabiliseerivad üksteist ja vähendavad seinte efektiivset nõtkepikkust, tagades suurema kandevõime. Ilma selleta on algoritmi jaoks jäikusseinte loomisel tähtis ainult inertsimoment. Siiski ei ole inertsimomendi seisukohast saadud tulemused alati kõige efektiivsemad. Algoritm ei kasuta alati lahendusi, mis moodustaksid suurima inertsimomendiga ristlõikeid nagu I- või ristkülikukujulised. Tihtipeale satuvad seinad ristlõigete keskele, kus nad töötavad peamiselt ühes suunas.

Sarnaselt varasematele samal teemal tehtud uurimistöödele, kasutati ka siin seinte võimalike asukohtade määramiseks ruudustikku, mille iga külg on võimalik jäikussein. Autori arvates tekitab niisugune ülesande püstitus algoritmile aluse väga keeruliste ja suurte jäikusseinte gruppide loomiseks, mille kasutusvõimalused reaalsuses oleksid piiratud. Seega peab algoritmi suunama sobilikema jäikusseina kujude poole. Üheks võimaluseks võiks olla lisada tugevuskontrolli teist järku mõjude hindamine nagu on mainitud eelnevas lõigus, kuid selle mõju ei pruugi olla piisavalt suur, et algoritm hakkaks kompaktsemaid ristlõikeid tegema. Antud töös vähendati keeruliste jäikusseinte loomist täiendava piiranguga, mis suunas algoritmi kasutama lihtsamaid ristlõikeid. Saadud tulemustest selgus, et algoritm suutis luua jäikusseinu, mis vastasid piirangule, kuid

olid siiski ebapraktilise kujuga. Algoritm lõi väga palju jäikusseinu, mis koosnesid ühest pikast seinast, mille küljest hargnesid risti välja lühemad harud. Kuna keerulisuse määras ainult sirgete seinte arv, siis ei saavutanud eelnimetatud seinad piisavalt suurt keerulisuse astet, et algoritm neid väldiks.

5.3 Programmi kasutusvõimalused praktikas

Nagu varasemalt mainitud on töös loodud programmi peamised eelised võrreldes teiste arvutustarkvaradega selle kiirus, kasutusmugavus ja universaalsus. Võrreldes lõplike elementide tarkvaraga on loodud programmi arvutusteks kuluv aeg tänu analüütilistele meetoditele oluliselt lühem ning Rhino kasutamise tõttu ka kasutajasõbralikum, mistõttu on jäikusseinte muutmine äärmiselt lihtne ning nende tagajärjed koheselt nähtavad. Võrreldes arvutusprogrammidega nagu Excel või Mathcad on loodud programm oluliselt universaalsem ega vaja iga erineva olukorra jaoks spetsiaalsete valemite defineerimist. Seetõttu sobib programmi kasutuskohaks projekteerimise varajane staadium, kus ei ole veel kindlat lahendust lukus ning eesmärk on proovida läbi palju erinevaid variante ning leida neist kõige parem.

Jäikusseinte laotuste automaatne genereerimine sobib oma olemuselt kõige paremini keerulisemate hoonete puhul, kus sobiva lahenduse leidmine ei ole inseneri jaoks lihtne. See võimaldab inseneril proovida läbi palju rohkem variante kui tavalisel moel ning leida variante, mille peale ta ise ei pruugi tulla. Antud programm ei peaks olema kasutusel täpse lahenduse leidmiseks, vaid kaudsete lahenduste genereerimiseks, mida hiljem korrigeerida ja täpsemalt läbi arvutada. Korrapäraste hoonete puhul ei tasu automaatne jäikusseinte laotuste loomine ennast ära, sest ajakulu selle jaoks on päris suur, samas kui sobiv lahendus ise on üldiselt päris lihtne, mistõttu jõuab insener selleni ise kiiremini. Lisaks ei ole programmi loodud variandid ehituskonstruktsioonide seisukohast nii efektiivsed kui inseneri enda loodud, mistõttu vajaksid need hiljem käsitsi muutmist.

Autori arvates on kõige rohkem tuleviku potentsiaali programmis, mis arvutab jäikusseinte laotusi analüütiliste meetoditega, kuna võrreldes teiste programmidega on sellel varasemates lõikudes kirjeldatud eelised.

5.4 Tuleviku edasiarendused

5.4.1 Jäikusseinte analüüsimine ja dimensioneerimine

Antud töös ei suuda jäikusseinte arvutamise programm arvestada kõikide olukordadega, mis võivad osutuda kasutamisel tähtsaks. Nendest peamised on jäikussüdamike väändetugevus, avad jäikusseintes ja põikjõu ülekandmine vahelaelt.

Jäikussüdamike väändetugevusega arvestamine on vajalik juhul kui hoones esineb ainult üks jäikussüdamik. Sellisel juhul tuleb enamik konstruktsiooni väändetugevusest südamikust endast, mis suletud ristlõigete korral on märkimisväärne. Juhul kui hoones on mitmeid jäikusseinu võib jätta seinte enda väändetugevuse arvestamata, sest enamik

väändetugevusest saadakse üksikute seinte kaugusest jäikustsentrist nagu on näidatud joonisel 2.5. Sellisel juhul saadakse konservatiivsem tulemus.

Kui jäikusseintes on igal korrusel ühes ja samas kohas ava, siis võib see oluliselt mõjutada selle jäikust. Konservatiivselt võib sellises olukorras arvestada, et tegemist on kahe eraldiseisva seinaga, kuid sellisel juhul võib jäikus olla palju väiksem kui reaalsuses. Seega oleks vaja meetodit, millega saaks arvestada avade mõju seinte jäikusele, et saada täpsem tulemus, mis ei oleks liiga konservatiivne.

Horisontaalkoormus kandub jäikusseintele üle läbi hoone vahelagede, mistõttu peavad seinte ja vahelagede vahelised ühendused olema võimelised vastu võtma kogu tekkiva põikjõu. Olukorras, kus väikesele seina osale koondub suur osa vahelaes esinevast põikjõust, võib ühenduse kandevõime olla ületatud, mistõttu tuleks seda kontrollida.

Esimese ja teise toodud olukorraga arvestamiseks saaks kasutada Karoly Zalka poolt välja töötatud analüütilisi meetodeid, kuid mida ei jõutud antud töös rakendada. Antud meetodid on kirjeldatud tema raamatus *Structural Analysis of Multi-Storey Buildings* [29].

Analüütilised meetodid ei suuda siiski tõenäoliselt iga võimaliku olukorraga arvestada. Kuigi lõplike elementide meetodit ei ole mõistlik kasutada suure ajakulu tõttu kõikide arvutuste jaoks, siis on sellel siiski mõte analüütiliste arvutuste kontrollimisel. Seega võiks kiire töövoolu nimel olla võimalik saata jäikusseinte arvutusmudel lõplike elementide programmi, millega oleks võimalik arvutustulemusi võrrelda.

5.4.2 Jäikusseinte automaatne loomine

Evolutsioonilised algoritmid on head potentsiaalsete lahenduste leidmisel, kuid nende võimekus jõuda kindlate lahendusteni on oluliselt halvem ning vajavad rohkem iteratsiooni kui teised algoritmid. Kasutades ainult evolutsioonilist algoritmi peab olema generatsioonide arv suur, mis koos suure populatsiooniga vajab palju aega arvutusteks. Selle lahenduseks on välja pakkunud Deb [25] hübriidmeetodi, kus kasutatakse optimeerimiseks kahte algoritmi. Esmalt leitakse evolutsioonilise algoritmiga potentsiaalsed lahendused ning seejärel kasutatakse teist algoritmi, et need täpselt üles leida. Autori arvates võimaldaks see kasutada veel suuremaid populatsioone, võimaldades uurida veel rohkem erinevaid variante, kuid hoides samal ajal vajaliku ajakulu mõistlikkuse piirides.

Nagu varasemalt mainitud ei ole autori arvates ruudustiku kasutamine jäikusseinte laotuste loomisel ainuõige lahendus ning selle asemel võiks proovida alternatiivseid variante. Näiteks võiks ruudustiku asemel kasutada varasemalt defineeritud jäikusseinte kujusid, mida algoritm võib paigutada hoonesse. Selle eesmärgiks oleks tagada, et loodud jäikusseinad vastaksid enamlevinud kujudele.

Jäikusseinte optimeerimine on kõige tulusam kui korraga arvestatakse nii seinte paiknemist, kuju kui ka paksust. Antud töös käsitleti ainult kahte esimest, sest paksuse lisamisega suureneks võimalike variantide arv tohutult ja muudaks optimeerimise

aja mõttes ebapraktiliseks. Üheks võimaluseks seina paksuse integreerimiseks oleks kasutada optimeerimise algoritmi kaks korda. Esimene kord võetakse seinte paksus konstantseks ning leitakse sobivad variandid. Teisel korral võetakse esimesel optimeerimisel saadud tulemused esmaseks populatsiooniks ning lubatakse seinte paksusel muutuda. Sellises olukorras jääks esimese optimeerimise ajakulu samaks ning teise oma peaks olema oluliselt väiksem kuna sobilikud lahendused on juba ette antud. Siiski ei pruugi antud meetod leida uusi lahendusi, mida võiks seinte paksuse varieerumine anda.

KOKKUVÕTE

Antud töös loodi programm hoone jäikusseinte analüüsimiseks, tugevuskontrollide teostamiseks ning automaatselt erinevate laotuste genereerimiseks, kasutades selleks Rhinocerous 3D joonestustarkvara ning sellega kaasnevat visuaalse programmeerimise lisandit Grasshopper. Programmi eesmärk on vähendada inseneri ajakulu sobivate jäikusseinte laotuste leidmiseks. Kuna programm on loodud spetsiifiliselt jäikusseinte jaoks, kasutades analüütilisi meetodeid, siis on sellega töötamine oluliselt kiirem kui näiteks praktikas levinud lõplike elementide programmidega. Samas säilib programmil võime tulla toime erinevate jäikusseinte kujudega, mis analüütiliste meetodite puhul on üldiselt raske. Antud programmi kasutati ka evolutsioonilise optimeerimise algoritmiga, et genereerida erinevaid jäikusseinte laotusi, võimaldades inseneril automatiseerida osa oma tööst.

Töö esimeses peatükis antakse ülevaade varasematest uurimistöödest, mis käsitlevad jäikusseinte laotuste optimeerimist erinevate algoritmidega. Peatükis 2 kirjeldatakse jäikusseinte projekteerimiseks loodud programmi. Programm kasutab arvutuste teostamiseks analüütilisi meetodeid, mistõttu on arvutusteks kuluv aeg oluliselt väiksem kui lõplike elementide meetodiga tehtav analoogne arvutus. Lisaks kasutab programm jäikusseinte tugevuskontrollide teostamisel lihtsustatud surutud ristlõike kontrolli, mis suudab toime tulla mitmesuguste jäikusseinte kujudega. Peatükis 3 kirjeldatakse jäikusseinte laotuste genereerimist evolutsioonilise optimeerimise algoritmiga. Kasutades Grasshopperi jaoks loodud mitme eesmärgiga evolutsioonilise optimeerimise liidest Wallacei, proovib algoritm läbi erinevaid jäikusseinte laotusi, kuni leiab nendest parimad. Optimeerimisel kasutati kolme eesmärki, millest esimeseks oli minimeerida jäikusseinte maht, teiseks tagada kandekonstruktsiooni tugevuse ja deformatsioonidega seotud piirangud ning kolmandaks vähendada algoritmi soodumust luua väga keerulise ristlõikega jäikusseinu. Peatükis 4 katsetakse jäikusseinte laotuste genereerimist erinevate põhiplaanidega hoonete peal. Algoritmi loodud lahendused olid sarnased sobilikele lahendustele, kuid nende materjali kasutus oli ebaefektiivne ning jäikusseinte kujud erinesid tihtipeale oluliselt enamlevinud kujudest. Kuna programmi loodud tulemused vajavad täiendavat korrigeerimist ning algoritmi ajakulu on ligikaudu pool ööpäeva, siis on jäikusseinte laotusi mõistlik genereerida ainult keeruliste põhiplaaniga hoonete puhul, kus ei ole head jäikusseinte laotused koheselt tuvastatavad. Viimasena analüüsitakse peatükis 5 loodud programmi nõrkusi ja tugevusi, millised on selle kõige paremad kasutuskohad ning kuidas seda tulevikus edasi arendada.

Autori arvates on jäikusseinte analüüsi ja dimensioneerimise programmil analüütiliste meetoditega suur potentsiaal ning selle täiustamisel võiks saada sellest hea tarkvara inseneridele, millega kiiresti proovida läbi erinevaid jäikusseinte skeeme ning saada kohest tagasisidet muudatustele. Automaatselt loodud jäikusseinte laotused on sarnased sobilikele, kuid need ei ole üldiselt efektiivsemad kui inseneri loodud. Seega jääks programmi peamiseks kasutuskohaks keerulised hooned, kus insener ei suuda kergelt tuvastada parimat jäikusseinte laotust.

ABSTRACT

In this thesis a program was created for the analysis, design and automatic generation of different shear wall layouts. The modeling software Rhinoceros 3D with the included visual programming plugin Grasshopper were used to achieve this task. The goal of the program is to reduce the nessecary time an engineer needs to find a suitable shear wall layout for a building. Because the program is created specifically for the design of shear walls and uses analytical methods, it is considerably faster than commonly available calculation software utilizing finite element methods. Furthermore the program can still deal with complex shear wall shapes, which is more difficult for analytical methods than for finite element ones. Lastly the created program was used in conjunction with an evolutionary optimization algorithm to generate different shear wall layouts allowing an engineer to automate part of his work.

In the first chapter a literature review is given of previous works dealing with the optimization of shear wall layouts. In chapter 2 the created program for analysing and dimensioning shear walls is described. The program uses analytical methods for the calculations which requires less time than a similar calculation done by finite element programs. In addition the program uses a simplified compressed member check in designing the shear walls which can be used for every kind of shaped cross section. In chapter 3 the method used to generate shear wall layouts is described. A multi objective evolutionary optimization algorithm is used which comes as an additional plugin for Grasshopper named as Wallacei. Three goals were used in the optimization algorithm. The first one was to minimize the area of the shear walls, the second one was used so the shear walls would pass their strength checks and the third one directed the algorithm toward using shear walls with a simpler cross section. In chapter 4 a number of shear wall layouts are generated for differently shaped floor plans. The layouts produced by the algorithm were similar to appropriate solutions, but the cross sections of the shear walls often had an unefficient shape from a structural viewpoint. As the automatically generated shear wall layouts require manual adjustment by an engineer and the algorithm requires approximately 12 hours to complete the optimization, it is only efficient to use it for buildings with very complicated floor plans where a good solution is not obvious. Finally chapter 5 discusses the strengths and weaknesses of the created program, when are the best cases for using it and how it could be improved in the future.

In the authors opinion there is a great potential for the program analyzing shear wall layouts using analyticial methods and with further improvements it could become a good software for engineers to try out different shear wall layouts. As the automatically generated shear wall layouts are similar to suitable ones but not as efficient then it should only be used for very complex floor plans.

KASUTATUD KIRJANDUSE LOETELU

[1] D. S. Philps, "Shear Wall Layout Optimization in Coordination with Architectural Floor Plans," M.S. thesis, Massachusetts Institute of Technology, May 2022.

[2] H. Lou, J. Ye, F. Jin, B. Gao, Y. Wan, and G. Quan, "A practical shear wall layout optimization framework for the design of high-rise buildings," *Structures*, vol. 34, pp. 3172–3195, Dec. 2021, ISSN: 23520124. DOI: 10.1016/j.istruc.2021.09.038.

[3] M. Afzal, Y. Liu, J. C. Cheng, and V. J. Gan, "Reinforced concrete structural design optimization: A critical review," *Journal of Cleaner Production*, vol. 260, no. 120623, Jul. 2020, ISSN: 09596526. DOI: 10.1016/j.jclepro.2020.120623.

[4] H.-L. Chi, X. Wang, and Y. Jiao, "BIM-Enabled Structural Design: Impacts and Future Developments in Structural Modelling, Analysis and Optimisation Processes," *Archives of Computational Methods in Engineering*, vol. 22, no. 1, pp. 135–151, Jan. 2015, ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-014-9127-7.

[5] "Eurokoodeks 2: Betoonkonstruktsioonide projekteerimine. Osa 1-1: Üldreeglid ja reeglid hoonetele."

[6] Y. Zhang and C. Mueller, "Shear wall layout optimization for conceptual design of tall buildings," *Engineering Structures*, vol. 140, pp. 225–240, Jun. 2017, ISSN: 01410296. DOI: 10.1016/j.engstruct.2017.02.059.

[7] H. Dehnavipour, H. Meshki, and H. Naderpour, "Torsion-based layout optimization of shear walls using multi-objective water cycle algorithm," *Advances in Structural Engineering*, vol. 24, no. 13, pp. 3030–3042, Oct. 2021, ISSN: 1369-4332, 2048-4011. DOI: 10.1177/13694332211017999.

[8] S. Talatahari and M. Rabiei, "Shear wall layout optimization of tall buildings using Quantum Charged System Search," *Frontiers of Structural and Civil Engineering*, vol. 14, no. 5, pp. 1131–1151, Oct. 2020, ISSN: 2095-2430, 2095-2449. DOI: 10.1007/s11709-020-0660-1.

[9] X. Zhou, L. Wang, J. Liu, G. Cheng, D. Chen, and P. Yu, "Automated structural design of shear wall structures based on modified genetic algorithm and prior knowledge," *Automation in Construction*, vol. 139, no. 104318, Jul. 2022, ISSN: 09265805. DOI: 10.1016/j.autcon.2022.104318.

[10] H. Lou, B. Gao, F. Jin, Y. Wan, and Y. Wang, "Shear wall layout optimization strategy for high-rise buildings based on conceptual design and data-driven tabu search," *Computers & Structures*, vol. 250, no. 106546, Jul. 2021, ISSN: 00457949. DOI: 10.1016/j.compstruc.2021.106546.

[11] M. Jinjie, S. Qingxuan, and H. Zhijian, "Optimal Design of Tall Residential Building with RC Shear Wall and with Rectangular Layout," *International Journal of High-Rise Buildings*, vol. 3, no. 4, pp. 285–296, Dec. 2014.

[12] M. Cote, "Shear Wall Layout Optimization of Dynamically Loaded Three- Dimensional Tall Building Structures," M.S. thesis, Massachusetts Institute of Technology, 2017.

[13] N. Haseeb, "Multi-Objective Optimization of Vertically Mixed Lateral Systems," M.S. thesis, Massachusetts Institute of Technology, 2017.

[14] S. Tafraout, N. Bourahla, Y. Bourahla, and A. Mebarki, "Automatic structural design of RC wall-slab buildings using a genetic algorithm with application in BIM environment," *Automation in Construction*, vol. 106, no. 102901, Oct. 2019, ISSN: 09265805. DOI: 10.1016/j.autcon.2019.102901.

[15] Ş. Atabay, "Cost optimization of three-dimensional beamless reinforced concrete shear-wall systems via genetic algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3555–3561, Mar. 2009, ISSN: 09574174. DOI: 10.1016/j.eswa.2008.02.004.

[16] V. J. Gan, C. Wong, K. Tse, J. C. Cheng, I. M. Lo, and C. Chan, "Parametric modelling and evolutionary optimization for cost-optimal and low-carbon design of high-rise reinforced concrete buildings," *Advanced Engineering Informatics*, vol. 42, no. 100962, Oct. 2019, ISSN: 14740346. DOI: 10.1016/j.aei.2019.100962.

[17] M. J. Fadaee and D. E. Grierson, "Design optimization of 3D reinforced concrete structures having shear walls," *Engineering with Computers*, vol. 14, no. 2, pp. 139–145, Jun. 1998, ISSN: 0177-0667, 1435-5663. DOI: 10.1007/BF01213587.

[18] S. R. Hoseini Vaez and H. Shahmoradi Qomi, "Bar Layout and Weight Optimization of Special RC Shear Wall," *Structures*, vol. 14, pp. 153–163, Jun. 2018, ISSN: 23520124. DOI: 10.1016/j.istruc.2018.03.005.

[19] W. Liao, X. Lu, Y. Huang, Z. Zheng, and Y. Lin, "Automated structural design of shear wall residential buildings using generative adversarial networks," *Automation in Construction*, vol. 132, no. 103931, Dec. 2021, ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103931.

[20] "Eurokoodeks 1: Ehituskonstruktsioonide koormused. Osa 1-4: Üldkoormused. Tuulekoormus."

[21] B. Mosley, J. Bungey, and R. Hulse, *Reinforced Concrete Design to Eurocode 2*, 7. ed. Basingstoke: Palgrave Macmillan, 2012, 484 pp., ISBN: 978-0-230-30285-3.

[22] M. Makki, M. Showkatbakhsh, and Y. Song, *Wallacei*, version 2.7.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, ISSN: 1089778X. DOI: 10.1109/4235.996017.

[24] C. A. C. Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Engineering Optimization*, vol. 32, no. 3, pp. 275–308, Jan. 2000, ISSN: 0305-215X, 1029-0273. DOI: 10.1080/03052150008941301.

[25] K. Deb, "Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction," in *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing*, 1st ed., Springer London, 2001, ISBN: 978-0-85729-652-8.

[26] A. Gardner, *Stability of Buildings. Part 3, Shear Walls*. London: Institution of Structural Engineers, 2015, ISBN: 978-1-5231-0247-1.

[27] *Microsoft Excel*, Microsoft, 2023.

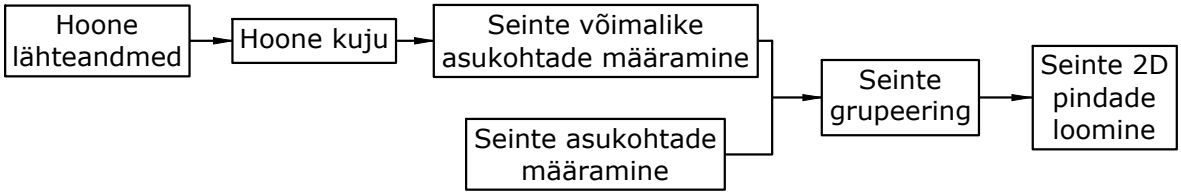
[28] *Robot Structural Analysis Professional*, Autodesk, 2023.

[29] K. A. Zalka, *Structural Analysis of Multi-Storey Buildings*, Second edition. Boca Raton: CRC Press, 2020, ISBN: 978-0-367-35025-3.

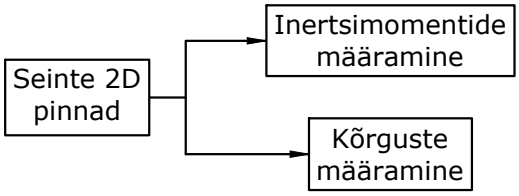
LISAD

Lisa 1 loodud programmi diagramm

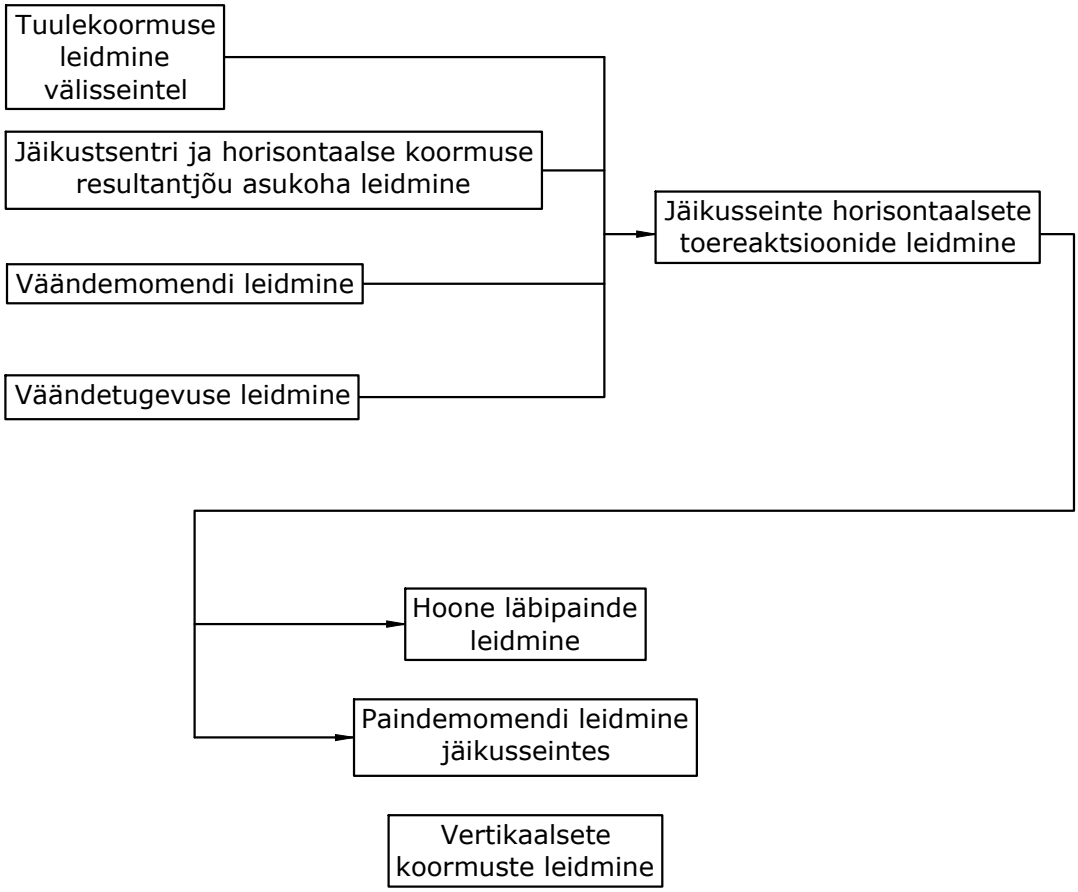
HOONE GEOMEETRIA (PEATÜKK 2.2)



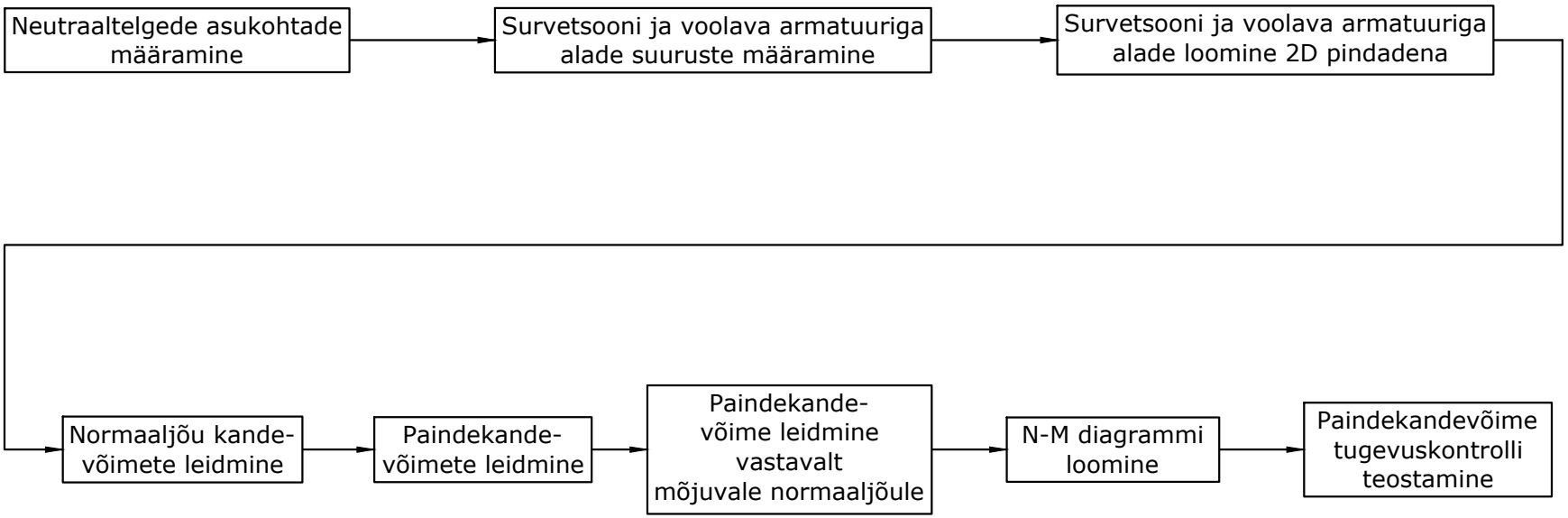
JÄIKUSSEINTE GEOMEETRILISED OMADUSED



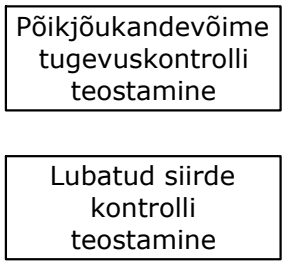
JÄIKUSSEINTE SISEJÕUDUDE LEIDMINE (PEATÜKK 2.3)



PAINDEKANDEVÕIME TUGEVUSKONTROLLI TEOSTAMINE (PEATÜKK 2.4.3)



MUUDE TUGEVUSKONTROLLI TEOSTAMINE (PEATÜKK 2.4.1 JA 2.4.2)



OPTIMEERIMISE ALGORITM (PEATÜKK 3)