TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Chris Matsiselts

Verifitseerimistööriistade kombineerimine andmejooksude tõhusamaks avastamiseks

Bakalaureusetöö (9 EAP)

Juhendajad: Vesal Vojdani, PhD
Karoliine Holter, MSc

Tartu 2025

Verifitseerimistööriistade kombineerimine andmejooksude tõhusamaks avastamiseks

Lühikokkuvõte:

Töö eesmärkideks on luua töövahend CoOpeRace CLI, mis kasutab verifitseerimiskoostöö põhimõtteid, et kombineerida olemasolevaid verifitseerimistööriistu ühes metaverifitseerijas ning leida parim konfiguratsioon olemasolevatest verifitseerimistööriistadest andmejooksude avastamiseks. Töös antakse ülevaade tarkvara verifitseerimisest, SV-COMP võistlusest ning CoOpeRace projektist. Lisaks tutvustatakse töö käigus loodud tööriistu ning CoOpeRace CLI abil tehtud katsete tulemusi.

Võtmesõnad:

CoOpeRace, veriftiseerimiskoostöö, andmejooks

CESCS: P175 Informaatika, süsteemiteooria

Combining Verification Tools for More Effective Data Race Detection

Abstract:

The purpose of this Bachelor's thesis is to create a tool, CoOpeRace CLI, that utilizes the principles of verification cooperation to combine existing verification tools into a single meta-verifier and to find the best configuration of available verification tools for detecting data races. The work provides an overview of software verification, the SV-COMP competition, and the CoOpeRace project. Additionally, the tools developed during the work and the results of experiments conducted using the CoOpeRace CLI are presented.

Keywords:

CoOpeRace, cooperative verification, data race

CESCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus....4

1. Mõisted ja terminid....5

2. Taust....7

2.1 Tarkvara automaatveriftiseerimine....7

2.2 SV-COMP....8

2.3 Verifitseerimiskoostöö....8

2.4 CoOpeRace....9

3. CoOpeRace CLI....11

3.1 Arenduskäik....11

3.2 SV-COMP 2025 tulemused....13

4. Analüüs....16

4.1 SV-COMP tulemuste analüüs parimate kombinatsioonide leidmiseks....16

4.2 Kombinatsioonide katsetamise tulemused piiratud mäluga....18

4.3 Kombinatsioonide katsetamise tulemused suurema mäluga....22

4.4 Tulemused....24

Kokkuvõte....26

Viidatud kirjandus....27

Lisad....29

I Valminud tööriistad ja analüüsiks kogutud toorandmed....29

II Verifitseerimistööriistade kombinatsioonide teoreetilised skoorid....30

III Kombinatsioonide katsetamise tulemused piiratud mäluga....31

IV Kombinatsioonide katsetamine tulemused suurema mäluhulga korral....33

V Litsents....35

Sissejuhatus

Tänapäeva tarkvarasüsteemide keerukus on märkimisväärselt kasvanud, mistõttu suureneb ka vajadus töökindlate ja turvaliste rakenduste järele. Eriti oluline on tuvastada ja ennetada andmejookse (ingl *data race*), mis tekivad mitme lõime samaaegsel ligipääsul jagatud andmetele ilma piisava sünkroniseerimiseta. Sellised olukorrad võivad põhjustada tarkvara ettearvamatut ja ebastabiilset käitumist, mis on sageli raskesti reprodutseeritav. Andmejooksude ennetamine on oluline, kuna need võivad viia süsteemi sisemiste vigade, andmete rikkumise või isegi turvariskideni. Taoliste probleemide ennetamiseks saab kasutada formaalseid verifitseerimismeetodeid, mis võimaldavad tarkvara käitumist analüüsida ilma seda reaalselt käivitamata [1].

Kuigi andmejooksude avastamiseks on välja töötatud mitmeid verifitseerimistööriistu, on igal neist oma spetsiifilised tugevused ja piirangud. Verifitseerimiskoostöö eesmärk on ületada need piirangud, ühendades erinevate tööriistade tugevaimad küljed ühtsesse analüüsiraamistikku ehk metaverifitseerijasse, mis võimaldab tuvastada rohkem vigu, hoides valede tuvastuste arvu madalana [2].

Selle töö esimeseks eesmärgiks oli välja töötada tööriist, mis rakendab verifitseerimiskoostöö põhimõtteid mitmelõimelistes programmides andmejooksude tõhusamaks tuvastamiseks. Loodav lahendus tugines olemasolevate verifitseerimistööriistade tugevustele ning nende kombineeritud kasutamisele. Töö teine eesmärk oli uurida erinevaid tööriistade kombinatsioone ja leida kõige tõhusam kombinatsioon andmejooksude avastamiseks ning välistamiseks.

Bakalaureusetöö koosneb neljast osast. Esimeses osas on kirjeldatud tööga seotud mõisteid ja termineid. Sellele järgnevalt on antud ülevaade tarkvara verifitseerimisest, võistlusest SV-COMP ning CoOpeRace projektist. Kolmandas osas on tutvustatud töö käigus valminud tööriista CoOpeRace CLI ning selle arenduskäiku. Neljandas osas on esitatud uurimise käigus saadud tulemused ning nende põhjal on välja toodud parimad kombinatsioonid andmejooksude tuvastamiseks.

Lõputöö koostamisel kasutati tehisintellektil põhinevat ChatGPT (GPT-4, versioon 4-turbo) keelemudelit eelkõige õigekirja kontrollimiseks ning akadeemilise sõnastuse täpsustamiseks ja sujuvamaks muutmiseks.

1. Mõisted ja terminid

Andmejooks (ingl *data race*) on olukord mitmelõimelises programmis, kus erinevad lõimed võivad samal ajal kasutada ühte mäluasukohta ning üks kasutustest on kirjutamine [3].

Automaatverifitseerimine on protsess, mis kasutab formaalset loogikat, matemaatilisi mudeleid ja automatiseeritud tööriistu, et tõestada või ümber lükata tarkvarasüsteemi teatud omadusi [4].

Korrektsustõend (ingl *correction witness*) on verifitseerimistulemuste vorm, mis pakub tõendusmaterjali või vihjeid selle kohta, et programm vastab antud tunnusele [5].

Lõim (ingl *thread*) on programmiosa, mida saab täita paralleelselt teiste programmiosadega [6].

Muteks (ingl *mutex*) on lukustusvahend, mis reguleerib ressursi jagamist mitme programmiosa vahel [6].

Mõõtlusalus (ingl *benchmark*) on norm või väärtuste kogum, mille abil mõõta kvaliteeti [6].

Püsiprogrammeeritud (ingl *hardcoded*) on programmi osa, mis on realiseeritud kujul, kus sellel puuduvad muutmise võimalused [6].

Rikkumistõend (ingl *violation witness*) on verifitseerimistulemuste vorm, mis pakub tõendusmaterjali või vihjeid selle kohta, et programm rikub antud tunnust [7].

Skript (ingl *script*) on väike kompileerimata käsujada, mida interpreteerib või täidab teine programm [6].

Staatiline analüüs, teise nimega staatiline programmianalüüs, on tarkvaraarenduses kasutatav tehnika, mille abil tuvastatakse võimalikke vigu ning turvanõrkusi ilma programmi käivitamata [8].

Tarkvara verifitseerimine (ingl *software verification*) on valdkond, kus uuritakse meetodeid ning luuakse tööriistu, mille abil tõestada programmi omadusi [2].

Tupik (ingl *deadlock*) on olukord, kus programmi töö seiskub, sest lõimed ootavad ressurssi, mis on antud teis(t)ele lõime(de)le [6].

Täitmismootor (ingl *execution engine*) on virtuaalmasin programmi täitmiseks, kontekstist sõltuvas tähenduses [6].

Valdkonnaspetsiifiline keel (ingl *domain-specific language*) on programmeerimiskeel, mis on keskendunud konkreetsele valdkonnale [9].

Verifitseerimiskoostöö (ingl *cooperative verification*) on tarkvaraverifitseerimismeetod, mis kasutab kombinatsiooni lähenemisviise, et saada tulemus, jättes olemasolevad tööriistad enamasti muutmata ning kasutades ära iga tööriista individuaalseid tugevusi [2].

Ülem-abiline arhitektuur (ingl *master-helper architecture*) on arhitektuuriline muster, kus töö või ülesannete lahendamise ning alamülesannete delegeerimise eest vastutab peakomponent (*master*) ning abikomponendid (*helper*) töötavad paralleelselt ilma omavahelise koostööta ja täidavad delegeeritud ülesandeid [10].

2. Taust

Selles peatükis on antud ülevaade tarkvara automaatverifitseerimisest ning verifitseerimiskoostööst. Lisaks on kirjeldatud tarkvara verifitseerimise võistlust SV-COMP ning verifitseerimiskoostöö projekti CoOpeRace.

2.1 Tarkvara automaatveriftiseerimine

Koodi kirjutamisel on inimlik teha vigu ning enamik neist vigadest on kompilaatori või arenduskeskkonna poolt kergesti tuvastatavad, suunates arendaja tähelepanu vea täpsesse kohta. Mõningad turvanõrkused ei ole aga nii silmnähtavad ning sellised vead esinevad sageli haruldastes olukordades, millega arendaja ei pruugi arvestada ning mis võivad ka traditsioonilise testimise käigus tuvastamata jääda. Selliste raskesti leitavate turvanõrkuste avastamiseks saab kasutada automaatverifitseerimist [1].

Automaatverifitseerimist kasutatakse põhiliselt tarkvara kvaliteedi parandamiseks ning defektide tuvastamiseks enne tarkvara käivitamist. Automaatverifitseerimises on üldiselt eelistatud tulemuste usaldusväärsust, mille tagajärjel esineb automaatverifitseerija tulemustes palju valepositiivseid tulemusi [4]. Valepositiivsete tulemuste all on mõeldud tulemusi, kus analüsaator tuvastab vea seal, kus seda tegelikkuses ei eksisteeri. Sellegipoolest kasutatakse automaatverifitseerimist, et kontrollida süsteemide ohutust [11]. Üks nõrkus, mida automaatverifitseerimisega üritatakse vältida, on andmejooks.

Andmejooksud võivad esineda mitmelõimelistes programmides, põhjustades vigu ja rikkeid, mis võivad olla kulukad ning raskesti parandatavad. Lisaks võivad andmejooksud põhjustada ka mittedeterminismi, kuna erinevate lõimede täpsed täitmisajad võivad viia erinevate tulemusteni. See omakorda muudab andmejooksude reprodutseerimise keeruliseks, sest dünaamilise analüüsi meetodid sõltuvad konkreetsetest täitmise tingimustest, mis võivad igal käivitamisel erineda [12].

Andmejooksude vältimiseks kasutatakse enamasti mutekseid, mis võimaldavad piirata lõimede ligipääsu ressurssidele juhul, kui üks lõimedest seda juba kasutab. Muteksite kasutamisel tuleb aga olla tähelepanelik, sest nende ebakorrektsel kasutamisel võib siiski andmejooks tekkida. Alternatiivselt võib tekkida ka tupik, kui muteks pole peale ühise ressursi kasutamist lõime poolt korrektselt vabastatud [13].

2.2 SV-COMP

Võistluse SV-COMP kirjeldamisel on refereeritud võistluse koduleheküljel [14] olevaid materjale ning Dirk Beyeri artiklit võistluse 13. väljaande kohta [15].

Software Verification Competition ehk SV-COMP on iga-aastaselt toimuv rahvusvaheline võistlus, mille eesmärgiks on hinnata ja võrrelda tarkvara verifitseerimise tööriistade kvaliteeti. Võistlus hõlmab standardiseeritud ülesannete kogumikku, mille abil saab erinevaid tööriistu võrdsetel tingimustel võrrelda. Lisaks annab võistlus võimaluse esitleda kasutatud tööriistu ja meetodeid, tõstes sellega tarkvara verifitseerimise arengu nähtavust. Võistluse ülesanded on jagatud kategooriatesse, mis on omakorda jaotatud alamkategooriatesse. See jaotus on tehtud vastavalt ülesannetes olevate programmidele ja verifitseeritavatele omadustele. Jaotuse eesmärgiks on paremini hinnata erinevate tööriistade tugevusi ning nõrkusi erinevate ülesannete lahendamisel.

SV-COMP 2024 oli võistluse 13. väljaanne ning sellest võttis osa 76 tööriista, millest 59 olid verifitseerimistööriistad ning 17 olid rikkumis- ja korrektsustõendite valideerimistööriistad. Võistlusel oli kokku 30 300 ülesannet programmeerimiskeeles C ning üldarvestuses osutus parimaks verifitseerimistööriistaks UAutomizer. Võistlusel said tööriistad osaleda ka väljaspool arvestust. Väljaspool arvestust saavad näiteks osaleda tööriistad, mis kasutavad oma verifitseerimisprotsessis teisi võistlusel osalevaid verifitseerimistööriistu või nende osi.

2.3 Verifitseerimiskoostöö

Tarkvara verifitseerimises on viimastel aastatel toimunud tohutu areng, mille tulemusel on muutunud verifitseerimistööriistad täpsemaks ning võimekamaks käsitlema erinevate programmeerimiskeelte funktsionaalsust [2]. Arengust sõltumata ei ole olemas ühte kõikehõlmavat tarkvara verifitseerimise tööriista, mis lahendaks iga probleemi. Igal tehnikal on oma tugevused ja nõrkused ning erinevad tööriistad kasutavad täpse tulemuse saavutamiseks mitme lähenemisviisi kombinatsioone. Erinevate lähenemisviiside kombineerimiseks on kaks meetodit [16]:

- Tihe kombineerimine: Erinevad lähenemisviisid töötavad tihedalt koos, mis suurendab tööriista efektiivsust. Ometi iga uue vahendi lisamisel tuleb see põhjalikult integreerida, mis omakorda muudab süsteemi keerukamaks ning selle edasiarendamise aeganõudvaks.

- Lõtv kombineerimine: Vahendeid kombineeritakse nii, et erinevad vahendid omavahel ei suhtle. Selline lähenemine lihtsustab uue vahendi lisamist, kuid vähendab tööriista efektiivsust, sest vahetulemusi, mida kasutavad mitu vahendit, tuleb igal vahendil eraldi arvutada.

Verifitseerimiskoostöö eesmärk on kombineerida tiheda ja lõdva kombineerimise tugevusi [16]. Kuigi verifitseerimiskoostöö on loomulik edasiareng lähenemisviiside kombineerimiseks, toob see endaga kaasa mitmeid väljakutseid, millest suurimaks on vahetulemuste kommunikatsioon. Samas mainitakse, et koostöö saavutamiseks on vaja, et tööriistad toodaksid vahetulemusi, mida saab teiste tööriistadega jagada, ning mida saavad teised verifitseerijad oma tööprotsessis kasutada [10]. Kaks verifitseerimiskoostööl põhinevat tööriista on näiteks CoVeriTeam ning CoVEGI.

CoVeriTeam arendati näitamaks, et verifitseerimisvahend, mis põhineb erinevate verifitseerijate kombineerimisel ning omavahelisel koostööl, on üks paljulubav teostus verifitseerimiskoostööst. CoVeriTeam loomisel arendati uus valdkonnaspetsiifiline keel ning täitmismootor, mille abil on võimalik kombineerida erinevaid verifitseerimisvahendeid, jättes kasutatavad vahendid muutmata kujule. Need arendused võimaldavad hõlpsasti uusi tööriistu ja nende kombinatsioone katsetada, vabastades kasutajad tööriistade seadistamisest ning skriptide kirjutamisest [16].

CoVEGI on koostööraamistik, mis lähtub samuti verifitseerimiskoostöö põhimõtetest. See kasutab ülem-abiline arhitektuuri, st ühte peamist verifitseerijat ehk ülemat, mis suudab invariantide genereerimise abistavatele verifitseerimistööriistadele delegeerida ning jätkab verifitseerimist saadud tulemuste abil. Abilised saadavad oma vahetulemused ülemale korrektsustõendite kujul, sest nende formaat on standardiseeritud ning paljud verifitseerijad juba toodavad korrektsustõendeid. Haltermann ja Wehrheim toovad välja, et CoVEGI raamistiku arendamise käigus katsetati 14 erinevat ülem-abiline kombinatsiooni ning tulemuseks oli, et koostööpõhine verifitseerimine parandab peamise verifitseerija võimekust ilma liigse ajakulu suurenduseta. Olulise tulemusena toodi välja, et mõnel kombinatsioonil suudeti koostöö tulemusena verifitseerimisaega vähendada [10].

2.4 CoOpeRace

Cooperative data race verification (CoOpeRace) on projekt, mille raames arendatakse koostööd erinevate tarkvara verifitseerijate vahel eesmärgiga parandada staatilise analüüsi

kasutusvõimekust erasrektoris. Eesmärgi täitmiseks on loodud mitmeid tööriistu, mis kombineerivad olemasolevaid verifitseerijaid verifitseerimiskoostöö põhimõtetel. Verifitseerijaid kombineeritakse viisil, mis võimaldab luua usaldusväärse ja tõhusa tööriista, kasutades alamtööriistade tugevusi, et pakkuda kasutajale parimaid tulemusi. Projekt keskendub andmejooksudele ning nende avastamisele.

Varasemalt on CoOpeRace projekti raames loodud veebirakendus, kus on võimalik üles laadida kasutaja programmi, mida analüüsib mitu erinevat verifitseerijat ning mis kuvab kasutajale informatsiooni võimalike andmejooksude asukohtadest. Veebirakendus kasutab kolme verifitseerijat: Goblint[1], Locksmith[2] ja Relay[3] ning kuvab kasutajale iga tööriista analüüsi tulemused. Veebirakendusel on ühtlasi ka funktsionaalsus, kus kasutajad saavad lisaks olemasolevatele verifitseerijatele neid juurde lisada [17].

[1] https://goblint.in.tum.de
[2] https://www.cs.umd.edu/projects/PL/locksmith/
[3] https://cseweb.ucsd.edu/~lerner/papers/relay.pdf

3. CoOpeRace CLI

Üheks bakalaureusetöö eesmärgiks oli arendada tööriist CoOpeRace CLI, mis võimaldab kombineerida erinevaid olemasolevaid verifitseerimistööriistu ühtsesse raamistiku. Arendatavas tööriistas peab olema võimalik hõlpsasti, st koodi originaalkuju säilitades, erinevaid verifitseerijaid kombineerida. Lisaks peab CoOpeRace CLIs olema kasutajal võimalik alamtööriistade käivitamise järjekorda paindlikult määrata ning konfigureerida, milliseid analüüsitulemusi aktsepteeritakse.

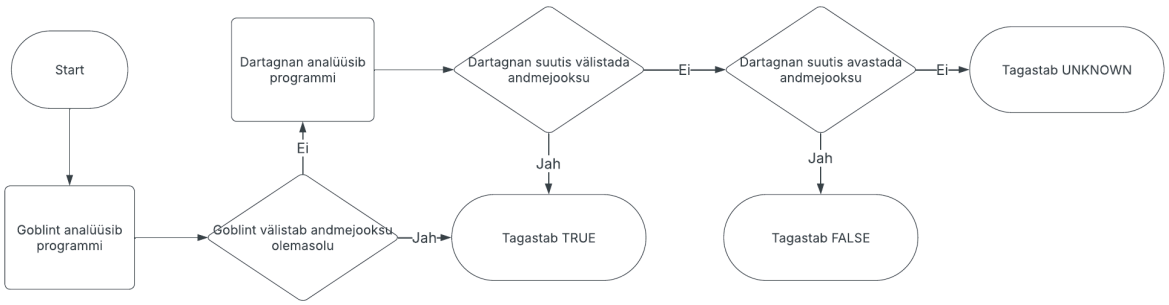
Kuna tegemist oli eksperimentaalse ja agiilse tarkvaraarendusega, ei olnud projekti alguses võimalik täpset nõuete kogumit määratleda – vajadused ja funktsionaalsus täpsustusid eesmärgipõhise arendusprotsessi käigus vastavalt katsetustele ja nende tulemustele.

3.1 Arenduskäik

CoOpeRace CLI on veriftiseerimiskoostöö põhimõtteid kasutav raamistik, mis kombineerib SV-COMP võistlusel osalevaid staatilise analüüsi tööriistu. CoOpeRace CLI valminud versioon arendati eesmärgiga kaardistada parimad kombinatsioonid tööriistadest, mis tuvastavad analüüsitavas programmis andmejooksude olemasolu või puudumist.

Esimeses versioonis kasutas CoOpeRace CLI verifitseerimiskoostööks kahte tööriista: Goblint ja Dartagnan[4]. Mõlemad tööriistad said valitud eelneva SV-COMP võistluse tulemuste baasil. Goblinti tugevuseks on vastavalt tõsipositiivsete tulemuste ja Dartagnanil tõsinegatiivsete tulemuste korrektne tuvastamine. CoOpeRace CLI raamistikus töötasid nad järjestikku: esimesena analüüsis programmi Goblint ning seejärel vajadusel ka Dartagnan. Kui Goblint suutis andmejooksu olemasolu välistada, tagastas CoOpeRace CLI vastuse, et programmis andmejookse ei leidu (TRUE). Vastasel korral alustas programmianalüüsi Dartagnan. Kui Dartagnan suutis potentsiaalse andmejooksu avastada, tagastas CoOpeRace CLI, et programmi käivitamisel võib tekkida andmejooks (FALSE). Kui Goblint ei suutnud andmejooksu välistada ega Dartagnan selle olemasolu välistada ega tuvastada, tagastas CoOperace CLI, et vastus on teadmata (UNKNOWN), st, et CoOpeRace CLI ei olnud suuteline antud programmis andmejooksu ei kinnitama ega välistama. Joonis 1 illustreerib raamistiku kirjeldatud töökäiku.

[4] https://github.com/hernanponcedeleon/Dat3M



Joonis 1. Tööriista CoOpeRace esimese versiooni töökäiku seletav juhtvoograaf

Selle versiooni puudused olid, et CoOpeRace polnud modulaarne: Goblint ja Dartagnan olid püsiprogrammeeritud ning nende eemaldamine ja uute tööriistade lisamine oli aeganõudev. Lisaks suutis CoOpeRace oma alamtööriistu käitada ainult järjestikku, piirates võimalike kombinatsioonide hulka.

Teises arengutsüklis eemaldati Goblinti ja Dartagnani püsiprogrammeeritud versioonid ning võeti kasutusele Benchexec[5] (versioon 3.27), mis võimaldas programmi modulaarselt disainida ning seega lihtsustas verifitseerimistööriistade lisamist ja eemaldamist. Tööriistale arendati funktsionaalsus käitada alamtööriistu paralleelselt, mis koos järjestikku käitamisega võimaldas teostada suurema koguse kombinatsioone, parandades tööriista kasutusvõimalusi. Seetõttu sai selles versioonis kombinatsioonide loomisel lisaks Goblintile ja Dartagnanile kasutada ka tööriistu Deagle[6], ULTIMATE Automizer[7] ning ULTIMATE GemCutter[8]. Kombinatsioonide teostamisprotsessi lihtsustamiseks loodi konfiguratsioonide haldussüsteem, mis võimaldas erinevaid kombinatsioone kirjeldada *JavaScript Object Notation* (JSON) formaadis, jättes programmikoodi enda muutmata kujule. Konfiguratsioon tähendab siin kontekstis seadistust, mis täpsustab, milliseid tööriistu ning mis järjekorras neid kasutatakse. Lisafunktsionaalsusena sai konfiguratsioonide haldussüsteemi osana teostatud ka võimalus täpsustada, kas aktsepteerida alamtööriista puhul nii TRUE kui ka FALSE tulemus, või ainult ühte neist kahest. See muudatus võimaldas välistada tööriistade tulemusi, mida võis pidada ebausaldusväärseks: näiteks kui tööriist andis väärnegatiivseid tulemusi ehk väitis, et andmejooksu ei leidunud, kui tegelikkuses oli andmejooksu teke antud programmis võimalik.

[5] https://github.com/sosy-lab/benchexec
[6] https://github.com/thufv/Deagle
[7] https://www.ultimate-pa.org/automizer
[8] https://www.ultimate-pa.org/gemcutter

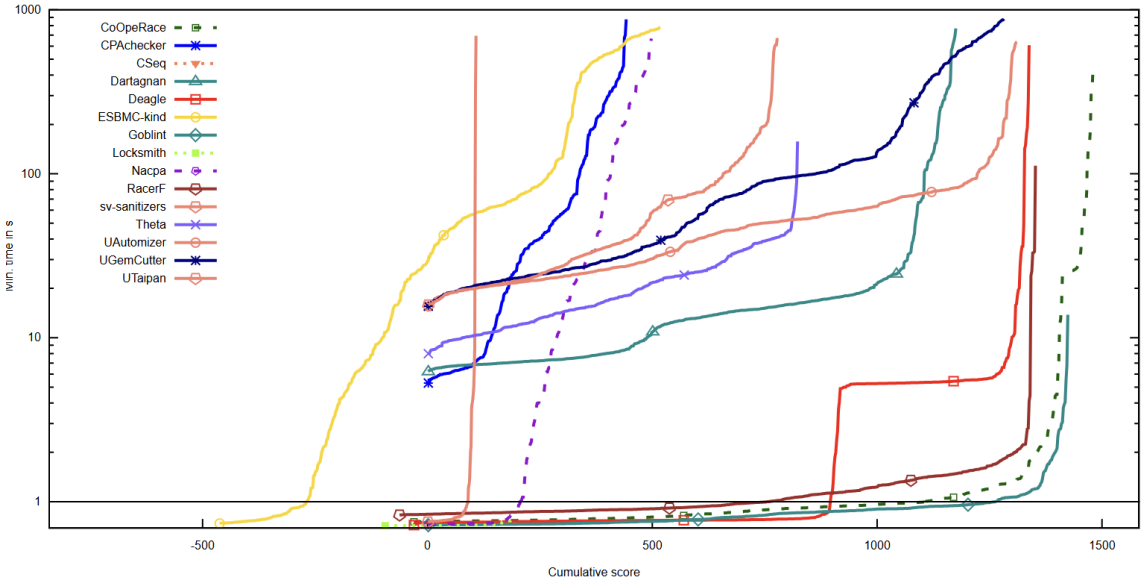
Tööriistade usaldusväärsuse ja analüüsitulemuste kontrollitavuse tagamisel mängib olulist rolli rikkumis- või korrektsustõendite valideerimine. Seega, kui alamtööriistad neid genereerisid, väljastas CoOpeRace CLI vastava tõendi, mida said eraldiseisvad valideerimistööriistad analüüsida. Selline mehhanism võimaldas kolmandatel osapooltel sõltumatult kontrollida, kas leitud rikkumine või väidetav korrektne käitumine oli formaalselt tõestatud, suurendades analüüsi läbipaistvust ja reprodutseeritavust.

Kolmandas, ühtlasi lõplikus versioonis täiendati loodud raamistikku nii, et toetatud tööriistade arv kasvas kümneni. Samuti muudeti koodibaasi üldisemaks, võimaldades raamistikku rakendada ka teiste nõrkuste tuvastamiseks peale andmejooksude, ilma lähtekoodi muutmata. Erinevalt teisest versioonist, mis osales võistlusel SV-COMP 2025 ning kus ilmnesid puudused alamtööriistade korrektsustõendite ja rikkumistõendite käsitlemisel, suutis kolmas versioon neid korrektselt töödelda.

CoOpeRace CLI töötab Linux-põhisel operatsioonisüsteemil, on käsurealt käivitatav ja aktsepteerib käivitamisel mitmeid parameetreid. Ainuke kohustuslik parameeter on analüüsitava faili asukoht, kuid lisaks saab täpsustada otsitavat nõrkust kirjeldava faili asukohta, valida analüüsis kasutatava verifitseerimistööriistade kombinatsioon ning määrata, kas analüüsitav fail oli loodud 32-bitise või 64-bitise arhitektuuri jaoks. CoOpeRace CLI lähtekood ja täpsem kasutusjuhend on toodud Lisas I.

3.2 SV-COMP 2025 tulemused

Joonisel 2 on kujutatud SV-COMP 2025 andmejooksu kategoorias osalenud tööriistad viisil, mis näitab, kui palju aega kulus igal tööriistal selle lõpptulemuse saavutamiseks. Y-teljel on verifitseerimisele kulunud aeg, ning x-teljel tööriista kumulatiivne tulemus. Legendil on toodud kategoorias osalenud tööriistad ja neile vastav märgendus. Joonis võimaldab visuaalselt analüüsida seost iga tööriista tulemuse ja tööaja vahel ning annab ühtlasi ülevaate ka tööriistade efektiivsusest, tuues esile, millised tööriistad suudavad saavutada kõrge verifitseerimistulemuse väiksema ajakuluga. Selle järgi saab hinnata tööriistade praktilist kasutatavust olukordades, kus arvutusressursid ja tööaeg on piiratud.



Joonis 2. Tööriistade tulemuse ja tööaja seos SV-COMP 2025[9] andmejooksu kategoorias

Näiteks saab jooniselt lugeda, et tööriistad Goblint ja RacerF[10] saavad esimesed ~1250 punkti kätte nii, et igale ülesandele kulub lahendamiseks aega vähem kui 1s, kuid näiteks Dartagnanil ja CPAcheckeril läheb iga ülesande lahendamiseks vähemalt 5 sekundit. Lisaks on näha, et pea kõikidel tööriistadel suureneb lahendusele kulunud aeg eksponentsiaalselt, mida keerukam on lahendatav ülesanne. Jooniselt saab ka vaadata, millised tööriistad olid täielikult korrektsed, st ei andnud valesid vastuseid, sest nende tööriistade punktide alguspunkt on x-teljel 0-st paremal pool, ning miinus punkte kogunud ehk valesid vastuseid andnud tööriistade graafikud algavad nullpunktist vasakult poolt.

CoOpeRace CLI kasutas SV-COMP 2025 võistlusel konfiguratsiooni, milles käivitati esmalt tööriista Goblint. Kui Goblint ei suutnud antud programmis andmejookse välistada, käivitati täiendavalt paralleelselt analüsaatorid: Dartagnan, ULTIMATE GemCutter, ULTIMATE Automizer ja Deagle. Selline konfiguratsioon valiti, sest SV-COMP 2024 tulemuste põhjal suutis Goblint suure osa ülesannete puhul kiiresti ja tõhusalt andmejooksu välistada. Goblint käivitati esimesena, et optimeerida analüüsi aega – kui see suutis kiiresti andmejooksu välistada, siis ei olnud vajadust ressursimahukamate tööriistade kaasamiseks. Kuna Goblint on tugev andmejooksude puudumise tõestamisel, kuid ei ole loodud potentsiaalsete andmejooksude tuvastamiseks, pidid täiendavad tööriistad selle puudujäägi täitma suurendamaks tuvastamise katvust. Tööriistad, mis Goblinti järel paralleelselt käivitati, valiti

[9] https://sv-comp.sosy-lab.org/2025/results/results-verified/
[10] https://doi.org/10.48550/arXiv.2502.04905

samuti SV-COMP 2024 tulemuste põhjal: need olid andmejooksude kategoorias parimate tulemustega analüsaatorid ning osutusid andmejooksude tuvastamisel usaldusväärseteks, st ei andnud ühtegi, või ainult üksikuid valesid tulemusi.

CoOpeRace CLI saavutas selle konfiguratsiooniga andmejooksu kategoorias esimese koha, kogudes 1482 punkti, mis edestas teise koha tulemust 58 punktiga. Kokku analüüsiti 1029 ülesannet, milleks kulus 16 tundi ja 24 minutit protsessoriaega ning 1100 GB mälu. See tähendab keskmiselt 57 sekundit protsessoriaega ja ligikaudu 1 GB mälu iga ülesande kohta.

4. Analüüs

Töö teiseks eesmärgiks oli leida parim verifitseerimistööriistade konfiguratsioon andmejooksude avastamiseks. SV-COMP 2025 osalenud CoOpeRace CLI konfiguratsioon põhines subjektiivsel individuaalsete tööriistade tugevuste ja nõrkuste eelteadmistel. Seetõttu oli mõistlik andmeanalüüsi abil uurida, kas on võimalik leida veelgi parem tööriistade kombinatsioon – ehk konfiguratsioon, millega saavutada kõrgem tulemus. See peatükk annab struktureeritud ülevaate parima konfiguratsiooni leidmise metoodikast, läbiviidud katsetest, ja saadud tulemustest, hõlmates järgmisi samme:

- SV-COMP teoreetiliste tulemuste analüsaatori tööpõhimõtted ja selle roll sobivate kombinatsioonide valikul nende reaalseks käivitamiseks;
- Teoreetiliste tulemuste analüüsi põhjal valitud tööriistade kombinatsioonide katsetustulemused piiratud muutmälu tingimustes, kus iga ülesande lahendamiseks oli CoOpeRace CLI kasutuses 4 gigabaiti (GB) muutmälu;
- Katsetulemused suurema muutmäluga (16 GB ülesande kohta) keskkonnas ning nende võrdlus piiratud mäluga katsetega;
- Koondanalüüs, milles tuuakse välja parimad konfiguratsioonid erinevate mõõdikute lõikes.

Katsed viidi läbi keskkonnas, mis jäljendab SV-COMP 2025 võistluse tingimusi: igale ülesandele oli eraldatud kuni 15 minutit protsessoriaega ning vastavalt 4 GB või 16 GB mälu. Piiratud mäluga katsed tehti eesmärgiga uurida millised tööriistade kombinatsioonid osutusid kõige tõhusamaks piiratud mäluga olukordades, mis võivad esineda praktilistes rakendustes.

4.1 SV-COMP tulemuste analüüs parimate kombinatsioonide leidmiseks

Kõikide võimalike tööriistade kombinatsioonide tegelik käivitamine oleks olnud arvutuslikult ressursimahukas ning praktikas teostamatu. Seetõttu osutus otstarbekaks kasutada teoreetilist hindamismeetodit, mille abil oli võimalik erinevate tööriistade kombinatsioonide potentsiaalset tulemuslikkust hinnata olemasolevate tulemuste põhjal, ilma neid kõiki reaalselt käivitamata. Nende kaalutluste põhjal teostati programm, mis arvutab teoreetilise tulemuse erinevatele võimalikele tööriistade kombinatsioonidele, toetudes SV-COMP 2025 andmestikule.

Teoreetilised tulemused kujutavad endast tööriistade võimaliku tulemuslikkuse hinnangut, mis arvutati olemasolevate andmete põhjal, ilma et tööriistu oleks reaalselt konkreetsetel sisenditel käivitatud. Selline lähenemine võimaldas suure hulga tööriistade kombinatsioonide potentsiaali efektiivselt hinnata ning vähendada arvutuskoormust. Seevastu saadi praktilised tulemused tööriistade reaalsel rakendamisel katseprogrammidele, mis kajastavad nende tegelikku toimivust konkreetsetes olukordades. Praktilised tulemused võimaldavad kinnitada teoreetiliste hinnangute paikapidavust ja on otseselt seotud tööriistade tegeliku rakendatavusega analüüsiprotsessis.

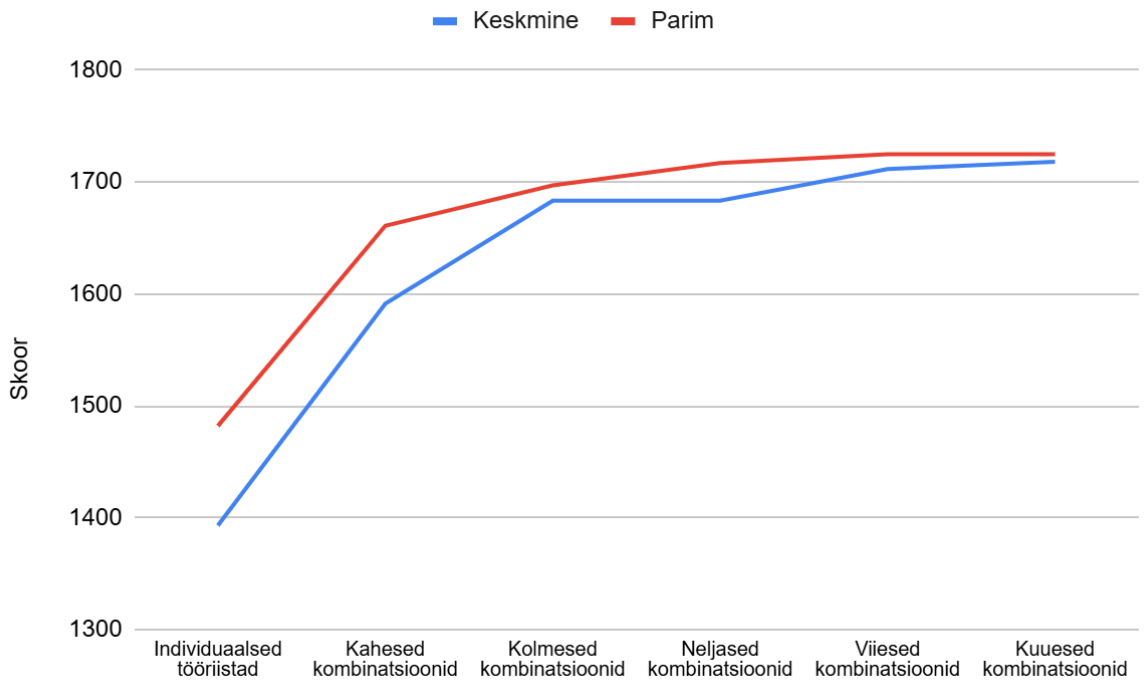
Tulemuste analüsaator töötles järjestikku kõikide tööriistade tulemused ja koondas need ühtsesse kogumikku. Seejärel moodustas analüsaator hulgad, kus ühes hulgas olid kõik sama arvu tööriistadega kombinatsioonid. Kui hulgad olid moodustatud, hakkas analüsaator hulgas olevaid kombinatsioone analüüsima, kombineerides kõik SV-COMP 2025 andmejooksu kategooria tulemused. Kombineeritud tulemuse koostamisel jälgis analüsaator järgmiseid reegleid:

1. Kui üks tööriist kombinatsioonist sai programmi jaoks vale tulemuse, siis oli kombinatsiooni tulemus selle programmi jaoks vale. Reegel premeeris kombinatsioone, kus oli minimaalne hulk vääri tulemusi, mis on turvaliste ja usaldusväärsete staatilise analüüsi tööriistade korral oluline omadus.
2. Kui kombinatsioonis oli vähemalt ühel tööriistal õige tulemus ning mitte ühelgi tööriistal ei olnud väära tulemust, oli kombinatsiooni tulemus õige.
3. Juhul kui kombinatsioonis ei leidunud ühelgi tööriistal ei õiget ega vale tulemust, siis selle programmi kohta ei suutnud kombinatsioon tulemust anda ning vastav programm kombinatsiooni üldist skoori ei mõjutanud.

Analüsaator väljastas oma tulemused CSV-failidesse, kus iga fail sisaldas kõiki sama tööriistade arvuga kombinatsioone ning nendele vastavaid teoreetilisi tulemusi. Saadud tulemused võimaldasid valida kombinatsioonid, mida SV-COMP ülesannete kogumikul praktiliselt katsetada. Analüsaator on võimeline andma nii algseid kui ka valideeritud tulemusi. Algsete tulemuste all on mõeldud tööriistade tulemusi enne valideerimisprotsessi ning valideeritud tulemuste all tulemusi, mille on kinnitanud valideerimistööriistad.

SV-COMP tulemuste analüsaator töötles kõikide tööriistade kombinatsioone, mis saavutasid SV-COMP 2025 andmejooksu kategoorias positiivse tulemuse. Analüüs tugines SV-COMP 2025 ametlikele tulemustele ning võimaldas igas kombinatsiooni suuruse kategoorias välja

selgitada viis teoreetiliselt parimat kombinatsiooni edasiseks analüüsiks. Joonis 3 illustreerib kombinatsiooni suuruse ja saavutatud teoreetilise skoori vahelist seost.



Joonis 3. Kombinatsioonide suuruse ja teoreetilise skoori vaheline seos

Tulemustest joonistub välja, et koos kombinatsiooni suurusega, liikudes kahestest kombinatsioonidest kuni kuuesteni, suureneb nii keskmine kui ka parim skoor. See viitab, et aina rohkemate tööriistade kombineerimine võib üldist võimekust andmejookse tuvastada parandada. Parima teoreetilise skooriga kombinatsioonid on viiene kombinatsioon tööriistadest sv-sanitizers, Deagle, Dartagnan, ULTIMATE GemCutter ja Goblint ning võrdse skooriga kuuene kombinatsioon, mis lisab olemasolevatele tööriistadele ka Nacpa[11]. Mõlemad kombinatsioonid saavutasid teoreetilise skoori 1725. Kõigi kombinatsioonide arvude viie parima kombinatsiooni teoreetilised skoorid on esitatud Lisas II.

4.2 Kombinatsioonide katsetamise tulemused piiratud mäluga

Kokku katsetati 25 kombinatsiooni 1029 ülesandest koosneva testhulga peal. Kombinatsioonides olevad tööriistad käivitati kõik koos paralleelselt, et võimaldada tööriistadel üksteisest sõltumatult võimalikult kiiresti tulemused esitada. Iga ülesande

[11] https://www.sosy-lab.org/research/pub/2025-TACAS.Nacpa_Native_Checking_with_Parallel-Portfolio_Analyses_Competition_Contribution.pdf

lahendamiseks oli kombinatsioonidel võimalik kasutada 15 minutit protsessoriaega ning 4 GB muutmälu. Tulemusi analüüsiti kombinatsioonide arvu kaupa ning seejuures uuriti järgmiseid parameetreid:

- Lahendatud ülesannete arv - Ülesannete arv, millele suutis kombinatsioon leida, kas TRUE või FALSE, tulemuse olenemata kas see oli õige või vale;
- Teoreetiline skoor - Kombinatsiooni saavutatud skoor tööriista “SV-COMP 2025 tulemuste analüsaator” abil;
- Praktiline skoor - Kombinatsiooni saavutatud skoor, kus õige TRUE andis kaks punkti, õige FALSE ühe punkti, vale TRUE võttis ära 32 punkti ning vale FALSE 16 punkti;
- Protsessori aeg - Kõikide lahendatud ülesannete peale kulunud protsessori aeg kombinatsioonil sekundi täpsusega;
- Kogu mälukasutus - Kõikide lahendatud ülesannete peale kulunud mälu kombinatsioonil megabaidi täpsusega.

Analüüsi käigus võeti vaatluse alla võimalikud 2-6 tööriista sisaldavad kombinatsioonid. Iga kombinatsioonide arvu puhul keskenduti viiele parimale kombinatsioonile, mis valiti SV-COMP 2025 tulemuste alusel arvutatud teoreetiliselt võimaliku skoori põhjal. See lähenemine võimaldas keskenduda kombinatsioonidele, mis olid teoreetilise analüüsi kohaselt parima potentsiaaliga ning muuta testimisprotsessi praktilisemaks ja tõhusamaks, säilitades samas vajalikud tingimused usaldusväärsete ja oluliste tulemuste saavutamiseks.

Katsete sooritamisel tehti kindlaks iga katsetatava kombinatsiooni tulemus ning arvutati mälukasutuse ja protsessorikasutuse kogusumma kogu testhulga kohta. Lisaks filtreeriti analüüsiks välja ainult need programmid, mille puhul saadi kindel vastus TRUE või FALSE, olenemata selle õigsusest. Kõik juhud, kus tööriist ei andnud selget tulemust (nt UNKNOWN, ERROR jt), jäeti analüüsist kõrvale. Filtreerimise eesmärk oli mitmetahuline: esiteks võimaldas see keskenduda ainult nendele juhtumitele, kus tööriistade kombinatsioon andis lõpliku ja interpreteeritava vastuse, mis omakorda võimaldas võrrelda kombinatsioonide efektiivsust usaldusväärsetel alustel. Teiseks välditi ebaselgete tulemuste kaasamisel tekkivat potentsiaalset kallutatust, mis võib tuleneda tööriistade sisemistest piirangutest või keskkondlikest teguritest. Kolmandaks hoidis see analüüsi fookuse kombinatsioonide praktilisel võimekusel – st kui palju teste suudeti tegelikult lahendada.

Keskmiselt suutsid viis parimat verifitseerimistööriista individuaalselt lahendada 769 ülesannet 1029 ülesandest ehk ligikaudu 75%. Nende keskmine praktiline skoor oli 1401 punkti, mille saavutamiseks kulus 8 tundi ja 13 minutit ning 590 GB mälu. Viis parimat kahest tööriistast koosnevat kombinatsiooni suutsid keskmiselt lahendada 850 ülesannet, mis on 83% kõigist ülesannetest. Nende keskmine praktiline skoor oli 1500 punkti, mille saavutamiseks kulus ligikaudu 3 tundi ja 12 minutit, kasutades 186 GB mälu.

Võrreldes individuaalsete tööriistadega, suutsid kahesed kombinatsioonid keskmiselt:

- Lahendada 81 ülesannet rohkem;
- Saavutada 99 punkti võrra kõrgema skoori;
- Töötada 4 tundi ja 58 minutit kiiremini;
- Kasutada 404 GB vähem mälu.

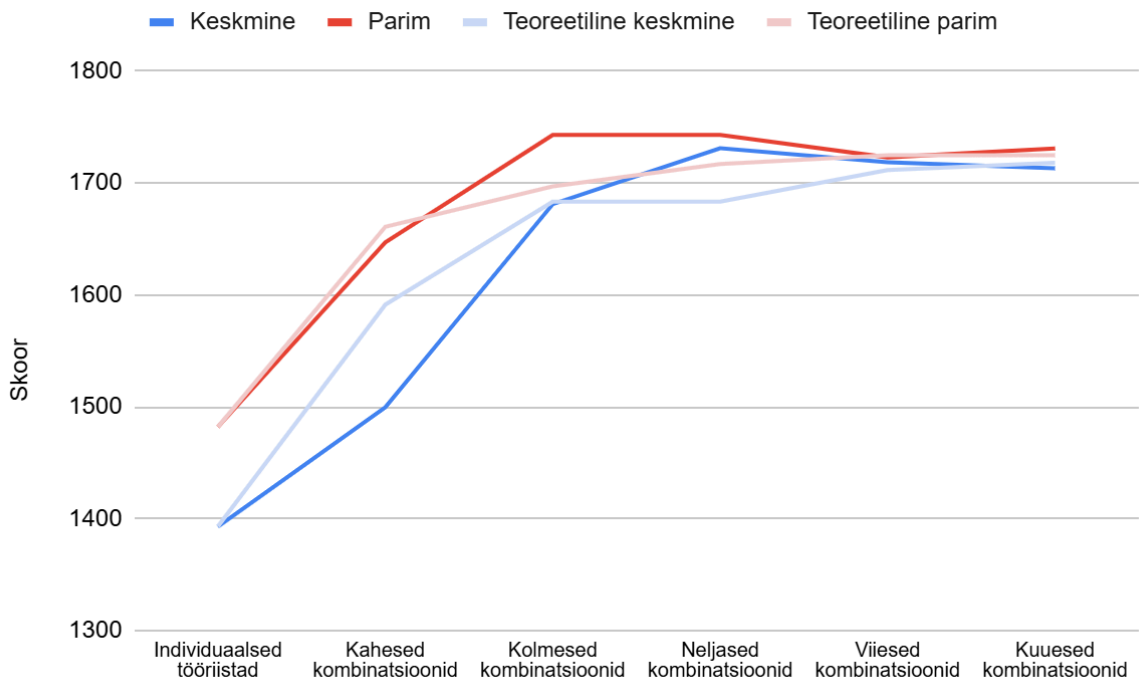
Tabelis 1 on esitatud viie kahese kombinatsiooni tööriistade nimed, tööriistade individuaalsed skoorid ning nende kombinatsiooni skoor. Kõikide piiratud mäluga katsete tulemused on toodud Lisas III.

Tabel 1. Kaheste kombinatsioonide tööriistad, nende individuaalsed skoorid ning kombinatsioonide skoorid

1. tööriista nimi ja skoor	**2. tööriista nimi ja skoor**	**Kombinatsiooni skoor**
Dartagnan - 1175	Goblint - 1424	1647
UGemCutter - 1317	Goblint - 1424	1598
Dartagnan - 1175	Deagle - 1541	1569
UAutomizer - 1339	Goblint - 1424	1555
Dartagnan - 1175	RacerF - 1382	1129

Neljal viiest kombinatsioonist ületas kombinatsiooni skoor kummagi üksiku tööriista skoori, mis näitab, et tööriistade kombineerimine andis paremaid tulemusi kui üksikute tööriistade kasutamine. Erandiks oli Dartagnani ja RacerFi kombinatsioon, mis oli ainus kahest tööriistast koosnev katsetatud kombinatsioon, mis sai halvema skoori kui kumbki tööriist eraldi. See halvem tulemus oli peamiselt tingitud ajapuudusest, mille tõttu jäi lahendamata 201 ülesannet.

Keskmiselt andsid parimaid tulemusi aga neljast tööriistast koosnevad kombinatsioonid. Need suutsid õigesti lahendada keskmiselt 974 ülesannet ehk 95% kõikidest ülesannetest, ning saavutasid keskmiselt skoori 1731 punkti. Üks viiest neljasest kombinatsioonist andis ühe vale tulemuse, samas kui ülejäänud nelja kombinatsiooni puhul valesid tulemusi ei esinenud. Ülejäänud juhtudel ei suutnud konfiguratsioon anda konkreetset hinnangut, mis tähendab, et tööriistade kombinatsioon ei jõudnud nende ülesannete puhul määratud aja või ressursside piires järelduseni. Ülesannete lahendamiseks kulus keskmiselt 2 tundi ja 13 minutit protsessoriaega ning 218 GB mälu. Joonisel 4 on kujutatud viie individuaalselt parima tööriista keskmine ja parim skoor ning erineva suurusega tööriistakombinatsioonide keskmised ja parimad tulemused. Graafikult on näha, et tulemused paranevad märgatavalt kuni neljaste kombinatsioonideni, kuid seejärel ei toimu skooris märgatavat kasvu.



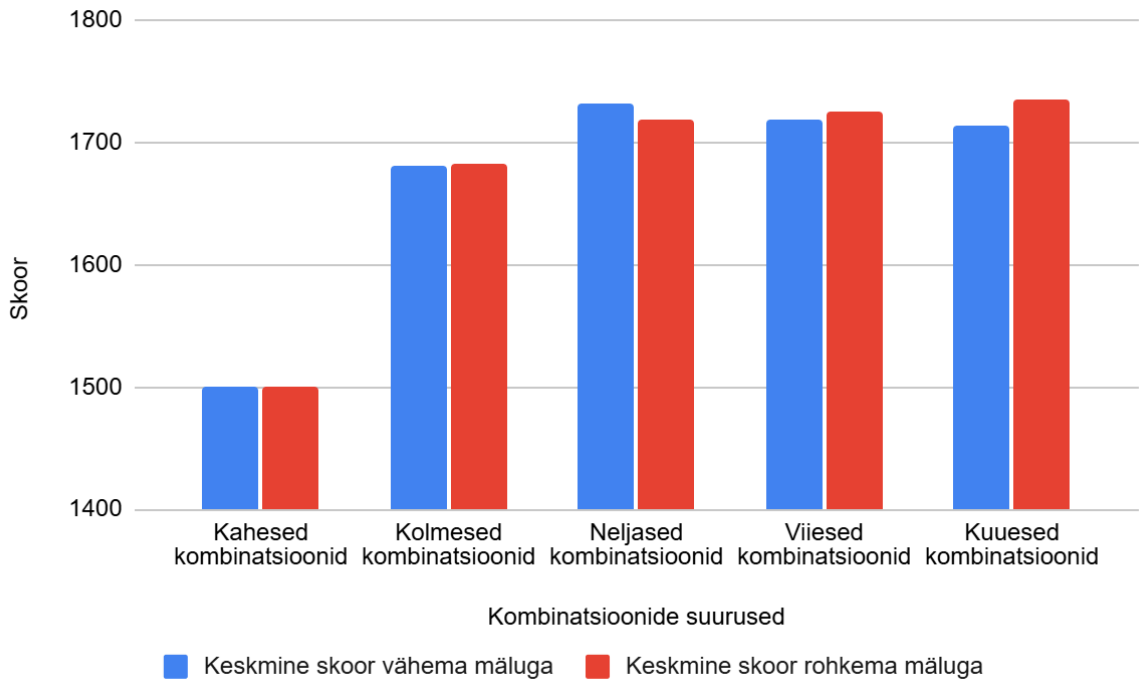
Joonis 4. Kombinatsioonide suuruse ja teoreetilise skoori vaheline seos piiratud mäluga katsetingimustes

Teoreetiliste ja praktiliste skooride vaheline erinevus oli keskmiselt 16 punkti. See näitab, et SV-COMP tulemuste analüsaator töötas tulemuste ennustamisel edukalt, kuna teoreetilised skoorid andsid usaldusväärse ülevaate kombineeritud tööriistade tegelikust tulemuslikkusest. Vähene erinevus teoreetiliste ja praktiliste skooride vahel kinnitab, et analüsaator suutis valida efektiivsed kombinatsioonid, mis tegelikult ka katsetes hästi esinesid.

Praktiliste katsete käigus saavutasid parima tulemuse kaks kombinatsiooni: kolmest tööriistast (Goblint, Deagle, Dartagnan) ning neljast tööriistast koosnevad kombinatsioonid (Goblint, Deagle, Dartagnan, Nacpa). Mõlemad kombinatsioonid suutsid lahendada 980 ülesannet ning said skooriks 1743 punkti. Erinevus seisnes protsessoriajas ning mälukasutuses, kus kolme tööriista kombinatsioon kasutas 2 tundi ja 41 minutit protsessoriaega ning 184 gigabaiti mälu, kuid nelja tööriista kombinatsioon kasutas 2 tundi ja 35 minutit protsessoriaega ning 262 GB mälu. Võrreldes nelja tööriista kombinatsiooniga, kasutas kolme tööriistaga kombinatsioon ülesannete lahendamiseks peaaegu kolmandiku võrra vähem mälu, kuid rohkem aega, viimast küll vaid 6 minutit ehk 0.04 korda.

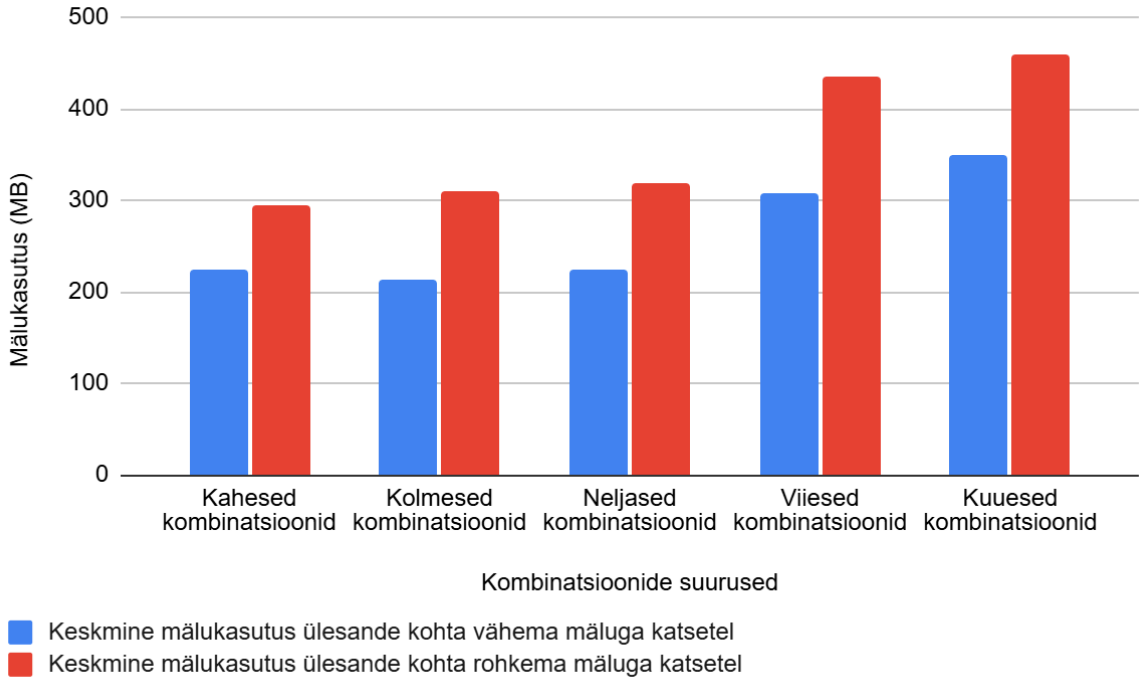
4.3 Kombinatsioonide katsetamise tulemused suurema mäluga

Lisaks eelmises alampeatükis kirjeldatud katsetele, viidi samade kombinatsioonide ja testhulgaga läbi katsed suurema mälumahuga, kus 4 GB asemel oli igal kombinatsioonil võimalik maksimaalselt kasutada 16 GB muutmälu. Suurema muutmäluga katsete tulemused on toodud Lisas IV. Joonisel 5 on toodud iga kombinatsiooni suuruse keskmised tulemused nii väiksema kui ka suurema mäluhulga korral.



Joonis 5. Kombinatsiooni suuruste keskmiste skooride võrdlus suuremal (16 GB) ning väiksemal (4 GB) mäluhulgal.

Jooniselt on näha, et väiksema ja suurema mälumahu kasutamisel ei esine tulemustes märgatavat erinevust. Erinevalt väiksema mälumahuga katsetest ei täheldatud neljaste kombinatsioonide puhul enam skoori haripunkti ning keskmised skoorid jätkasid stabiilset kasvu. Erinevus tekkis keskmises mälukasutuses, mida on näha joonisel 6, kus on välja toodud keskmine mälukasutus ülesande kohta igas kombinatsiooni suuruses.



Joonis 6. Keskmine mälukasutus ülesande kohta nii vähema kui ka suurema mäluga katsetel.

Rohkema mälumahuga katsetes oli keskmine mälukasutus kõigi kombinatsiooni suuruste puhul suurem kui vähema mälumahuga katsetes. Mõlemas katsestenaariumis on alates viiestest kombinatsioonidest näha märgatavat hüpet mälukasutuses, mis viitab kombinatsioonide keerukuse kasvule ja sellest tulenevale suuremale ressursivajadusele. Kuigi keskmine mälukasutus suurema mäluhulga korral suureneb, siis skoorid muutuvad ainult marginaalselt.

1749 punktise skooriga saavutasid parima tulemuse suurema mäluhulgaga katsetes kaks kombinatsiooni: viiest tööriistast kombinatsioon (sv-sanitizers, Deagle, Dartagnan, ULTIMATE GemCutter, Goblint), mis lahendas õigesti 989 ülesannet 1029 ülesandet, kasutades nende 989 ülesande lahendamiseks 4 tundi ja 10 minutit protsessoriaega ning 369 GB muutmälu. Teine parim kombinatsioon sisaldas kuute tööriista (Nacpa, ULTIMATE Taipan, CPAchecker, Deagle, Dartagnan, Goblint) ning lahendas õigesti 988 ülesannet, kuid

kasutab selleks rohkem protsessoriaega (5 tundi ja 17 minutit) ning muutmälu (461 GB) kui varasemalt kirjeldatud viiene kombinatsioon.

4.4 Tulemused

Parima konfiguratsiooni leidmiseks oli vaja sooritada katsed erinevate kombinatsioonidega ning analüüsida saadud tulemusi. Ainuüksi saavutatud punktiskoor SV-COMP formaadis ei olnud piisav kriteerium konfiguratsiooni kvaliteedi hindamiseks; arvesse tuli võtta ka täiendavaid mõõdikuid nagu mälu kasutus ja protsessoriaeg, mis kulusid vastava tulemuse saavutamiseks. Selline mitmemõõtmeline lähenemine võimaldas hinnata konfiguratsioonide praktilist efektiivsust ning sobivust erinevates kasutusstsenaariumides. Selle asemel, et välja tuua üks universaalselt parim lahendus, oli lõputöös välja toodud parim konfiguratsioon iga nimetatud faktori põhjal eraldi. Kõikide katsete tulemuste asukoht on toodud Lisas I.

Protsessoriaja põhjal osutus kõige tõhusamaks kombinatsiooniks tööriistade Deagle, Dartagnan ja Goblint ühiselt kasutamine, mis lahendas korrektselt 988 ülesannet ehk ligikaudu 96% koguhulgast. Katsetused viidi läbi suurema muutmäluga keskkonnas ning nimetatud konfiguratsioon kulutas ühe ülesande lahendamiseks keskmiselt 7 sekundit. Kokku kulus keskmiselt ühe ülesande lahendamiseks 7 sekundit 7 kombinatsioonil, kuid teised kombinatsioonid lahendasid vähem ülesandeid õigesti. See teeb antud kombinatsioonist sobiva valiku olukordades, kus määravaks teguriks on lahenduskiirus.

Muutmälu kasutuse suhtes osutus kõige efektiivsemaks kombinatsiooniks Dartagnan ja Goblint, mille puhul kulus ühe ülesande lahendamiseks keskmiselt 162 megabaiti muutmälu. Sarnaselt eelnevalt kirjeldatud konfiguratsioonile saavutati ka see tulemus piiratud muutmäluga katsetingimustes. Kokku lahendati õigesti 929 ülesannet ning valesti üks ülesanne. Antud konfiguratsioon on sobiv kasutamiseks keskkondades, kus süsteemi mälumaht on piiratud või oluline on madal ressursikasutus.

SV-COMP skoori alusel oli parimaks konfiguratsiooniks kombinatsioon tööriistadest Nacpa, sv-sanitizers, Deagle, Dartagnan, ULTIMATE GemCutter ja Goblint, mis saavutas suurema muutmäluga katsetingimustes 1748 punkti, lahendades õigesti 987 ülesannet, kusjuures ühtegi ülesannet ei lahendatud valesti. Kõrge punktisumma viitab sellele, et antud konfiguratsioon suudab edukalt tasakaalustada avastamise ja välistamise võimekust, minimeerides samal ajal eksimuste arvu.

Kõige rohkem ülesandeid suutis õigesti lahendada kombinatsioon tööriistadest sv-sanitizers, Deagle Dartagnan ja Goblint. Suurema muutmäluga katsetingimustes suutis see konfiguratsioon lahendada 995 ülesannet 1029-st, millest 994 ehk ligikaudu 97% olid õigesti analüüsitud ning üks ülesanne valesti. Viimase puhul tuvastas kombinatsioon ekslikult, et programm ei sisalda andmejookse, kuigi tegelikkuses oli ülesandes andmejooks olemas. Selle konfiguratsiooni koguskoor SV-COMP hindamismudeli alusel oli 1728 punkti, mis jäi madalamaks mitme teise konfiguratsiooni tulemustest. Saadud tulemus viitab sellele, et konfiguratsioon suutis küll suurel määral välistada võimalikke andmejookse, kuid ei suutnud neid sama tõhusalt avastada. Sellest hoolimata võib andmejooksude välistamise võimekus olla mitmetes rakendustes olulisem kui nende avastamine, eriti juhul, kui süsteemi töökindlus ja katkestuste vältimine on suurem prioriteet kui väikeste vigade avastamine.

Välja toodud tulemused baseeruvad 2025. aasta SV-COMP võistluse tulemustel, keskendudes erinevate tööriistade ja nende kombinatsioonide tõhususele määratud katsetingimustes. Kasutatud analüüsimeetodid on aga rakendatavad ka teiste aastate SV-COMP tulemustele. Näiteks saab sama analüütilist lähenemist kasutada tulevaste võistluste, sealhulgas 2026. aasta tulemuste hindamisel, et jälgida muutusi tööriistade või nende kombinatsioonide kvaliteedis. See võimaldab tuvastada, millised lahendused on aja jooksul paranenud, stabiilsena püsinud või oma tõhususes langenud, pakkudes väärtuslikku sisendit nii tööriistade arendajatele, kasutajatele kui ka valdkonna uurijatele.

CoOpeRace CLI suudab jooksutada mitme tööriista kombinatsioone ning anda tulemuse esimeselt lõpetanud tööriistalt ehk kasutada kiirusepõhist otsustusstrateegiat. Edasises arenduses oleks võimalik täiustada metaverifitseerijat viisil, mis võimaldab tulemuste aksepteerimist täpselt määratletud usaldustingimuste alusel. Näiteks võiks metaverifitseerija aktsepteerida tulemust vaid juhul, kui vähemalt kaks määratud tööriista jõuavad samale järeldusele. Alternatiivselt võiks tulemust usaldada juhul, kui tööriist suudab lisada oma väljundisse kas korrektsustõendi või rikkumistõendi, mis võimaldaks hilisemat automaatset kontrolli või valideerimist. Taolised mehhanismid suurendaksid tulemuste usaldusväärsust, võimaldades rakendada metaverifitseerijat ka kriitiliste tarkvarasüsteemide analüüsi kontekstides.

Kokkuvõte

Käesoleval bakalaureusetööl oli kaks peamist eesmärki: esiteks luua tööriist, mis rakendab verifitseerimiskoostöö põhimõtteid ning teiseks, leida parim kombinatsioon olemasolevatest verifitseerimistööriistadest andmejooksude avastamiseks.

Töö tulemusena valmis tööriist CoOpeRace CLI, mis on verifitseerimiskoostöö põhimõtteid järgiva arhitektuuriga käsurealt kasutatav raamistik, mis võimaldab erinevate staatilise analüüsi tööriistade kombineeritud kasutamist, et maksimeerida nende individuaalsete tugevuste rakendamist programmides nõrkuste tuvastamisel. CoOpeRace CLI on loodud modulaarse disainiga, mis võimaldab kasutajatel lisada uusi tööriistu või muuta olemasolevaid kombinatsioone konfiguratsioonifailide kaudu, ilma lähtekoodi muutmata. Selline lähenemine tagab paindlikkuse ning hõlbustab tööriista kohandamist erinevatele kasutusstsenaariumitele ja nõrkusetüüpidele.

Katsete käigus analüüsiti kokku 25 erinevat kombinatsiooni verifitseerimistööriistadest millest igas suuruses (kaheseid, kolmeseid, neljaseid, viieseid ja kuueseid) kombinatsioone oli viis. Katsetused viidi läbi 1029 ülesandest koosneva testhulgaga, kus kõiki kombinatsioone testiti nii piiratud muutmälu tingimustes kui ka olukordades, kus kombinatsioonide käsutuses oli suurem muutmälumaht. Iga kombinatsiooni puhul mõõdeti skoori, protsessoriaega, mälukasutust ning õigesti ja valesti lahendatud ülesannete arvu, eesmärgiga tuvastada iga parameetri osas kõige tõhusam lahendus.

Katsetulemuste põhjal tuvastati, et protsessoriaja kui ka muutmälu suhtes oli kõige tõhusam tööriistade kombinatsioon Deagle, Dartagnan ja Goblint. Kõrgeima tulemusskoori saavutas tööriistade kooslus Nacpa, sv-sanitizers, Deagle, Dartagnan, ULTIMATE GemCutter ja Goblint. Kõige suurema arvu õigesti lahendatud ülesandeid suutis täita kombinatsioon, mis koosnes tööriistadest sv-sanitizers, Deagle, Dartagnan ja Goblint.

Viidatud kirjandus

[1] Chess B., McGraw G. Static analysis for security. *IEEE Security & Privacy*, 2004, vol 2, nr 6, lk 76-79. DOI:10.1109/MSP.2004.111

[2] Beyer D., Wehrheim H. Verification Artifacts in Cooperative Verification: Survey and Unifying Component Framework. *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*. Cham: Springer International Publishing, 2020, lk 143-167. DOI:10.1007/978-3-030-61362-4_8

[3] Vojandi V. Static data race analysis of heap-manipulating C programs. TÜ arvutiteaduse instituudi doktoritöö. 2010. https://dspace.ut.ee/items/9db091d9-0895-471b-bf18-8a5685ca6c34. (11.01.2025)

[4] D'Silva V., Kroening D., Weissenbacher G. A Survey of Automated Techniques for Formal Software Verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2008, vol 27, nr 7, lk 1165-1178. DOI:10.1109/TCAD.2008.923410

[5] Beyer D., Strejček J. Case Study on Verification-Witness Validators: Where We Are and Where We Go. *Static Analysis*. Auckland, New Zealand: Springer Nature Switzerland, 2022. p 160-174.

[6] AKIT: Andmekaitse ja Infoturbe Portaal. https://akit.cyber.ee.

[7] Beyer. D, Friedberger K. Violation Witnesses and Result Validation for Multi-Threaded Programs. https://www.sosy-lab.org/research/witnesses-concurrency/ (28.04.2025)

[8] Nachtigall M., Nguyen Quang Do L., Bodden E. Explaining Static Analysis - A Perspective. *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, 2019, lk 29-32. DOI:10.1109/ASEW.2019.00023

[9] Fowler M., Parsons R. Domain-Specific Languages. Boston: Addison-Wesley. 2010.

[10] Haltermann J., Wehrheim H. CoVEGI: Cooperative Verification via Externally Generated Invariants. *Fundamental Approaches to Software Engineering*. Cham: Springer International Publishing, 2021, lk 108-129. DOI:10.1007/978-3-030-71500-7_6

[11] Keul S. Tuning Static Data Race Analysis for Automotive Control Software. *Proceedings - 11th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2011*, 2011, lk 45-54. DOI:10.1109/SCAM.2011.16

[12] Bora U., Vaishay S., Joshi S., Upadrasta R. OpenMP aware MHP Analysis for Improved Static Data-Race Detection. *2021 IEEE/ACM 7th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC)*, 2021, lk 1-11. DOI:10.1109/LLVMHPC54804.2021.00006

[13] Yoo Y.-C.., Gangopadhyay A. What Are Data Races and How to Avoid Them During Software Development. *MathWorks*, 2020. https://www.mathworks.com/products/polyspace/static-analysis-notes/what-data-races-how-avoid-during-software-development.html (02.01.2025)

[14] 14th Competition on Software Verification (SV-COMP 2025). https://sv-comp.sosy-lab.org/2025/ (11.01.2025)

[15] Beyer D. State of the Art in Software Verification and Witness Validation: SV-COMP 2024. *Tools and Algorithms for the Construction and Analysis of Systems*, Cham: Springer International Publishing, 2024, lk 299-329. DOI:10.1007/978-3-031-57256-2_15

[16] Beyer D., Kanav S. CoVeriTeam: On-Demand Composition of Cooperative Verification Systems. *Tools and Algorithms for the Construction and Analysis of Systems*. Cham: Springer International Publishing, 2022, lk 561–579. DOI:10.1007/978-3-030-99524-9_31

[17] Haavasalu E. Töövahendi CoOpeRace loomine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2024. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=79993 (22.03.2025)

Lisad

I Valminud tööriistad ja analüüsiks kogutud toorandmed

Tööriistade CoOpeRace CLI, SV-COMP tulemuste analüsaatori ning muude töö käigus loodud töövahendite lähtekood on leitav aadressil https://github.com/sws-lab/cooperace. Töö käigus läbi viidud katsete tulemused on kättesaadavad aadressil https://github.com/sws-lab/cooperace/blob/main/results.zip.

II Verifitseerimistööriistade kombinatsioonide teoreetilised skoorid

Tabel 2. Iga suuruse viis kõrgeima teoreetilise skooriga kombinatsiooni SV-COMP 2025 tulemuste analüüsist

Kombinatsiooni nimi	**Teoreetiline skoor**
dartagnan_goblint	1661
ugemcutter_goblint	1605
deagle_dartagnan	1555
uautomizer_goblint	1565
racerf_dartagnan	1571
deagle_dartagnan_goblint	1697
utaipan_dartagnan_goblint	1677
uautomizer_dartagnan_goblint	1678
dartagnan_ugemcutter_goblint	1683
sv-sanitizers_dartagnan_goblint	1682
utaipan_deagle_dartagnan_goblint	1711
sv-sanitizers_deagle_dartagnan_goblint	1717
nacpa_deagle_dartagnan_goblint	1701
uautomizer_deagle_dartagnan_goblint	1712
deagle_dartagnan_ugemcutter_goblint	1717
nacpa_utaipan_deagle_dartagnan_goblint	1711
utaipan_sv-sanitizers_deagle_dartagnan_goblint	1721
uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1722
sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1725
nacpa_uautomizer_deagle_dartagnan_goblint	1712
nacpa_utaipan_sv-sanitizers_deagle_dartagnan_goblint	1721
nacpa_uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1722
nacpa_sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1725
nacpa_utaipan_uautomizer_deagle_dartagnan_goblint	1712
nacpa_utaipan_cpachecker_deagle_dartagnan_goblint	1711

III Kombinatsioonide katsetamise tulemused piiratud mäluga

Tabel 3. Kombinatsioonide tulemused 4 GB muutmäluga keskkonnas

Kombinatsioon	Skoor	Õigesti lahendatud ülesannete arv	Valesti lahendatud ülesannete arv	Mälu-puudusega lõppenud ülesannete arv	Keskmine protsessori aeg ülesande kohta (s)	Keskmine mälukasutus ülesande kohta (MB)
dartagnan_goblint	1647	929	1	8	12	162
ugemcutter_goblint	1598	876	0	78	10	186
deagle_dartagnan	1569	910	1	23	14	236
uautomizer_goblint	1555	832	0	167	7	168
racerf_dartagnan	1129	699	4	8	25	368
deagle_dartagnan_goblint	1743	980	0	23	10	188
utaipan_dartagnan_goblint	1642	911	0	100	7	225
uautomizer_dartagnan_goblint	1664	931	0	78	9	247
dartagnan_ugemcutter_goblint	1675	941	0	56	10	237
sv-sanitizers_dartagnan_goblint	1682	949	0	13	7	172
utaipan_deagle_dartagnan_goblint	1719	963	0	60	7	207
sv-sanitizers_deagle_dartagnan_goblint	1734	988	1	18	8	201
nacpa_deagle_dartagnan_goblint	1743	980	0	23	9	267
uautomizer_deagle_dartagnan_goblint	1726	967	0	57	7	221
deagle_dartagnan_ugemcutter_goblint	1734	972	0	47	9	221
nacpa_utaipan_deagle_dartagnan_goblint	1718	963	0	60	7	315
utaipan_sv-sanitizers_deagle_dartagnan_goblint	1715	968	0	55	10	280
uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1722	972	0	52	10	307
sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1715	977	1	40	14	298
nacpa_uautomizer_deagle_dartagnan_goblint	1723	966	0	58	8	334
nacpa_utaipan_sv-sanitizers_deagle_dartagnan_goblint	1716	968	0	55	9	338
nacpa_uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1723	971	0	53	10	355

nacpa_sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1731	976	0	42	12	340
nacpa_utaipan_uautomizer_deagle_dartagnan_goblint	1666	952	1	72	12	375
nacpa_utaipan_cpachecker_deagle_dartagnan_goblint	1730	976	0	42	16	341

IV Kombinatsioonide katsetamine tulemused suurema mäluhulga korral

Tabel 4. Kombinatsioonide tulemused 16 GB muutmäluga keskkonnas

Kombinatsioon	Skoor	Õigesti lahendatud ülesannete arv	Valesti lahendatud ülesannete arv	Mälu-puudusega lõppenud ülesannete arv	Keskmine protsessori aeg ülesande kohta (s)	Keskmine mälukasutus ülesande kohta (MB)
dartagnan_goblint	1647	929	1	0	13	171
ugemcutter_goblint	1608	885	0	18	14	270
deagle_dartagnan	1563	916	2	5	17	290
uautomizer_goblint	1571	848	0	95	14	346
racerf_dartagnan	1115	700	5	0	25	401
deagle_dartagnan_goblint	1723	988	1	5	7	251
utaipan_dartagnan_goblint	1669	935	0	74	13	446
uautomizer_dartagnan_goblint	1675	941	0	49	13	340
dartagnan_ugemcutter_goblint	1681	947	0	20	13	320
sv-sanitizers_dartagnan_goblint	1667	950	1	3	8	189
utaipan_deagle_dartagnan_goblint	1710	979	1	41	9	333
sv-sanitizers_deagle_dartagnan_goblint	1728	994	1	7	9	251
nacpa_deagle_dartagnan_goblint	1721	987	1	5	12	329
uautomizer_deagle_dartagnan_goblint	1716	983	1	30	11	369
deagle_dartagnan_ugemcutter_goblint	1720	986	1	20	13	317
nacpa_utaipan_deagle_dartagnan_goblint	1709	979	1	41	10	454
utaipan_sv-sanitizers_deagle_dartagnan_goblint	1738	983	0	39	13	420
uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1710	985	1	26	15	412
sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1749	989	0	20	15	373
nacpa_uautomizer_deagle_dartagnan_goblint	1717	985	1	30	14	522
nacpa_utaipan_sv-sanitizers_deagle_dartagnan_goblint	1737	981	0	39	11	449
nacpa_uautomizer_sv-sanitizers_deagle_dartagnan_goblint	1745	986	0	27	14	455

nacpa_sv-sanitizers_deagle_dartagnan_ugemcutter_goblint	1748	987	0	19	16	480
nacpa_utaipan_uautomizer_deagle_dartagnan_goblint	1693	969	1	53	14	442
nacpa_utaipan_cpachecker_deagle_dartagnan_goblint	1749	988	0	20	19	467

V Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Chris Matsiselts,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Verifitseerimistööriistade kombineerimine andmejooksude tõhusamaks avastamiseks, mille juhendajad on Vesal Vojdani ja Karoliine Holter, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Chris Matsiselts

15.05.2025