Kauno technologijos universitetas

Informatikos fakultetas

Automatinių žaidimo Minesweeper sprendimo metodų tyrimas

Baigiamasis magistro studijų projektas

Gediminas Masaitis

Projekto autorius

Doc. dr. Mantas Lukoševičius

Vadovas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Automatinių žaidimo Minesweeper sprendimo metodų tyrimas

Baigiamasis magistro studijų projektas

Programų sistemų inžinerija (6211BX011)

Gediminas Masaitis

Projekto autorius

Doc. dr. Mantas Lukoševičius

Vadovas

Prof. dr. Tomas Blažauskas

Recenzentas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Gediminas Masaitis

Automatinių žaidimo Minesweeper sprendimo metodų tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs;

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Gediminas Masaitis

Patvirtinta elektroniniu būdu

Gediminas Masaitis. Automatinių žaidimo Minesweeper sprendimo metodų tyrimas. Magistro baigiamasis projektas / vadovas doc. dr. Mantas Lukoševičius; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis: Programų sistemų inžinerija.

Reikšminiai žodžiai: žaidimai, Minesweeper, automatinis sprendimas.

Kaunas, 2021.

Santrauka

Minesweeper – tai vienas žinomiausių žaidimų pasaulyje, pateikiamas kartu su *Microsoft Windows* operacinėmis sistemomis. Tai yra loginis žaidimas, kurį spręsti geba ne kiekvienas. Norint padaryti šį žaidimą prieinamesnį, gali būti pasitelkiamas automatinis žaidimo sprendimas, padedantis naujokams apeiti galimas keblias žaidimo situacijas, kylančias žaidžiant žaidimą įprastai.

Norint užtikrinti vartotojo pasitenkinimą, automatinis žaidimo sprendimas turi būti ir tikslus ir greitas. Šiame tyrime ištirti keli egzistuojantys sprendimo algoritmai, sukurti tikslumo bei greitaveikos patobulinimai jiems, nustatyta optimali algoritmų kombinacija teikianti aukštą tikslumą bei sparčią greitaveiką, bei sukurtas Minesweeper žaidimo prototipas su integruotu automatinio sprendimo funkcionalumu.

Gediminas Masaitis Research of automatic Minesweeper solving methods. Final Degree Project of Master's Studies / supervisor assoc. prof. Mantas Lukoševičius; The faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Program systems engineering.

Keywords: games, Minesweeper, automatic solving.

Kaunas, 2021.

Summary

Minesweeper is one of the most famous games in the world, installed together with *Microsoft Windows* operating systems. It is a logic game, not easily understandable by everyone. To make the game more accessible, it is possible to use automatic solving, allowing beginners to circumvent difficult situations that may arise when playing the game unassisted.

To ensure user satisfaction, automatic solving should be both accurate and fast. In this research, multiple Minesweeper solving algorithms have been tested, improvements for both accuracy and performance have been created, additionally a prototype for both playing and automatically solving Minesweeper has been created.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

P (angl. *polynomial*) – Uždavinių, kategoriją, kuri gali būti išsprendžiama polinominiu laiku

NP (angl. *nondeterministic polynomial*) – Uždavinių kategorija, kuri negali gali būti išsprendžiama polinominiu laiku, bet sprendimas gali būti patikrinamas polinominiu laiku.

NP-sunkus (angl. *NP-hard*) – Uždaviniai, kurie yra tokie pat sunkūs arba sunkesni, nei sunkiausias įmanomas NP uždavinys

NP-pilnas (angl. *NP-complete*) – Uždaviniai, kurie tuo pačiu metu yra NP bei NP-sunkūs

SAT (angl. *Boolean satisfiability problem*) – Uždavinys, klausiantis ar esant tam tikrai Bulio logikos schemai įmanoma konkreti išeitis. Dažnai naudojamas kaip NP-pilnumo etalonas.

CSP (angl. *Constraint satisfaction problem*) – Uždavinių kategorija, ar jų sprendimo būdas, kuris pasižymi apribojimų (angl. *constraints*) iškėlimu, ir jų patikrinimu pagal esamą sistemos būseną.

RL (angl. *Reinforcement learning*) – Viena iš trijų pagrindinių mašininio mokymosi šakų, kurioje agentai veikia jiems pateikiamoje aplinkoje, ir yra apdovanojami arba baudžiami už savo veiksmus.

ILP (angl. *Inductive logic programming*) – Dirbtinio intelekto sritis, kurioje generuojamos taisyklės, ir atmetamos pagal turimą faktų duomenų bazę.

PĮ – Programinė įranga.

CSV (angl. Comma separated values) – Paprastas duomenų saugojimo formatas naudojamas lentele, kur kiekvienas įrašas yra nauja eilutė, o kiekvieno įrašo reikšmės atskiriamos tam tikru simboliu, dažniausiai kableliu (angl. comma).

Terminai:

Lastęlinis automatas (angl. *Cellular automaton*) – Sistema, turinti celes, baigtinį celių būsenų skaičių, bei taisykles, kuriomis nurodomi celių būsenų pasikeitimai. Populiarus pavyzdys – „Gyvenimo žaidimas“ (angl. *Game of life*).

Turbo-greitinimas (angl. Turbo boosting) – Procesorių technologija leidžianti pagreitinti vieno ar kelių branduolių darbą esant tam tikroms sąlygoms – dažniausiai mažai temperatūrai bei kitų branduolių apkrovai.

x86 – Plačiai paplitusi procesorių architektūra naudojama daugelyje namų kompiuterių bei serverių.

Beužraktinis kodas (angl. *Lock-free code*) – programos įvesties kodas naudojantis lygiagretinimą, tačiau nenaudojantis tradicinių gijų sinchronizavimo metodų kaip monitoriai, semaforai ir t.t.

1. Įžanga

1.1. Dokumento paskirtis

Dokumento paskirtis – aprašyti automatinių žaidimo Minesweeper sprendimo metodų tyrimo analizę. Aprašyti tyrimui sukurtą prototipą, ištirtus algoritmus, parodyti tyrimo rezultatus.

1.2. Įvadas

Minesweeper – tai vienas žinomiausių žaidimų pasaulyje [15], pateikiamas kartu su *Microsoft Windows* operacinėmis sistemomis iki *Microsoft Windows 7* versijos, ir nemokamai prieinamas tolesnėse versijose per *Microsoft Windows Store* parduotuvę.

Minesweeper yra loginis žaidimas, kurį spręsti geba ne kiekvienas. Norint padaryti šį žaidimą prieinamesnį, gali būti pasitelkiamas automatinis žaidimo sprendimas, padedantis naujokams apeiti galimas keblias žaidimo situacijas, kylančias žaidžiant žaidimą įprastai.

Norint užtikrinti vartotojo pasitenkinimą, automatinis žaidimo sprendimas turi būti ir tikslus ir greitas. Šiame tyrime ištirti keli egzistuojantys sprendimo algoritmai, sukurti tikslumo bei greitaveikos patobulinimai jiems, nustatyta optimali algoritmų kombinacija teikianti aukštą tikslumą bei sparčią greitaveiką, bei sukurtas Minesweeper žaidimo prototipas su integruotu automatinio sprendimo funkcionalumu.

1.3. Tikslas

Darbo tikslas yra ištirti algoritmus skirtus automatiniam Minesweeper žaidimo sprendimui.

1.4. Uždaviniai

Darbo tikslui pasiekti keliami šie uždaviniai:

- Atlikti egzistuojančių Minesweeper sprendimo algoritmų analizę.
- Sukurti arba pasirinkti egzistuojančius automatinio Minesweeper sprendimo algoritmus tyrimui.
- Sukurti Minesweeper žaidimo bei automatinio sprendimo programos prototipą.
- Atlikti automatinių Minesweeper sprendimo algoritmų tyrimą.

2. Analizė

2.1. Analizuojama veikla ir projekto motyvacija

Analizuojama veikla yra kompiuterinio žaidimo „Minesweeper“ sprendimas. Minesweeper yra loginis kompiuterinis žaidimas, kuriame žaidėjo tikslas – pasitelkiant loginį mąstymą rasti visas minas paslėptas minų lauke. Daugumą žaidimo situacijų įmanoma lengvai išspręsti, tačiau priklausant nuo pasirinkto sunkumo lygio gali būti ir tokių, kurios sunkiai sprendžiamos žaidėjams, ir net automatiniams sprendimo algoritmams. Pasitaikius sunkiai išsprendžiamai situacijai, žaidėjams galėtų patikti užuominos, ar pilnas šios situacijos išsprendimas.

Tiriamas projektas – žaidimo sprendimo automatizavimas, galintis greitai pateikti atsakymus sunkioms situacijoms. Dabartiniai klasikiniais algoritmai šių situacijų arba neišsprendžia arba sprendžia per ilgai. Šiai problemai išspręsti panaudota skirtingų algoritmų kombinacija, turinti skirtingus tikslumo bei greitaveikos parametrus, algoritmai optimizuoti greitaveikai, panaudotas greitinimas pasitelkiant grafinio apdorojimo koprocesorių. Taip pat sukurta pilną programa, veikianti ne tik kaip žaidimo sprendėjas, bet ir kaip pats Minesweeper žaidimas, tam kad sukurti vartotojams patogią žaidimo bei automatinio sprendimo integraciją.

2.2. Minesweeper

„Minesweeper“ – tai žaidimas, kuriame žaidėjas „išminuoja“ lauką pasitelkiant loginį mąstymą.

2.1 pav. „Minesweeper“ žaidimo langas

Žaidimo tikslas – atidaryti visus langelius žaidimo lentoje, išvengiant tų, kuriuose yra minos. Žaidėjui parodomos užuominos – kiekviename atidarytame langelyje gali būti skaičius, kuris parodo kiek minų, įskaitant kampus, yra aplinkiniuose langeliuose. Skaičiaus nebuvimas tolygu skaičiui 0. Atradus 0, yra atidaromi visi aplink esantys langeliai, tai kartojama rekursyviai. Žaidėjui dažniausiai leidžiama padėti žymas langeliuose, kurių tikslas - padėti atsiminti, jog langelyje yra mina. Neteisingas žymos padėjimas neturi neigiamų pasekmių žaidimui, apart to, jog yra matoma neteisinga žyma, ir ja pasikliaujant bus padaroma daugiau klaidų ateityje. Žaidėjui rodomas likęs minų skaičius – t.y. bendras lauke esantis minų skaičius atėmus padėtų žymų skaičių. Vienintelis būdas laimėti –

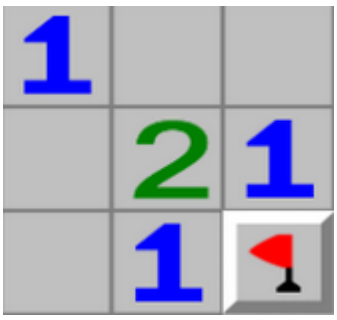
teisingai atidaryt visus langelius kuriuose yra mina, vienintelis būdas pralaimėt – atidaryti langelį su mina. Ne visada įmanoma priimti garantuotą sprendimą – kartais tenka ir spėlioti.

Yra daug šio žaidimo variacijų – kai kuriose langeliai ne kvadratiniai o trikampiai ar šešiakampiai, kitose – laukas begalinis, gali būti kelios minos viename langelyje, yra 3D ir net 4D variantų, bei žaidimų daugeliui žaidėjų internete.

Tiriamas bus klasikinis žaidimo variantas – kvadratiniai langeliai, tik viena mina per vieną langelį, yra rodomas likęs minų skaičius. Pirmas spėjimas niekada negali būti mina.

2.3. Žaidimo sudėtingumas

Žaidimo sudėtingumui suprasti svarbu paminėti užuominų bei minų neprieštaringumą (angl. *consistency*).



2.2 pav. Prieštaringa situacija

2.2 pav. Prieštaringa situacija matoma situacija, kuri logiškai neįmanoma - prie užuominos 2 yra tik vienas langelis, taigi ši situacija laikoma prieštaringa sau.

Neprieštaringumo tikrinimas blogiausiu atveju yra NP-pilno asimptotinio sudėtingumo, t.y. tiek pat sudėtingas kiek sudėtingiausias NP sunkumo uždavinys, ir tuo pačiu metu pats yra NP. Tai 2000 metais įrodė Richard Kaye sukonstruodamas iš minų lauko loginius elementus, kurie sugeba pernešti loginę informaciją, taip sulygindamas Minesweeper neprieštaringumo tikrinimą su SAT uždaviniu[6].

SAT buvo pirmasis uždavinys įrodytas esantis NP-pilno sudėtingumo 1971 metais Stephen A. Cook bei nepriklausomai už dviejų metų Leonid Levin, sukūrus *Cook-Levin* teoremą[4].

Tai aktualu, nes tai įrodo, jog jei priimtume, jog neprieštaringumo tikrinimas yra būtinas tobulam Minesweeper sprendimui - tai reikštų jog neįmanoma garantuotai išspręsti polinominiu laiku, ir esant sudėtingai situacijai, nesvarbu koks algoritmas bus sukurtas - geriausia ką įmanoma daryti tai taikyti kokias nors aproksimacijas. Tai kol kas nėra įrodyta, taigi nėra įrodyta jog Minesweeper pats savaime yra NP-pilno sunkumo bet algoritmiškai žiūrint, Minesweeper gana sunkus vienaip ar kitaip[10].

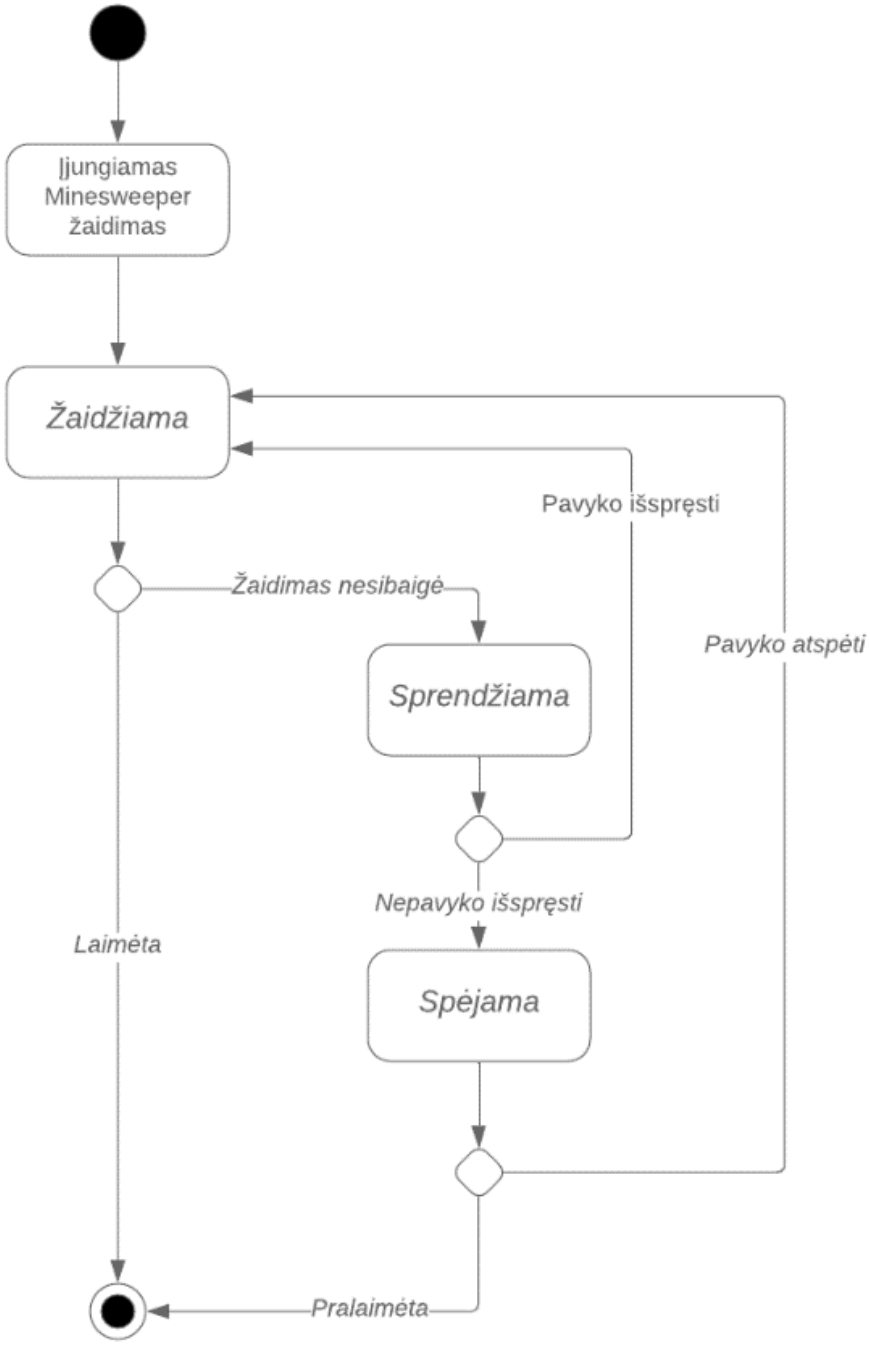
2.4. Projekto vartotojai ir klientai

Projekto vartotojai – Minesweeper žaidimo žaidėjai. Šie žaidėjai yra galutiniai produkto vartotojai, kuriems bus skirtas galutinis produktas. Šie vartotojai bus pažįstami su projekto sritimi – žaidimu, ir turėtų lengvai perprasti teikiamas idėjas. Vartotojai turės bent bazines žinias kaip naudotis programa. Prognozuojamas amžius – nuo 10 iki 70 metų, tiek vyrai tiek moterys.

Žaidėjams bus aktualu gauti kuo tikslesnius ir greitesnius pasiūlymus, ypač sunkioms situacijoms žaidime.

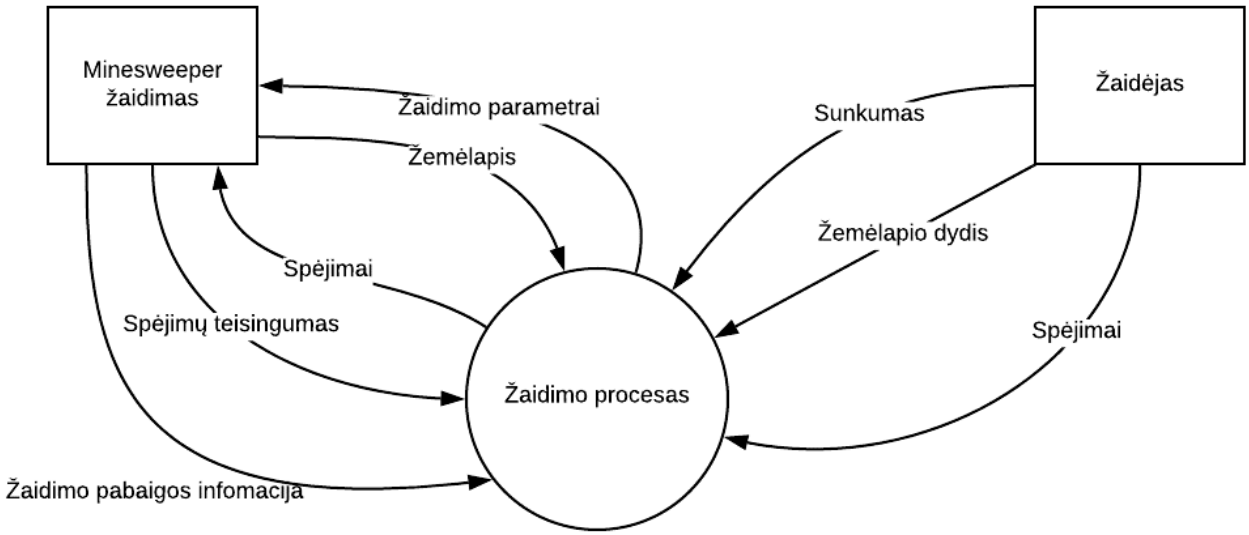
2.5. Rankinis žaidimas

Esama padėtis, kurią paveiks kuriamas projektas, matoma **2.3 pav.** Esamos padėties veiklos diagrama.



2.3 pav. Esamos padėties veiklos diagrama

Dabartinio veiklos konteksto diagrama matoma **2.4 pav**. Veiklos konteksto diagrama.



2.4 pav. Veiklos konteksto diagrama

2.6. Bendra automatinio sprendimo strategija

Minesweeper žaidimo metu galima atlikti tris veiksmus – pažymėti langelį kaip garantuotai miną (žymima vėliavėle), pažymėti kaip galbūt miną (žymima klaustuku), ir langelį atidaryti.

Potencialios minos žymėjimas klaustuku yra patogumas vartotojui, tačiau nesuteikia jokios naudos sprendžiant algoritmiškai, nes kiekviena nauja žaidimo situacija yra įvertinama iš naujo, ir kiekviename neatidarytame langelyje taip ar taip gali būti mina. Šis žymėjimas nenaudojamas tyrimuose. Garantuotos minos pažymėjimas yra naudingas, tai leidžia sekančiose sprendimo iteracijose ignoruoti šį langelį, nesprendžiant jo iš naujo, taigi esant galimybei garantuotai rastos minos yra pažymimos.

Minų pažymėjimas nesuteikia papildomos informacijos apie žemėlapį, norint gauti papildomos informacijos būtina atidaryti langelį tam, kad pamatyti jame esančia užuominą apie kaimynines minas. Sprendžiant keliais skirtingais algoritmais, yra logiška sustabdyti sprendimo iteraciją radus bent vieną garantuotai saugų langelį ir nesitelkti galingesnių tačiau lėčiau veikiančių algoritmų, nes gavus papildomą užuominą, yra labiau tikėtina jog žaidimo situacija išsprendžiama paprastais, greičiau veikiančiais algoritmais.

2.5 pav. Neišsprendžiama žaidimo situacija

Kai kuriose situacijose neįmanoma rasti garantuotai saugaus langelio, ir tenka spėti, tai vaizduojama **2.5 pav.** Neišsprendžiama žaidimo situacija. Šiuo atveju neįmanoma išgauti daugiau informacijos, tačiau kitais atvejais gali būti įmanoma toliau žaisti žaidimą, ir galimai gauti daugiau informacijos padedančios išspręsti situaciją, arba sumažinti spėjimo riziką. Visais atvejais spėjimai paliekami galui, jei nėra jokios kitos išeities. Jei algoritmas gali nustatyti tikimybę rasti miną turėtų būti spėjamas langelis su mažiausia tikimybe rasti miną.

2.7. Egzistuojantys automatiniai sprendimai

Kandangi rinka yra jauna, rinkoje kol kas neegzistuoja konkurencija. Visi rasti sprendimai yra tiriamojo pobūdžio ir nėra kurimas vartojimui paruoštas produktas; prie daugumos tyrimų net nėra pateikiamas nei išeities kodas, nei sukompiliuota programa.

2.7.1. Vieno taško metodas

Tai turbūt paprasčiausias įmanomas sprendimo būdas. Žiūrint į vieną užuominą, patikrinami kaimynai. Jei kaimynų tiek, koks užuominos skaičius - reiškia jie visi minos, ir yra pažymimi. Jei pažymėtų kaimynų skaičius lygus užuominai, reiškia visi kiti nepažymėti kaimynai nėra minos, ir gali būti saugiai atidaryti[2].

Šis algoritmas yra trivialus, ir nesugeba išspręsti situacijų kuriose atsakymui gauti reikalinga informacija iš daugiau nei vienos užuominos.

2.7.2. Minesweeper kaip ląstelinis automatas

Šis metodas pirmą kartą aprašytas Andrew Adamatzky 1997 metais. Šiame algoritme Minesweeper paverčiamas į ląstelinį automatą, t.y. kiekviena celė ar ląstelė mąsto už save, ir vykdo būsenos pakeitimus pagal savo kaimynus. Šiame konkrečiame ląstelinio automato variante ląstelės žiūri į save, kaimynus, bei kaimynų kaimynus, ir pagal aprašytas taisykles gali vykdyti būsenos pakeitimus iš nepažymėtos į pažymėtą, arba iš neatidarytos į atidarytą[1].

Šio algoritmo pagrindinis trūkumas - tendencija užstrigti prie tam tikrų specifinių konfigūracijų, kuriose reikia didesnės įžvalgos, ar galimybės atlikti spėjimą[2].

2.7.3. Minesweeper kaip CSP

Minesweeper įmanoma spręsti kaip CSP. Sudaromos lygtys, kuriose 1 = mina, 0 = saugu, lygtys prastinamos, ir randami saugūs langeliai, kurie atidaromi tam, kad gauti daugiau informacijos. Jei saugių langelių rasti nepavyksta, vyksta keliavimas atgal, ir sudaromas minų tikimybių sąrašas, pagal kurį galima atlikti spėjimą mažiausią tikimybę rasti miną turimam langeliui[12].

Originaliame šaltinyje tai nepaminėta, bet sudarytos tiesinės lygtys galėtų būti efektyviai prastinmos paverčiant jas į kintamųjų koeficientų svorių matricą, ir ją sprendžiant Gauso metodu[9]. Nors po to norint išgauti tikimybių sąrašą, vis tiek reikėtų taikyti atgalinio ėjimo algoritmą, ar kokią nors jo variaciją.

Kartais net langelis turintis mažesnę tikimybę turėti miną nėra optimalus. Taip yra todėl, kad atvėrus rizikingesnį langelį, gali būti gaunama naudingesnė informacija, nei atvėrus mažiau rizikingą langelį.

Naudojant CSP algoritmą kaip pagrindą, įmanoma pritaikyti medžio paiešką, ir rasti optimalius ėjimus atsižvelgiant į ateitį[11].

2.7.4. Minesweeper pasitelkiant ILP

Jau 2003 metais buvo pasiūlytas Minesweeper sprendimo būdas pasitelkiant ILP per bendro naudojimo programą *Mio*. Yra sugeneruojama didelė duomenų bazė su žaidime sutinkamomis situacijomis, bei teisingais jų sprendimais. Pasitelkiant *Mio* generuojamos taisyklės, pagal kurias būtų atliekami veiksmai žaidime. Sugeneruotos taisyklės tikrinamos pagal korektiškumą, ir neteisingos yra atmetamos. [3]

2.7.5. Minesweeper pasitelkiant sustiprinamąjį mokymąsi

2003 metais taip pat buvo pasiūlytas būdas spręsti Minesweeper pasitelkiant RL (angl. *Reinforcement Learning*) [8]. Siūloma gana paprasta atlygio funkcija – 1 jei atspėjama teisingai, 0 jei atspėjama neteisingai.

2.7.6. Minesweeper pasitelkiant Gauso redukciją

2013 metais Robertas Massaioli pasiūlė būdą spręsti Minesweeper formalizuojant žaidimo žemėlapį kaip išplėstinę matricą [7]. Neatidaryti langeliai bei užuominos joms paverčiami į kintamuosius bei lygčių sistemas joms, jos paverčiamos į išplėstinę matricą, kuri yra sprendžiamą Gauso metodu.

2.7.7. Minesweeper naudojant permutacijų išbandymą

Minesweeper gali būti sprendžiamas tiesiog išbandant visas minų permutacijas, ir žiūrint kurios iš jų logiškai atitinka aplink esančias užuominas [13]. Iš esmės tai yra SAT uždavinio sprendimas, kuriame minų radimas yra perrinkimo logika. Pasitelkiant šį sprendimo būdą galima nustatyti ne tik minų buvimo ar nebuvimo faktą, bet taip pat ir tikimybes rasti minas konkrečiuose langeliuose.

3. Projektinė dalis

Šiame skyriuje aprašoma sukurto prototipo reikalavimai, architektūra, bei pateikiamas galutinis prototipo vaizdas. Kuriamas prototipas reikalingas algoritmų analizei, bei Minesweeper žaidimo bei automatinio sprendimo integracijai. Projektas nėra apmokamas.

3.1. Architektūros tikslai ir apribojimai

Programinės įrangos tikslai ir reikalavimai, turintys esminį poveikį architektūrai:

- Sistema turėtų turėti grafinę sąsają.
- Sprendimų radimas turėtų veikti aproksimuojant sprendimus esant sudėtingoms situacijoms.
- Sistema turėtų vykdyti visus panaudotus algoritmus realiu laiku.
- Sistemoje turėtų būti integruotas Minesweeper žaidimas.
- Sistema turėtų leisti išsaugoti bei užkrauti žaidimo būseną.

3.2. Panaudos atvejai

Vartotojas turėtų galėti:

1. **Rankiniu būdu nurodyti žemėlapį.** Žemėlapis nurodomas kaip M x N dydžio simbolių matrica. Simbolių paaiškinimas pateikiamas

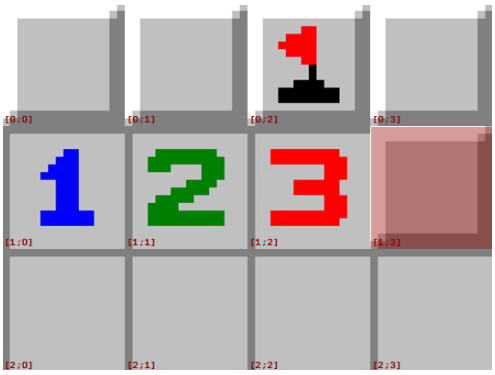
3.1 lentelė. Rankinės žemėlapio įvesties simbolių paaiškinimai

Simbolis	**Aprašymas**
X	Siena, su šiuo laukeliu neįmanoma atlikti veiksmų
#	Neatidengtas laukelis
!	Žinoma neatidengta mina
.	Atidengtas tuščias laukelis
1 - 8	Atidengtas tuščias laukelis su užuomina, kur nurodytas skaičius yra aplink esančių minų skaičius
m*n*	Rašomas tik gale, nurodo likusį nerastų minų kiekį žemėlapyje. Pvz. m8 reikštų, jog likę 8 minos.

Pavyzdžiui, rankinė įvestis:

```
##!#
123#
....
m3
```

Taps žemėlapiu vaizduojamu **3.1 pav.** .



3.1 pav. Rankinės žemėlapio įvesties vaizdavimas

2. **Įvesti žaidimo parametrus.** Žaidėjui norint pradėti naują žaidimą, privalo būti įvedami žaidimo parametrai, kurie nurodys žemėlapio aukštį, plotį, bei pasirinktinai minų kiekį arba minų tankumą.

3. **Žaisti žaidimą.** Žaidžiant žaidimą, žaidėjas gali spustelti ant neatidengto laukelio kairiuoju pelės klavišu, ir jį atidengti, arba spustelti dešiniuoju pelės klavišu ir pažymėti, jog langelyje yra mina.

4. **Išsaugoti žaidimą.** Žaidimą galima išsaugoti, norint pratęsti jį vėliau. Žaidimas išsaugomas įrašant užkoduotą žaidimo būseną į failą saugomą vartotojo programų aplanke.

5. **Užkrauti žaidimą.** Žaidėjas gali pasirinkti pratęsti anksčiau pradėtą ir išsaugotą žaidimą pasirenkant bei užkraunant anksčiau išsaugotą žaidimo sesijos failą.

6. **Pasirinkti sprendimo algoritmą bei parametrus.** Norint automatiškai spręsto žaidimo situacijas, galima pasirinkti sprendimo algoritmą, bei nurodyti jo parametrus. Šis žingsnis nebūtinas, jo neatlikus bus pagal nutylėjimą naudojami standartiniai optimalūs tyrimo metu nustatyti parametrai.

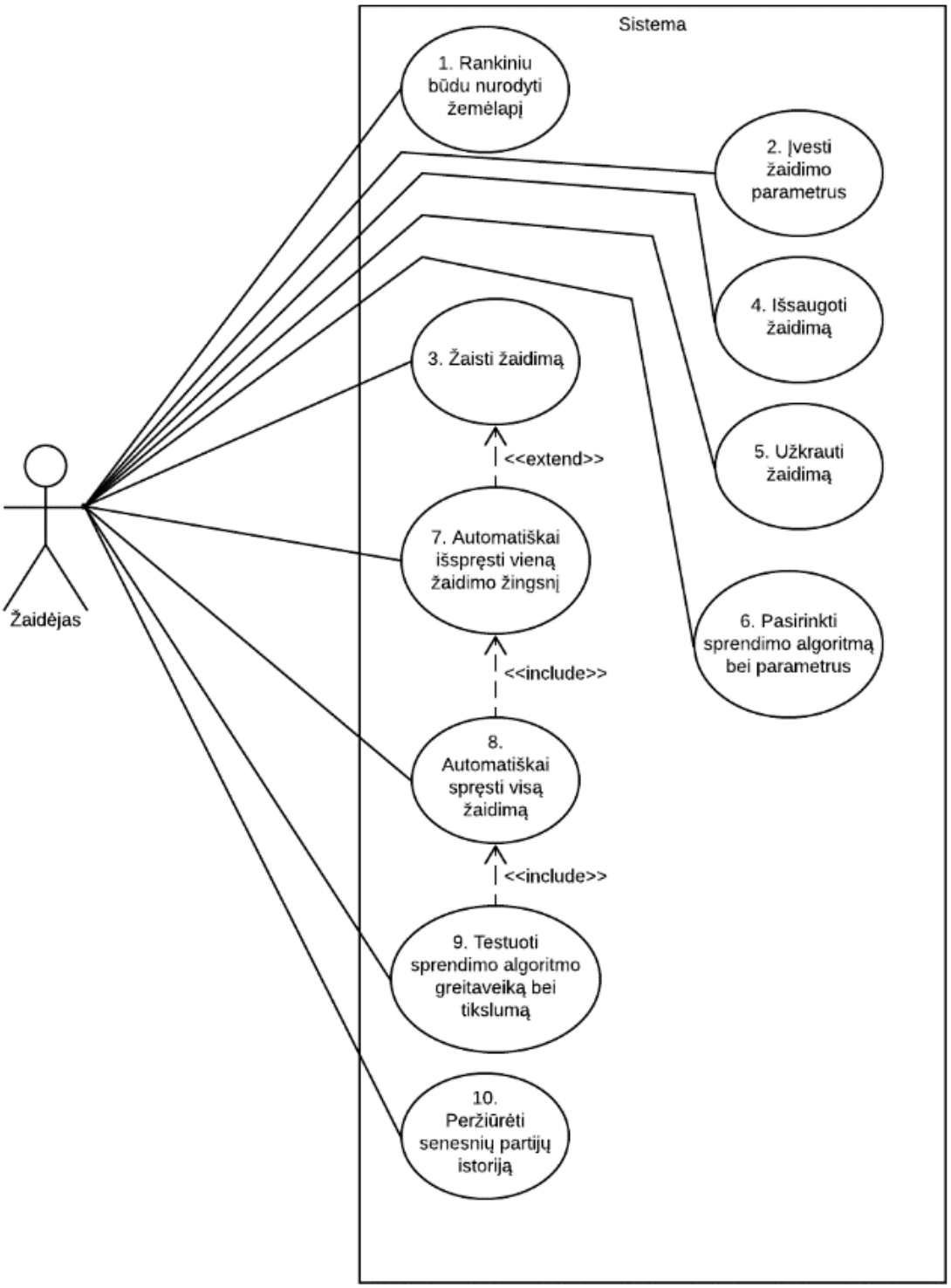
7. **Automatiškai išspręsti vieną žaidimo žingsnį.** Žaidžiant žaidimą ar rankiniu būdu nurodžius žemėlapį galima spręsti žaidimo žemėlapį, šį veiksmą atlieka sprendimo algoritmas pagal sukonfigūruotus nustatymus.

8. **Automatiškai išspręsti visą žaidimą.** Vykstant žaidimui, o ne rankiniu būdu įvestos situacijos sprendimui, žaidėjas gali pasirinkti kad programa automatiškai žaistų visą likusį žaidimą automatiškai, pagal parinktus automatinio sprendimo nustatymus.

9. **Testuoti sprendimo algoritmo greitaveiką bei tikslumą.** Žaidėjas gali testuoti pasirinktų automatinio sprendimo nustatymų greitaveiką bei tikslumą, bus sugeneruojama ataskaita *csv* formatu, kurią būtų galima užkrauti į norimą peržiūros programą, pvz. *Microsoft Excel*.

10. **Peržiūrėti senesnių partijų istoriją.** Vartotojas gali matyti istoriją seniau žaistų partijų, kurioje būtų atvaizduojama partijos trukmė, žaidėjo slapyvardis, bei panaudotų automatinių sprendimų kiekis.

Bendra panaudos atvejų diagrama vaizduojama **3.2** pav. Panaudos atvejų diagrama



3.2 pav. Panaudos atvejų diagrama

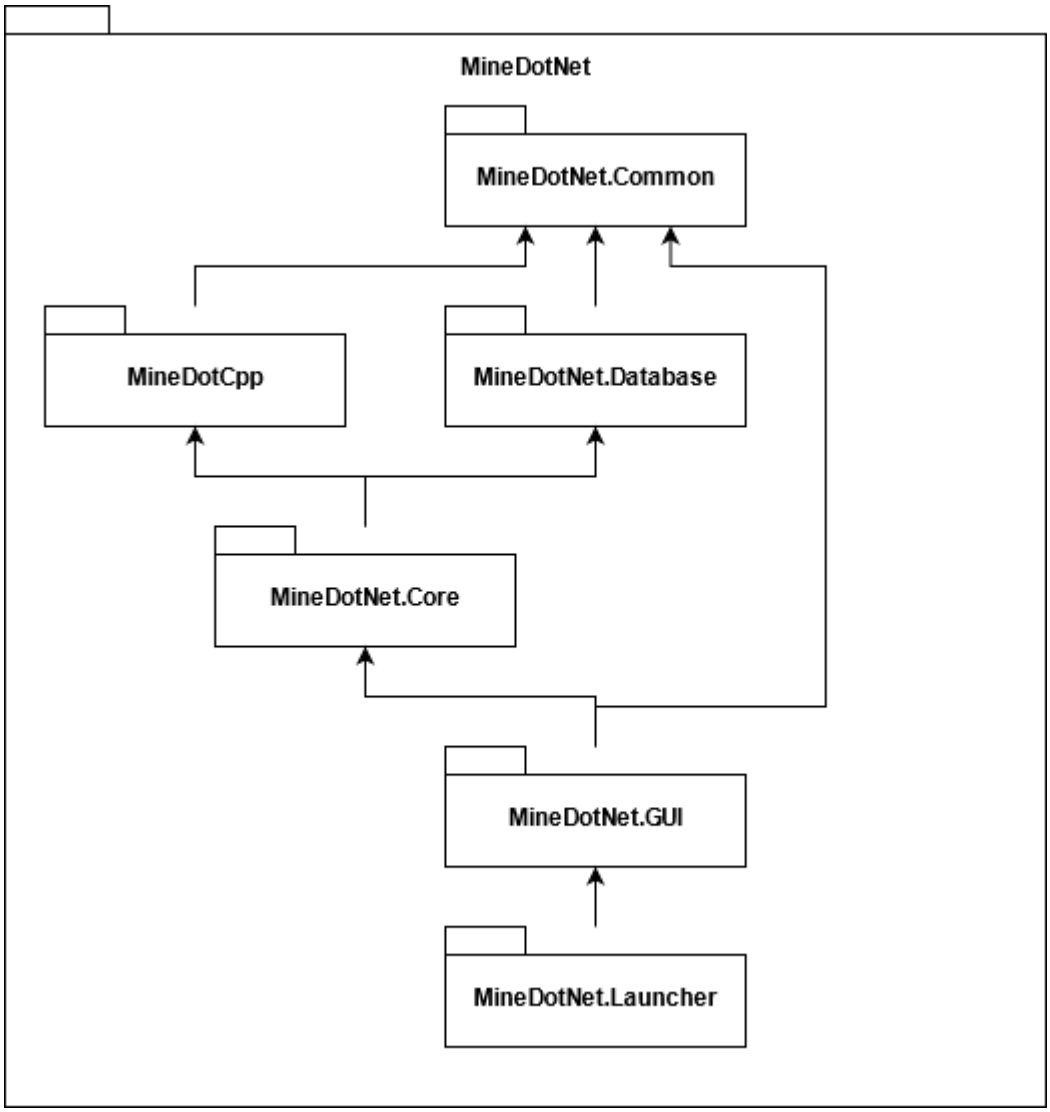
3.3. Sistemos statinis vaizdas

Kadangi sistemai nereikia serverio dalies, visa sistema bus pateikiama kaip viena programa, kuri bus suskaidyta į šias bibliotekas:

- *MineDotNet.Common* – Bazinis paketas esminiams duomenų tipams naudojamiems visose programos dalyse.
- *MineDotNet.Database* – Paketas skirtas bendravimui su *SQLite 3* įterptine duomenų baze.

– *MineDotCpp* – Klasikinių algoritmų paketas, realizuotas su C++ programavimo kalba.
– *MineDotNet.Core* – Centrinis sprendimo paketas, paleidžiantis visus sprendimo algoritmus bei žaidimo logiką.
– *MineDotNet.GUI* – Vartotojo sąsajos paketas.
– *MineDotNet.Launcher* – Paleidimo programa.

Kuriamo projekto architektūros suskaidymą į paketus galima matyti **3.3 pav.** paketų diagramoje.



3.3 pav. Paketų diagrama

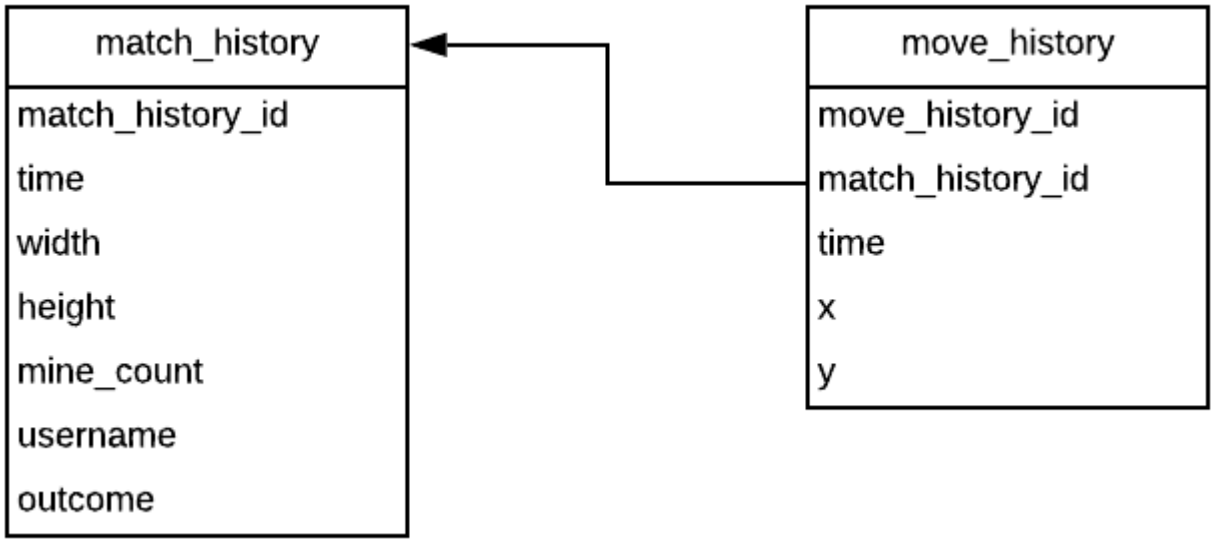
Kuriama sistema turės vartotojo sąsają, kuri bus suprogramuota su C# programavimo kalba, ši kalba pasirinkta dėl ankstesnio susipažinimo, bei kelių esminių kriterijų atitikimo:

- C# yra aprašoma tvirtai.
- C# galima naudoti Windows programų kūrimui.
- C# turi biblioteką darbui su mašininiu mokymusi.

Kadangi C# kalba parašytos programos turi lėtesnę greitaveiką nei lyginant su kai kuriomis kitomis kalbomis, pvz. C, C++, Rust ar Go [14], klasikinių algoritmų kūrimui pasirinkta naudoti C++ kalbą dėl greitaveikos priežasčių.

3.4. Saugojamų duomenų vaizdas

Įterptinės duomenų bazės modelio schema matoma **3.4 pav.**.



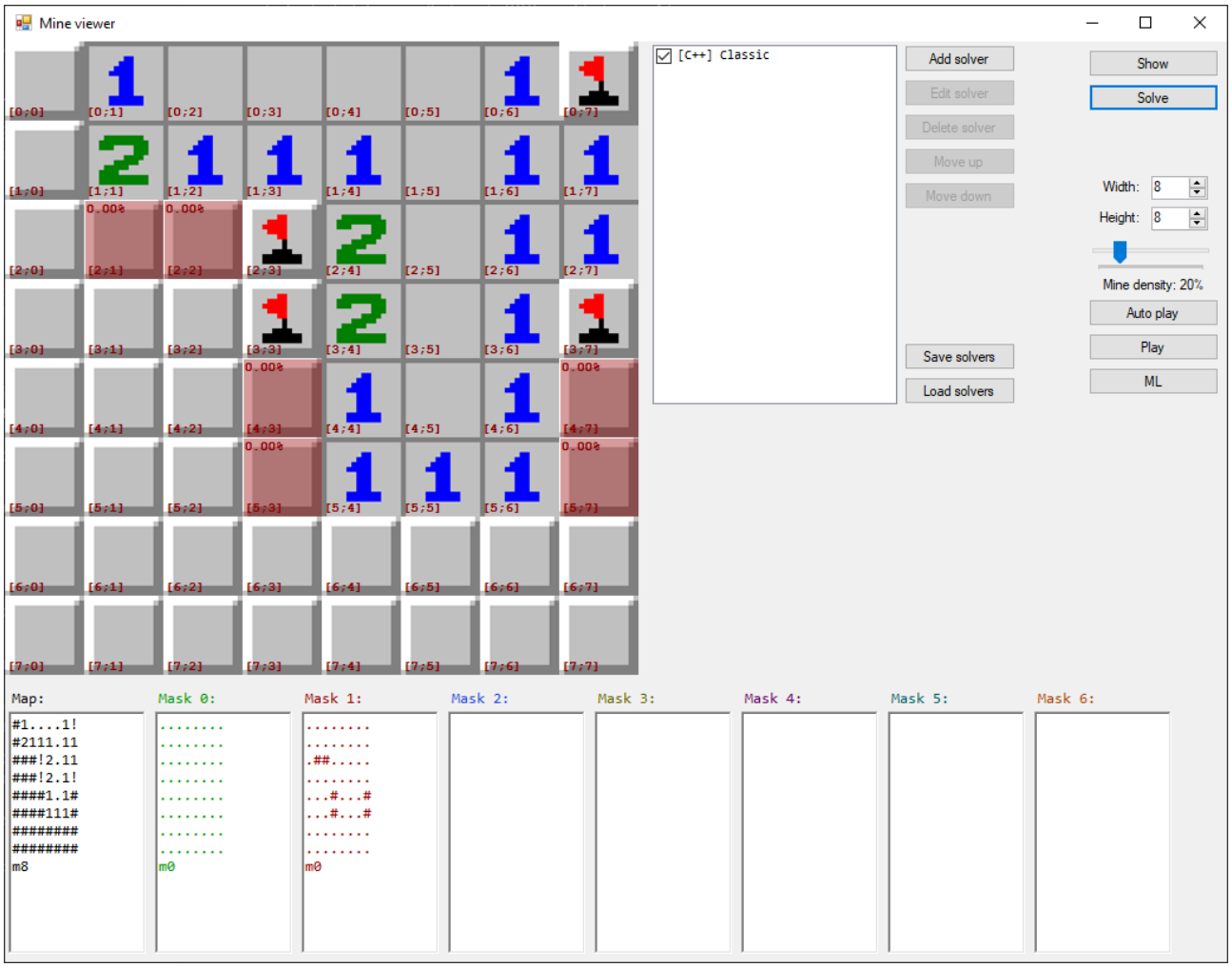
3.4 pav. Įterptinės duomenų bazės modelio schema

3.5. Kokybė

Programinės įrangos architektūra sukurta su mintimi jog ją būtų galima be kada praplėsti naujais klasikiniais ar mašininio mokymosi sprendimo algoritmais, ir integruotas greitaveikos testavimas leis greitai ištestuoti naujai įdiegtis algoritmus.

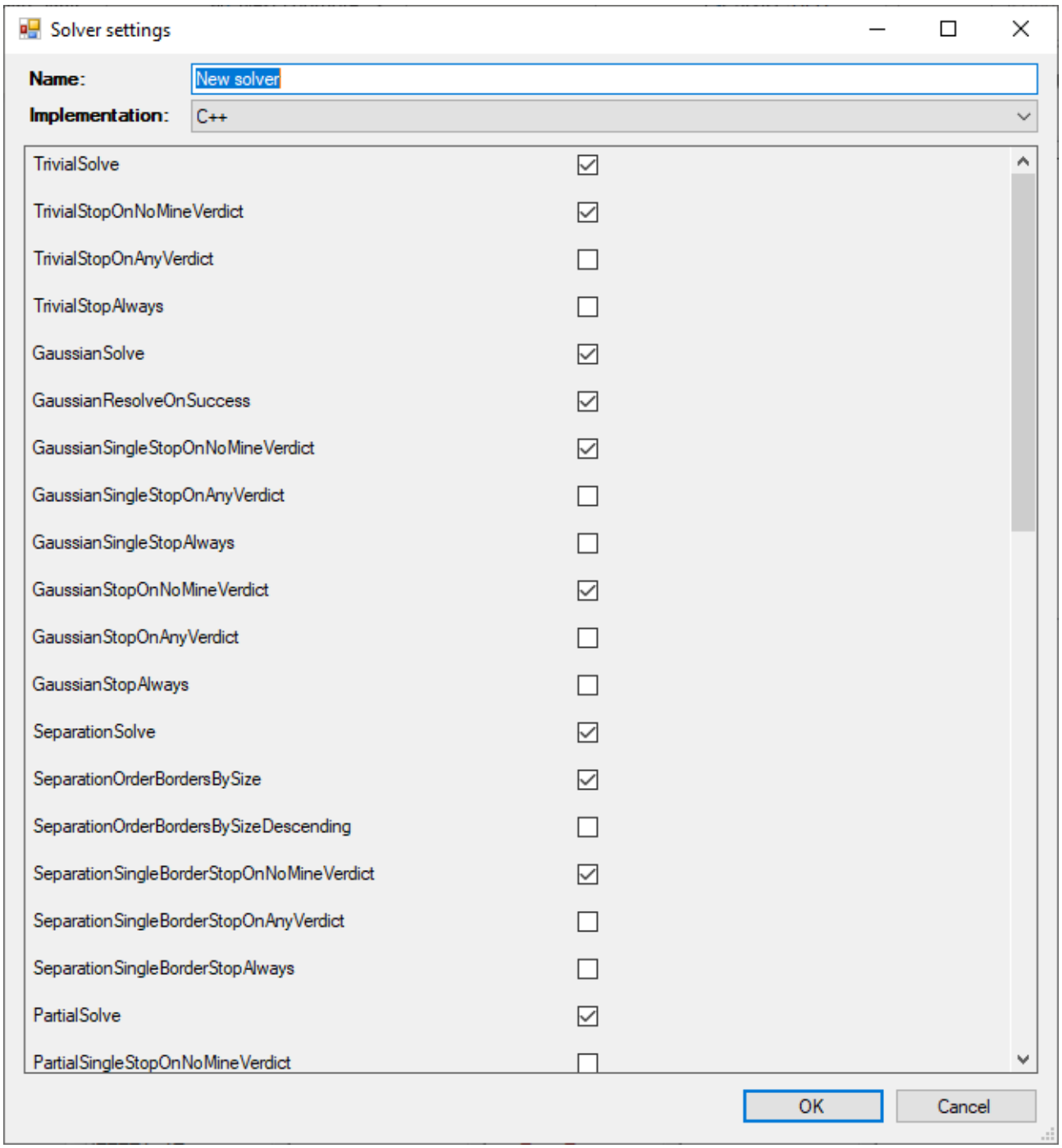
Visa sistema rašoma atsižvelgiant į kertinius objektinio programavimo principus, *SOLID* principus bei, Microsoft C# kodo konvencijas ir kodo kokybės rekomendacijas. Algoritmų kūrimui C++ bibliotekoje duodama didesnė laisvė – leidžiama mažinti kodo kokybę dėl greitaveikos priežasčių.

3.6. Sukurtas prototipas



3.5 pav. Pagrindinis PĮ sąsajos langas

Pagrindinė vartotojo sąsaja yra vienas langas, teikiantis visas sistemos funkcijas, vaizduojamas **3.5 pav.** Pagrindinis PĮ sąsajos langas. Kairėje pusėje matomas žaidimo langas, kuriame vartotojas gali pats bandyti atspėti galimas žaidimo situacijas. Per vidurį viršuje matomi įjungta sprendimo algoritmų biblioteka, dešinėje žaidimo konfigūracija. Apačioje – derinimui bei vartotojų pasirinktų žaidimo situacijų įvedimui skirta informacija.



3.6 pav. Automatinių sprendimo pasirinkimų langas

Automatinių sprendimo nustatymų pasirinkimo langas vaizduojamas **3.6 pav.** Automatinių sprendimo pasirinkimų langas. Vartotojui paliekama laisvė valdyti automatinio sprendimo algoritmus. Vartotojui šių nustatymų nekeičiant paliekami optimalūs tyrimo metu nustatyti sprendimo parametrai.

4. Tyrimas

Aprašomi keli skirtingi sprendimo algoritmai ir jų kombinacijos. Kiekvienas skyrelis remiasi ankstesnio skyrelio rezultatais – kiekviename skyrelyje aprašomas algoritmas ar algoritmo patobulinimas yra naudojamas kartu su anksčiau aprašytais algoritmais, lyginama bei vertinama naujai aprašomo algoritmo suteikiama nauda bendram sprendimui.

4.1. Tyrimo tikslas bei kriterijai

Tyrimo tikslas – išmatuoti algoritmų tikslumą, greitaveiką, bei nustatyti optimalią algoritmų panaudojimo strategiją. Detaliau tyrimo kriterijai vaizduojami **4.1 lentelė.** Tyrimo vertinami algoritmų kriterijai.

4.1 lentelė. Tyrimo vertinami algoritmų kriterijai

Nr.	Kriterijus	Pagrindimas	Pagrindimas
1	Tikslumas	Kiek dažnai sprendimo algoritmai geba pilnai sužaisti pilną Minesweeper partiją esant skirtingiems sudėtingumo parametrais. Tikslumas turėtų būti kuo didesnis.	Kadangi tiriamos sistemos paskirtis yra sėkmingai išspręsti Minesweeper, sprendimo tikslumas yra akivaizdus tirtinas kriterijus.
2	Sprendimo laikas	Kiek ilgai sprendimo algoritmai užtrunka sužaisti pilną Minesweeper partiją esant skirtingiems sudėtingumo parametrais. Sprendimo laikas turėtų būti kuo mažesnis.	Esant per ilgam sprendimo laikui vartotojams gali atsibosti laukti atsakymo.

4.2. Tyrimo metodika

Minesweeper žaidimas turi tris parametrus valdančius žaidimo sunkumą – minų kiekį, žemėlapio plotį ir aukštį. Yra trys standartiniai sunkumo lygiai, jų parametrai rodomi **4.2 lentelė.** Standartiniai Minesweeper sunkumo lygiai.

4.2 lentelė. Standartiniai Minesweeper sunkumo lygiai

Pavadinimas	Plotis	Aukštis	Plotas	Minų kiekis	Minų tankumas
Pradedantysis (angl. *Beginner*)	9	9	81	10	12.3%
Vidutinis (angl. *Intermediate*)	16	16	256	40	15.6%
Ekspertas (angl. *Expert*)	30	16	480	99	20.6%

Ankstesnis tyrimai parodė, jog ir didėjantis žemėlapio plotas ir minų tankumas sunkina sprendimo procesą[2]. Kadangi minų tankumas turi didesnį poveikį sprendimo rezultatams, algoritmų efektyvumas bus testuojamas keičiant minų tankumą ir paliekant žemėlapio dydį fiksuotą. Bus atliekamas vienas testas keičiant žemėlapio dydį, skirtas patikrinti ankstesnio tyrimo rezultatus.

Kiekvienas algoritmas ar algoritmų kombinacija bus testuojama su 16 pločio bei 16 aukščio žemėlapiu. Kiekvienam testui minų kiekis bus didinamas nuo 1 iki 100 minų, tai atitinka minų tankį nuo 0,4% iki 39%. Kiekvienam testuojamam minų tankiui bus automatiškai sužaidžiama po 1000 žaidimų, toks žaidimų kiekis yra reikalingas norint gauti patikimus tyrimo rezultatus. Bus fiksuojamas kiekvieno žaidimo laimėjimo arba pralaimėjimo faktas ir pilna žaidimo trukmė, iš šių rezultatų bus išvedamas vidurkis kiekvienam minų tankumui. Iš viso vykdant kiekvieną testą bus sužaidžiama 100 tūkst. žaidimų.

4.3. Tyrimo aplinka

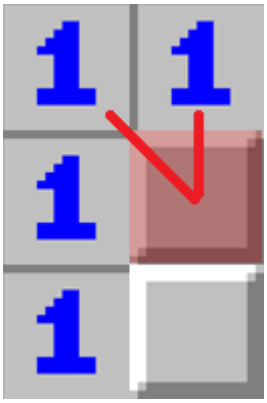
Tyrimas atliekamas su asmeniniu kompiuteriu, šio kompiuterio parametrai:

- *AMD Ryzen 9 3900X* 12 branduolių, 24 gijų procesorius.
- *NZXT X63 Kraken* procesoriaus aušintuvas.
- *Gigabyte GeForce GTX 1080 Ti Gaming OC* vaizdo plokštė turinti 11GB GDDR5X spartinančiosios atminties.
- *Crucial Ballistix RGB 2x16GB DDR4 3600 MHz CL16* operatyvioji atmintis.
- *Samsung 970 EVO Plus 1TB* standusis diskas
- *MSI MPG X570 Gaming Plus* motininė plokštė.

Procesoriaus dažnis nustatytas į 4.0 GHz ir yra visiems branduoliams fiksuotas, išjungtas „turbo-greitinimas“ (angl. *Turbo-bosting*). Dažnio fiksavimo tikslas – sumažinti rezultatų kintamumą testų vykdymo metu dėl dažnio šuolių ir darbo priskyrimo branduoliams galimai dirbantiems skirtingu dažniu. *AMD Ryzen* serijos procesoriai mažina darbo spartą didėjant temperatūrai, panaudotas vandens aušintuvas turėtų neleisti tam nutikti, maksimali užfiksuota temperatūrą pilnos procesoriaus apkrovos metu – 73C.

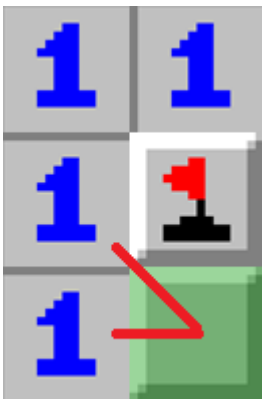
4.4. Sprendimas trivialiu metodu

Trivialiu toliau bus vadinamas šiame skyriuje aprašomas greitai veikiantis algoritmas atsižvelgiantis tik į sprendžiamos celės kaimynus. Tai turbūt paprasčiausias įmanomas sprendimo būdas [2].



4.1 pav. Minos nustatymas trivialiu algoritmu

Minos aptinkamos randant užuominas su sutampančiu kaimyninių neatidengtų langelių skaičiumi. Radus atitinkančią užuominą, visi aplinkiniai langeliai gali būti pažymėti kaip minos, Aptikimo pavyzdys rodomas **4.1 pav.** Minos nustatymas trivialiu algoritmu.



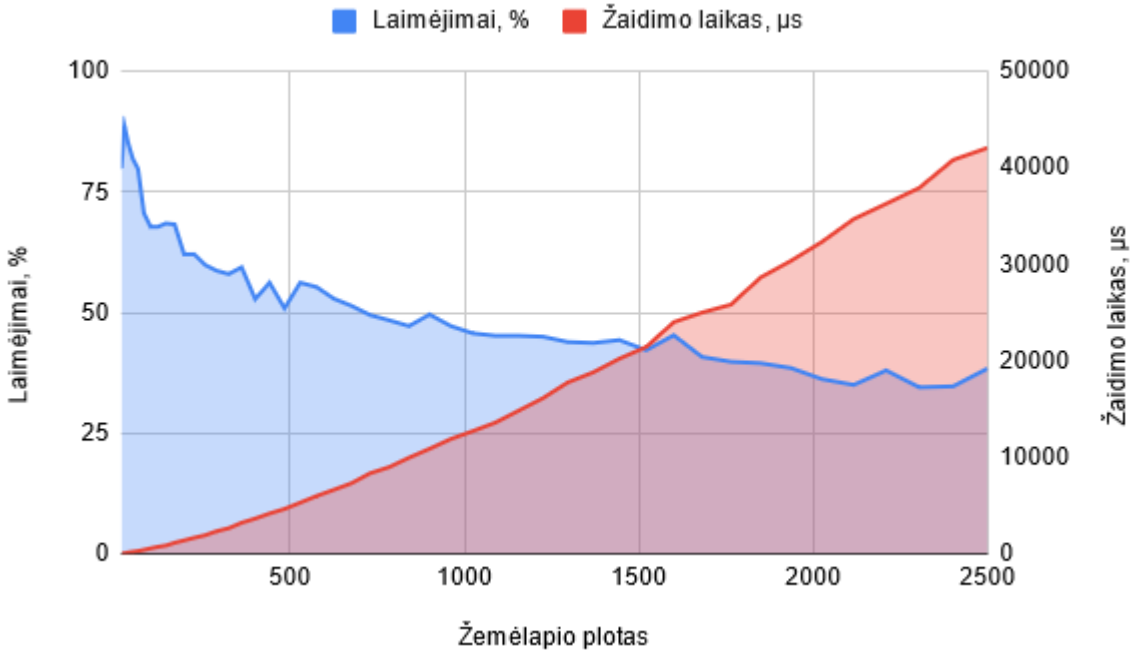
4.2 pav. Saugaus langelio nustatymas trivialiu algoritmu

Saugūs langeliai aptinkami randant užuominas kurių reikšmė sutampa su kaimyninių neatidengtų langelių skaičiumi, atmetant jau pažymėtus langelius. Aptikimo pavyzdys rodomas **4.2 pav.** Saugaus langelio nustatymas trivialiu algoritmu.

4.4.1. Trivialaus algoritmo rezultatai

4.3 pav. Trivialaus algoritmo rezultatai didėjant minų tankiui

Trivialaus algoritmo tyrimo rezultatai rodomi **4.3 pav.** Trivialaus algoritmo rezultatai didėjant minų tankiui. Blogiausiu atveju pilnas žaidimas truko tik apie 2,3 ms., ir esant mažam minų tankiui sėkmingai laimi daugelį žaidimų. Tai parodo, jog šis algoritmas gali būti sėkmingai naudojamas esant trivialioms situacijoms.



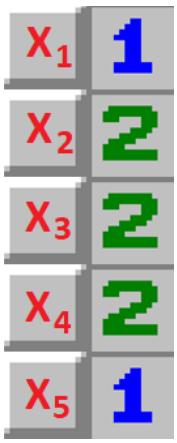
4.4 pav. Trivialaus algoritmo rezultatai didėjant žemėlapio plotui

Didėjant žemėlapio plotui išlaikant pastovų minų tankumą laimėjimų kiekis krenta, tačiau ženkliai lėčiau nei didinant minų tankumą, tai vaizduojama **4.4 pav.** Trivialaus algoritmo rezultatai didėjant žemėlapio plotui. Šių rezultatų kreivė sutampa su anksčiau atliktu tyrimu [2], tačiau konkretūs žaidimo laikai skiriasi dėl naudotos architektūros bei pasitelkto algoritmo. Taip pat matoma jog sprendimo laikas tiesiogiai proporcingas žemėlapio plotui, ir nepriklauso nuo laimėjimų kiekio.

4.5. Sprendimas Gauso metodu

Išbandytas algoritmas sprendžiantis Minesweeper kaip matricą modifikuotu Gauso redukcijos metodu. Šis algoritmas buvo sukurtas 2013 metais Robert‘o Massaioli [7].

4.5.1. Bazinis Gauso metodo algoritmas



4.5 pav. Gauso metodu sprendžiamas žemėlapis

Sprendžiamame žemėlapyje visos visi neatidaryti langeliai traktuojami kaip kintamieji. Pavyzdys vaizduojamas **4.5 pav.** Gauso metodu sprendžiamas žemėlapis. Šis žemėlapis negali būti išsprendžiamas trivialiu metodu.

$$\begin{cases} x_1 + x_2 = 1 \\ x_1 + x_2 + x_2 = 2 \\ x_2 + x_3 + x_4 = 2 \\ x_3 + x_4 + x_5 = 2 \\ x_4 + x_5 = 1 \end{cases}$$

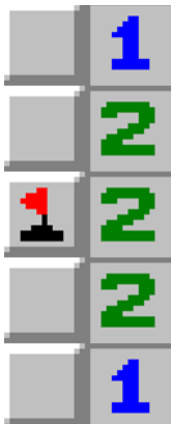
Sudaroma tiesinių lygčių sistema iš kiekvienos žemėlapyje turimos užuominos, kuriose suma tarp kaimyninių neatidarytų langelių kintamųjų lygi užuominos vertei. Tai yra Minesweeper uždavinio formalizavimas – užuominos parodo rodo minų kiekį kaimyniniuose langeliuose.

$$\left[\begin{array}{ccccc|c} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right]$$

Iš tiesinių lygčių sistemos sudaroma išplėstinė matrica. Atliekama matricos redukcija Gauso metodu.

$$\left[\begin{array}{ccccc|c} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Įprastai sprendimas Gauso metodu šiam pavyzdžiui negarantuotų unikalių sprendinių kiekvienam langelio kintamajam. Tačiau kadangi mina langelyje gali tik egzistuoti arba neegzistuoti, reikia priimti papildomą galimų reikšmių suvaržymą: $x \in [0,1]$. Turint tai omeny, galima matyti, jog vienintelis būdas kaip trečia eilutė gali turėti logišką sprendinį, yra jei $x_3 = 1$, taigi trečiame langelyje privalo būti mina. Bendrai – jei po redukcijos lygtyje yra tik vienas kintamasis su koeficientu 1, langelis turi miną jei lygties dešiniojoje pusė lygi 1. Priešingu atveju, langelis neturi minos[7]. Pavyzdžio sprendimo rezultatai rodomi **4.6 pav.** Gauso metodo sprendimo rezultatas.



4.6 pav. Gauso metodo sprendimo rezultatas

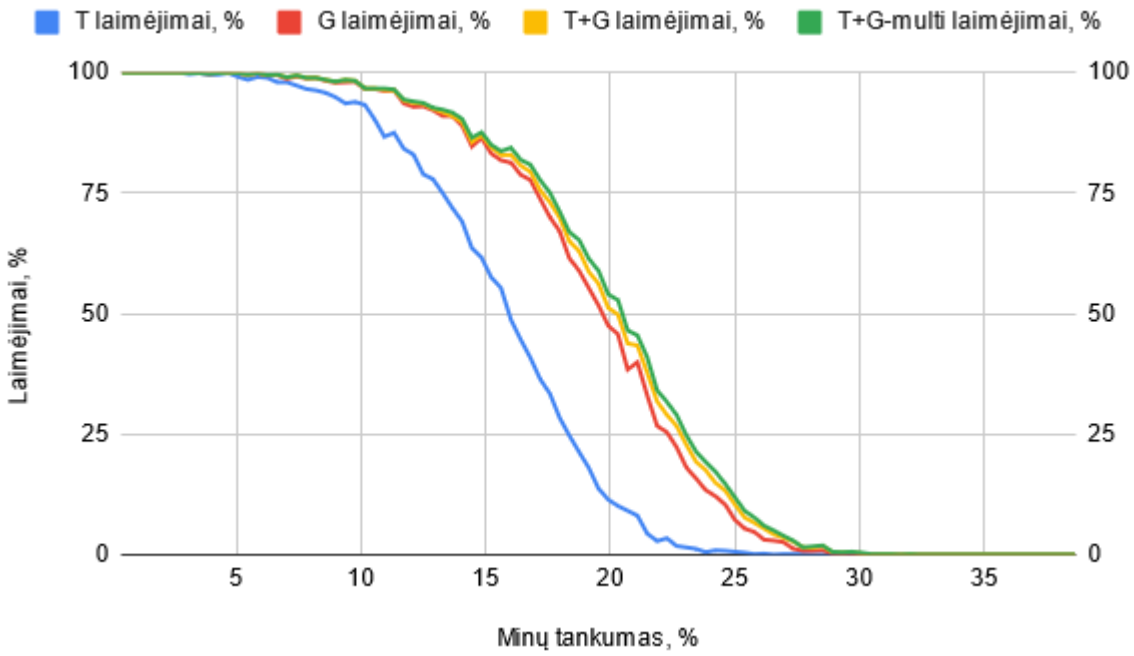
Įmanoma daryti papildomą žingsnį Gauso redukcijoje kuris padeda efektyviau rasti minas. Kiekvienai tiesinei lygčiai (arba matricos eilutei) nustatoma minimali ir maksimali reikšmė kurią galima pasiekti, kai $x \in [0,1]$, tai padaroma įstatant 0 ir 1 į visus eilutės kintamuosius atitinkamai[7]. Jei reikšmė lygties dešiniojoje pusėje sutampa su apskaičiuota minimalia reikšme, langeliai su teigiamu koeficientu neturi minos, langeliai su neigiamu koeficientu turi miną. Jei reikšmė lygties dešiniojoje pusėje sutampa su apskaičiuota maksimalia reikšme, priešingai – langeliai su teigiamu koeficientu neturi minos, langeliai su neigiamu koeficientu turi miną. Šis patikrinimas turėtų būti vykdomas atliekant kiekvieną redukcijos žingsnį, norint užtikrinti maksimalų aptikimų skaičių.

Algoritmiškai visiems atvejams nustatyti minas šiuo algoritmu NP-sunku nes atsiremiama į SAT uždavinio apribojimus[6], reikėtų daryti pilną perrinkimą, kas atitiktų kombinacijų išbandymo algoritmą aprašomą sekančiame skyriuje, tačiau veiktų lėčiau dėl papildomos kodo logikos reikalingos lygčių valdymui.

4.5.2. Pakartotinis Gauso algoritmo panaudojimas

Įmanoma pasiekti geresnių rezultatų vykdant Gauso redukciją kelis kartus, pasitelkiant skirtingus parametrus, pvz. atsitiktinį eilučių perdėliojimą kiekviename redukcijos žingsnyje, prieš atliekant SAT uždaviniui specifinius patikrinimus eilučių išskaidymui. Pasirinkta atlikti keturis atsitiktinius eilučių perdėliojimus.

4.5.3. Gauso algoritmo tyrimo rezultatai



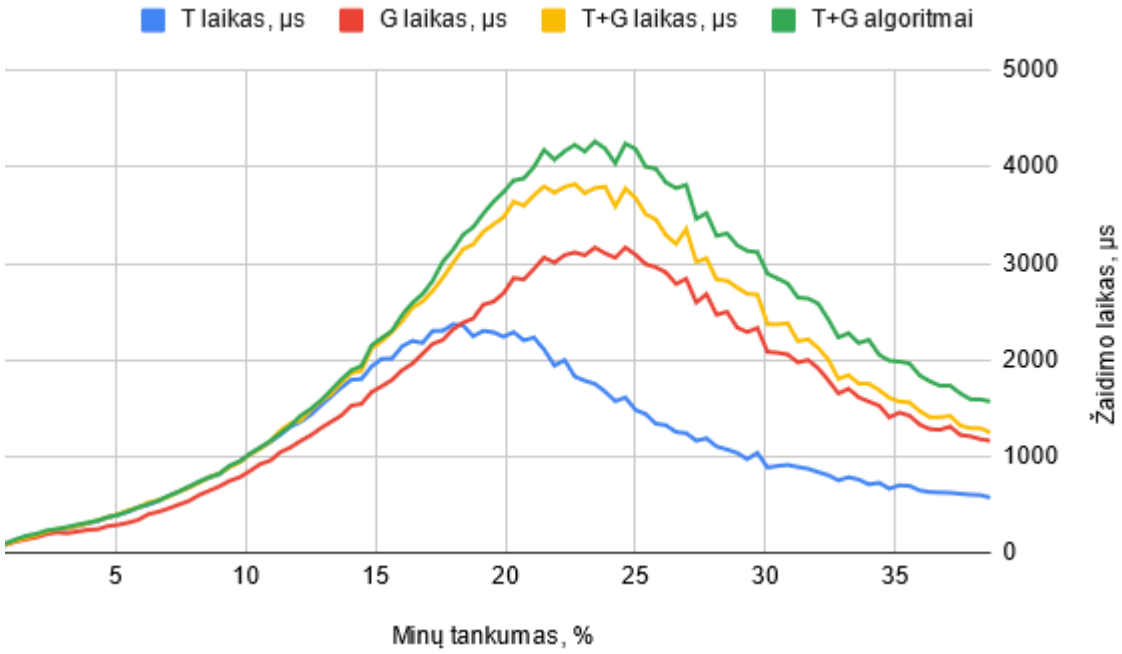
4.7 pav. Gauso metodo tikslumo tyrimas

Ištirtas Gauso metodas, bei jo panaudojimas kartu su trivialiu algoritmu. Tyrimo rezultatai matomi **4.7 pav.** Gauso metodo tikslumo tyrimas, čia T – trivialus algoritmas, G – Gauso algoritmas, T+G – pirma naudojamas trivialus algoritmas, nepavykus rasti sprendimo naudojamas Gauso algoritmas.

Gauso algoritmas veikia ženkliai tiksliau nei trivialus algoritmas. Taip pat matoma, jog trivialaus algoritmo panaudojimas prieš bandant Gauso algoritmą suteikia papildomo tikslumo. Iš to galima

spręsti jog egzistuoja situacijų kurias trivialus algoritmas išsprendžia tačiau bent sukurtas Gauso algoritmas nesugeba išspręsti.

Pakartotinio Gauso algoritmo panaudojimas suteikia papildomo tikslumo, tai itin naudinga jei Gauso algoritmas yra naudojamas ne kaip galutinis o tik kaip tarpinis algoritmas prieš pasitelkiant „sunkiasvorius“ algoritmus.



4.8 pav. Gauso metodo greitaveikos tyrimas

Greitaveikos tyrimo rezultatai rodomi **4.8 pav.** Gauso metodo greitaveikos tyrimas. Matoma, jog Gauso algoritmas veikia lėčiau nei trivialus algoritmas, tačiau atsižvelgiant į tai jog blogiausiu atveju žaidimas truko apie 3,8 ms., vartotojas nepastebėtų skirtumo. Papildomos Gauso algoritmo iteracijos pasitelkiamos tik tada, jei pirmoji iteracija nerado garantuotai saugaus sprendimo – matoma jog greitaveikos sulėtėjimas yra neženklus. Kadangi abiejų algoritmų kombinacija pasiekė geresnius rezultatus, tolimesniems tyrimams kaip etalonas bus naudojama trivialaus bei Gauso algoritmo kombinacija.

4.6. Sprendimas permutacijų išbandymu

Tobulam žaidimo situacijos sprendimui labai tikėtina jog būtinas pilnas galimų sprendimų perrinkimas [6]. Bandomas algoritmas – visų minų permutacijų išbandymas, atmetant neteisingas [2] [13]. Kiekvienam neatidarytam langeliui išbandomos visos minų permutacijos, kiekvienai jų atliekant validumo patikrinimą. Visos teisingumo patikrinimą atitinkančios permutacijos pridedamos prie sąrašo, iš kurio galima suskaičiuoti minų tikėtinumus kiekviename langelyje.

Net standartiniame 16 aukščio ir 16 pločio žemėlapyje permutacijų kiekis maksimaliai yra 2^{256}, tai yra ženkliai per daug, taigi optimizacijos yra būtinos. Svarbiausia jų – atmesti langelius kurie nesusiję su turimomis užuominomis, taigi į kuriuos nereikia kreipti dėmesio.

Kadangi užuominos duoda informaciją tik apie kaimyninius langelius, galima atmesti visus langelius kurie nėra kaimynai su turimomis užuominomis. Toliau langeliai turintys kaimynines užuominas bus

vadinami kraštiniais langeliais. Kraštinių langelių radimo greitaveikai užtikrinti nepakanka, tyrimas parodė jog net ir 16x16 žemėlapyje dažnai randamos 30+ ilgio kraštinės.

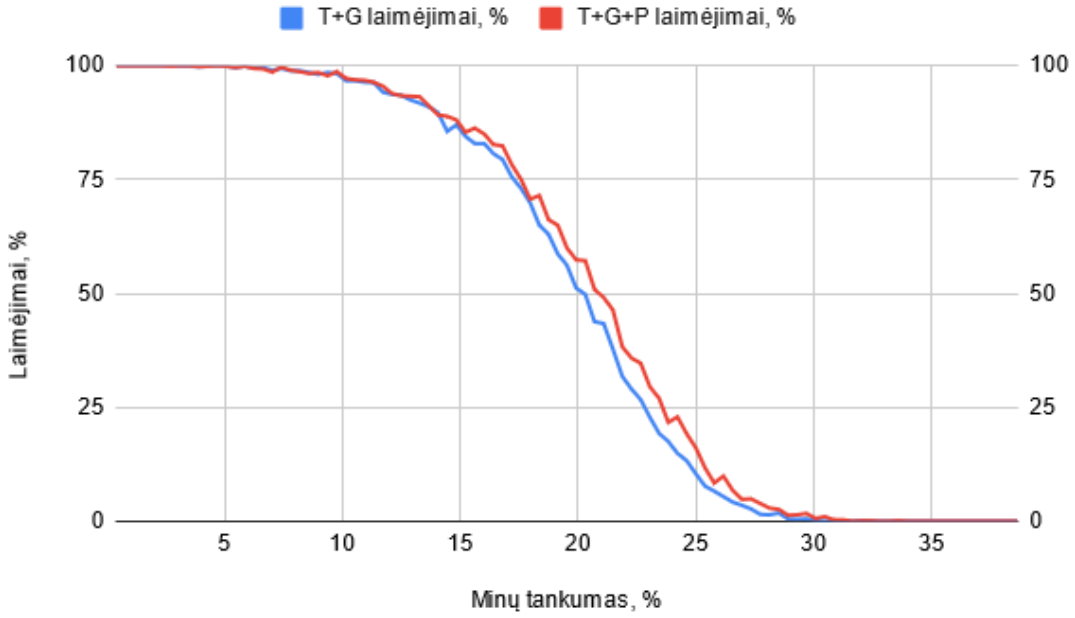
Svarbi optimizacija – identifikuoti bei atskirti visas nesusijusias kraštines, kurias galima spręsti atskirai. Langeliai yra susiję, jei jie ribojasi su ta pačia užuomina, nes ši užuomina turi informaciją apie abu langelius. Nesusijusių kraštinių aptikimui galima naudoti paiešką į plotį, pradedant nuo bet kurio neaplankyto kraštinio langelio. Esant neatidarytame langelyje, keliaujama į kaimynines užuominas, o esant užuominoje keliaujama į neatidarytą langelį.

4.9 pav. Nesusijusių kraštinių aptikimas

Trijų nesusijusių kraštinių aptikimas vaizduojamas **4.9 pav.** Nesusijusių kraštinių aptikimas. Nors matomos keturios neatidarytų langelių salos, dešinėje pusėje esančios salos yra susijusios, nes tarp jų esantys trejetai teikia informaciją apie abi salas, taigi sprendžiant jas reikia vertinti kartu. Šiame pavyzdyje neatlikus nesusijusių kraštinių atskyrimo permutacijų skaičius būtų $2^{7+4+1} = 4096$. Atlikus atskyrimą, permutacijų skaičius yra $2^7 + 2^4 + 2^1 = 146$.

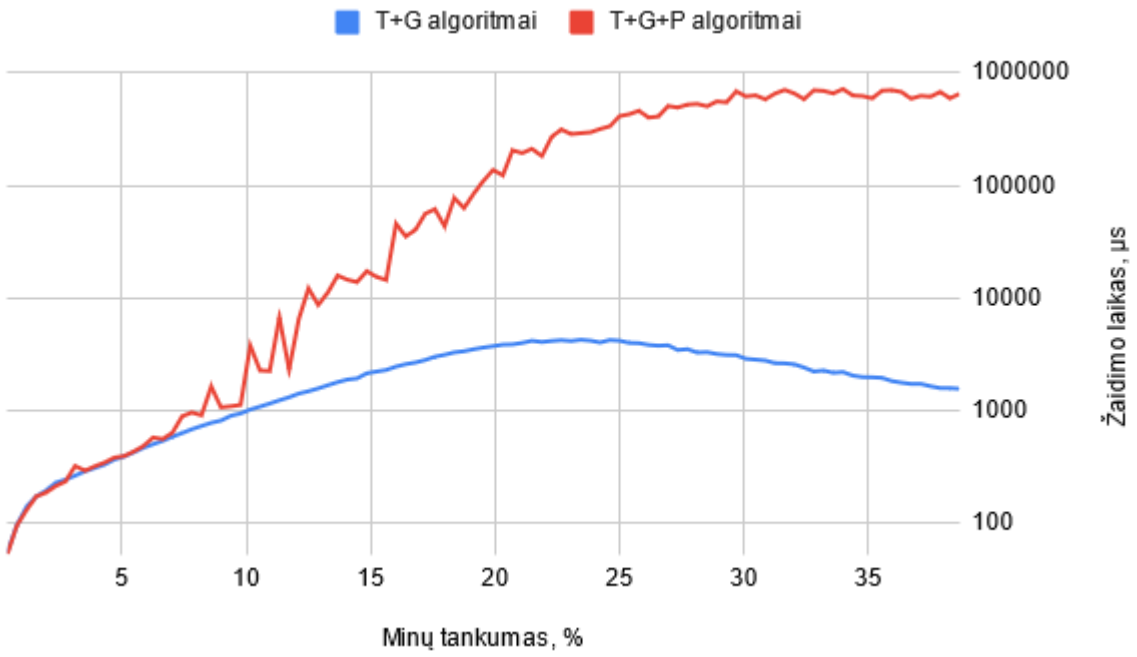
Kadangi po pilno perrinkimo žinomos visos galimos minų permutacijos, tikimybė rasti miną konkrečiame langelyje yra lygi kiekiui permutacijų turinčių miną tame langelyje iš visų teisingų permutacijų. Nežinant nė vieno garantuotai saugaus langelio tai leidžia spėti tikslingai, pasirenkant langelį kuriame mažiausiai tikėtina rasti miną.

4.6.1. Bazinės versijos rezultatai



4.10 pav. Permutacijų išbandymo bazinės versijos tikslumo tyrimo rezultatai

Bazinės versijos tikslumo tyrimo rezultatai vaizduojami **4.10 pav.** Permutacijų išbandymo bazinės versijos tikslumo tyrimo rezultatai. Matoma, jog pridedant net ir bazinę permutacijų išbandymo algoritmo versiją, tikslumas patobulėjo lyginant su ankstesnio tyrimo rezultatais.



4.11 pav. Permutacijų išbandymo bazinės versijos greitaveikos tyrimo rezultatai

Bazinės versijos greitaveikos tyrimo rezultatai vaizduojami **4.11 pav.** Permutacijų išbandymo bazinės versijos greitaveikos tyrimo rezultatai. Grafikas vaizduojamas pasitelkiant logaritminę skalę, norint pavaizduoti sprendimo spartą. Sulėtėjimas yra milžiniškas, lyginant lėčiausiai veikiančius minų tankumus, sulėtėjimas yra apie 170 kartų. Pilna vykdomo testo trukmė – 7 valandos.

4.6.2. Greitaveikos optimizacijos

Kadangi trivialus bei Gauso algoritmai yra greitai veikiantys, greitaveikos optimizacijos sutelktos į permutacijų išbandymo algoritmą.

4.6.2.1. Bendra algoritmo greitaveika

Perrinkimo logika sukurta iš pradžių kurta atsižvelgiant į greitaveiką. Permutacijų tikrinime reikalinga žinoti tikrinamos kraštinės langelių kaimyninius langelius su tam tikrais kriterijais, pavyzdžiui tik atidaryti langeliai teikiantys užuominas. Ši informacija suskaičiuojama vieną kartą iš anksto – sukuriamas papildomas filtruotas kaimynų žemėlapis, naudojamas minų permutacijų tikrinimui. Šio pagalbinio kaimynų žemėlapio užimama vieta operatyviojoje atmintyje mažinama naudojant optimizuotas struktūras bei bitų pakavimą, siekiant pagerinti procesoriaus spartinančiosios atminties (angl. *Cache*) efektyvumą.

Kiekviena minų permutacija kraštinėje yra 64 bitų sveikasis skaičius. Kiekvienas bitas skaičiuje atitinka langelį kraštinėje, 0 – minos nėra, 1 – mina yra. Pavyzdžiui: 0111 dvejetainėje sistemoje atitinka minas pirmuose trijuose kraštinės langeliuose, o ketvirtame langelyje nėra minos. Formuluojant kraštines šiuo būdu, jų iteracija tampa itin efektyvi – vykdomas ciklas nuo 0 iki 2^N kur N yra kraštinės ilgis, šiame cikle pats skaičius ir yra kraštinė, kuri toliau tikrinama pasitelkiant pagalbinį kaimynų žemėlapį. Minų skaičius tikrinamoje kraštinėje gali būti nustatomas nustatant suskaičiuojant įjungtų bitų kiekį skaičiuje (angl. *Hamming weight*). Tai gali būti efektyviai atliekama pasitelkiant bitų skaičiavimo operaciją *popcnt x86* architektūroje bei ją atitinkančias kompiliatorių teikiamas vidines funkcijas (angl. *Compiler intrinsic functions*).

4.6.2.2. Lygiagretinimas procesoriaus gijomis

Pasitelktas lygiagretinimas per daugelį gijų. Kadangi kiekvienos naujos gijos inicializavimas užtrunka fiksuotą laiko dalį, gijos sukuriamos ir paleidžiamos iš anksto, ir laukia užduodamų užduočių (angl. *Thread pooling*). Šiuo atveju užduotys – permutacijų tikrinimas. Dėl sukurtos architektūros kurioje permutacija yra sveikasis skaičius yra lengva paskirstyti darbą tarp skirtingų gijų. Esant N gijų, kiekviena gija atsakinga už $1/N$ permutacijų. Pagrindinė gija užduoda užduotis ir laukia visų gijų darbų pabaigos. Rezultatams saugoti sukurta specifinė paprogramė pasitelkianti beužraktinį (angl. *Lock-free*) kodą, naudojantį atomines operacijas. Toliau, turint patikrintų permutacijų sąrašą, žaidimo sprendimų priėmimas vykdomas įprastai.

4.6.2.3. Lygiagretinimas grafiniame procesoriuje

Siekiant toliau patobulinti greitaveiką, įdiegtas greitinimas pasitelkiant grafinio apdorojimo įrangą naudojant *OpenCL* karkasą. Parašytas *OpenCL* karkaso branduolys (angl. *Kernel*) perkeliantis skaičiavimą į vaizdo apdorojimo procesorių. Karkaso branduolyje vykdomas modifikuotas įprastos validacijos kodas. Branduoliui vykdymo pradžios metu pateikiamas grafinio procesoriaus branduolio identifikavimo numeris, jį pasitelkiant nusprendžiama kurias permutacijas tikrinti – vykdomas užduočių pasidalinimas.

Sprendimas grafiniu procesoriumi skaidomas į fiksuoto dydžio užduotis (angl. *Batches*). Taip daroma, nes vykdant vieną ilgai trunkančią užduotį yra paveikiamas tradicinis grafinis apdorojimas – pilnai sustoja monitoriaus vaizdo atnaujinimas kol vykdoma pateikta užduotis. Taikant užduočių skaidymą, grafinis procesorius gali atvaizduoti vartotojo aplinką tarp permutacijų tikrinimo užduočių

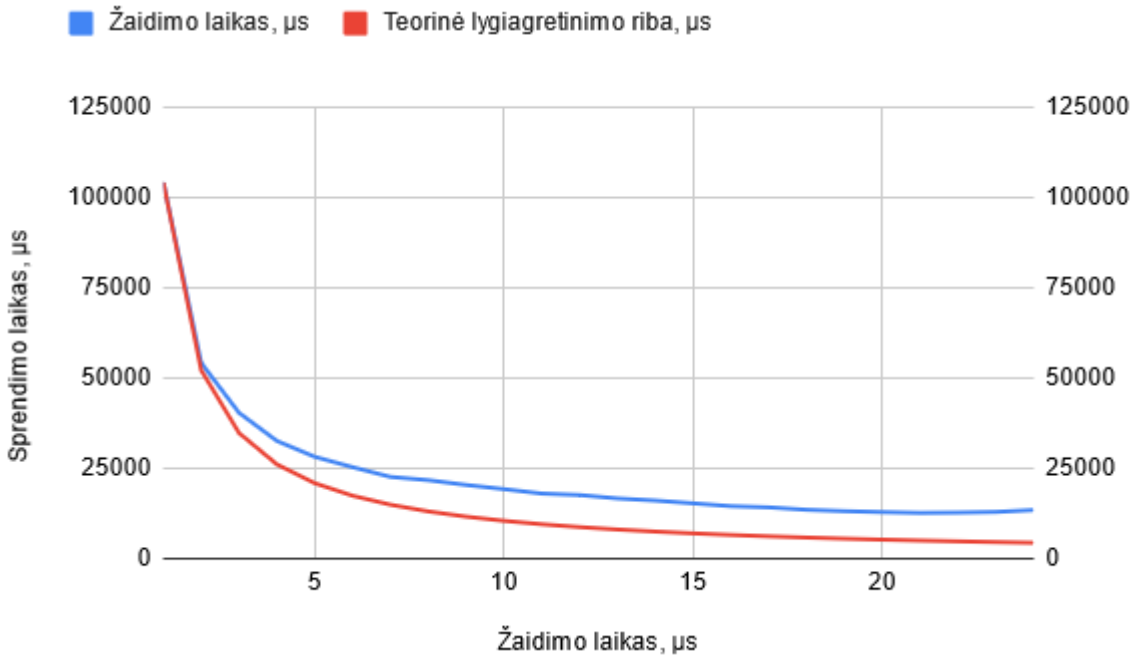
vykdymo. Skaidoma į $2^{20} = 1048576$ permutacijų užduotis. Naudojant šį maksimalų užduoties dydį vykdymo greitis nenukenčia lyginant su greitaveika neskaidant užduočių, o vartotojo sąsaja lieka pilnai interaktyvi.

Dėl pilnai neperprastos klaidos *Intel OpenCL SDK 1.2* tvarkyklėje teko išjungti išankstinį permutacijos tikrinimo sustabdymą aptikus nelogišką užuominą. Paliekant išankstinio ciklo sustabdymo teiginį (angl. *break*), kuris pilnai veikia ne grafiniame procesoriuje, karkaso branduolys elgiasi neteisingai – pažymėdamas regis atsitiktines minų permutacijas kaip teisingas. Jei nebūtų šios tvarkyklės klaidos, sprendimo su grafiniu procesoriumi sparta būtų dar greitesnė.

4.6.2.4. Lygiagretinimo apribojimai

Šiame skyrelyje aprašomi tyrimai vykdyti 16 aukščio bei 16 pločio žemėlapiams su fiksuotu 50 minų skaičiumi, tai atitinka 19,5% minų tankį. Šiems tyrimams atlikta 10 tūkst. žaidimų vietoj 1 tūkst., nes tyrimo su vienu konkrečiu minų tankumu tendencijoms pamatyti reikalingas didesnis žaidimų kiekis.

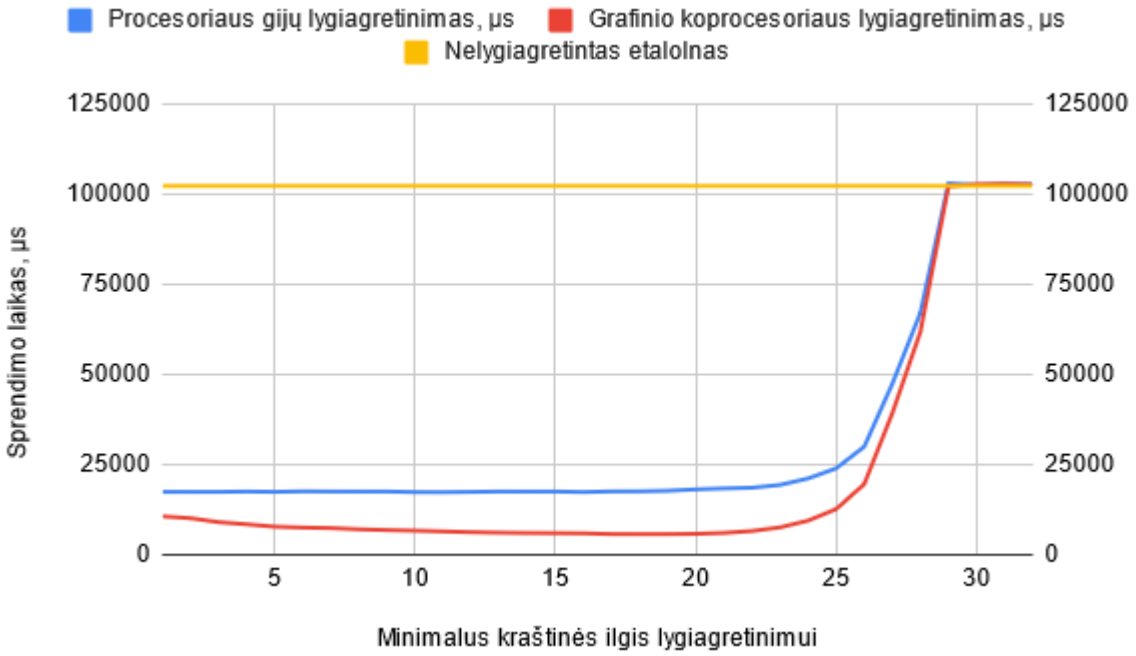
Atliekant du skaičiavimus vienu metu, teoriškai bendra greitaveika turėtų padvigubėti, tačiau praktiškai tai priklauso nuo testavimo aplinkos bei testuojamo algoritmo. Atliktas tyrimas nustatyti gijų kiekio priklausomybę nuo pasiektos greitaveikos. Testuota tik lygiagretinimas pasitelkiant papildomas pagrindinio procesoriaus gijas, nes bent jau *OpenCL 1.2* standartas nepateikia galimybės valdyti naudojamų grafinio procesoriaus gijų kiekį.



4.12 pav. Greitaveikos priklausomybė nuo pasitelktų gijų kiekio

Atlikto tyrimo rezultatai vaizduojami **4.12 pav.** Greitaveikos priklausomybė nuo pasitelktų gijų kiekio. Matoma, jog lygiagretinimo atnešama nauda atsiskiria nuo teorinės ribos, tai rodo jog sukurtas lygiagretinimas yra teoriškai nėra teoriškai idealus, tačiau aiškiai matoma lygiagretinimo nauda. Testuojant su 24 fizines gijas turinčia testavimo aplinka, skaičiavimo gijų skaičiaus didinimas yra naudingas iki apie 20 gijų, tačiau tolimesniuose tyrimuose naudojama 12 gijų, nes tai labiau atitinka labiau paplitusių procesorių galimybes.

Ne visais atvejais gali apsimokėti lygiagretinti permutacijų tikrinimą, Esant mažoms kraštinėms gali būti greičiau patikrinti turimas kraštines nei pasitelkti papildomas gijas dėl papildomo uždavimo pateikimo gijai reikalingo laiko. Atliktas tyrimas nustatyti optimalų kraštinės ilgį nuo kurios apsimoka lygiagretinti darbą. Ištirti abu lygiagretinimo būdai – pasitelkiant papildomas procesoriaus gijas ir pasitelkiant grafinį koprocesorių.



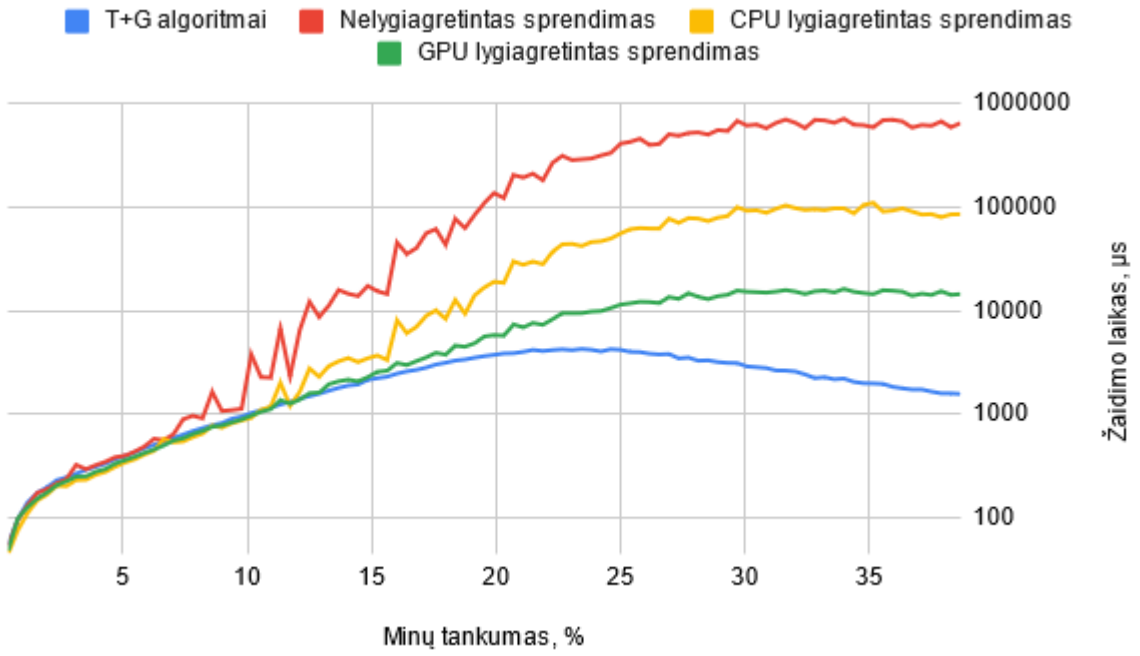
4.13 pav. Lygiagretintų algoritmų spartos priklausomybė nuo minimalios lygiagretinamos kraštinės ilgio

Minimalaus kraštinės ilgio lygiagretinamo tyrimo rezultatai vaizduojami **4.13 pav.** Lygiagretintų algoritmų spartos priklausomybė nuo minimalios lygiagretinamos kraštinės ilgio. Atliktas tyrimas parodė jog su abejomis lygiagretinimo platformomis egzistuoja maksimalus kraštinės ilgis ties kuriuo nelygiagretinant pradedamas prarasti algoritmo efektyvumas. Su testuojama sistema tai yra apie 20, arba $2^{20} = 1048576$ minų permutacijų. Esant trumpesnėms kraštinėms nematoma skirtumo ar lygiagretinama pasitelkiant papildomas procesoriaus gijas. Naudojant skaičiavimų lygiagretinimą grafiniu procesoriumi net matoma greitaveikos regresija kraštinėms trumpesnėms nei 12, dėl ilgos vieno skaičiavimo paleidimo kainos.

Matoma riba ties kraštinės ilgiu 28 nuo kurios rezultatai sutapo, taip yra todėl, kad dėl ir taip ilgos skaičiavimo trukmės teko panaudoti dirbtinę ribą kraštinių ilgiui, virš kurio kraštinės nebus skaičiuojamos, siekiant išlaikyti skaičiavimą realiu laiku. Verta paminėti, jog esant didesniems žemėlapiams maksimalus kraštinių ilgis taip pat didėtų.

Nors lygiagretinimas pasitelkiant grafinį procesorių rodomai pranašesnis už lygiagretinimą pasitelkiant papildomas procesoriaus gijas visais atvejais, dedikuotų grafinių procesorių prieinamumas nėra užtikrintas vartotojo sistemoje. Papildomos procesoriaus gijos buvo prieinamos ir populiarėjo nuo 2005 metų [5] ir egzistuoja praktiškai visuose dabar naudojamuose procesoriuose, taigi lygiagretinimas procesoriuje paliekamas kaip atsarginis planas nerandant grafinio procesoriaus. Siekiant taupyti resursus, tolesniuose tyrimuose pagrindinio procesoriaus gijų lygiagretinimas naudojamas tik nuo kraštinės ilgio 8, o grafinio procesoriaus lygiagretinimas tik nuo kraštinės ilgio 18.

4.6.2.5. Greitaveikos optimizacijų rezultatai



4.14 pav. Greitaveikos optimizacijų tyrimo rezultatai

Atliktas standartinis tyrimas greitaveikos nustatymui, rezultatai vaizduojami **4.14 pav.** Greitaveikos optimizacijų tyrimo rezultatai. Matoma, jog esant visiems minų tankiams greitaveikos optimizacijos ženkliai pagreitino spartą, naudojant lygiagretinimą vaizdo plokštėje greitaveika spartesnė apie 30 kartų. Permutacijų perrinkimo algoritmas vis tiek veikia iki 10 kartų lėčiau esant didesniems minų tankiams, tačiau atsižvelgiant į jo pasiekiamą didesnį tikslumą, tai yra priimtina.

Visų šio algoritmo lygiagretinimo būdų pasiekiamas tikslumas buvo identiškas.

4.6.3. Likusių minų skaičiavimas

Papildoma žaidimo teikiama informacija apart langelių užuominų – bendras likęs minų skaičius. Toliau aprašomi sukurti permutacijų tikrinimo algoritmo patobulinimai, atsižvelgiantys į likusias minas.

4.6.3.1. Filtravimas permutacijų tikrinimo metu

Suskaičiuojami visi nekraštiniai langeliai, apie kuriuos nėra jokios informacijos iš turimų užuominų. Jei imtume, jog nė viename jų nėra minos, tai reikštų jog visos likusios minos yra sprendžiamose kraštinėse. Jei visuose nekraštiniuose langeliuose yra mina, tai sprendžiamose kraštinėse gali būti tik likusios minos atėmus nekraštinių langelių skaičių.

Šis apskaičiavimas duoda viršutinį bei apatinį rėžius minų skaičiui, kurį galima naudoti vykdant permutacijų tikrinimą. Tam tikrais atvejais, ypač žaidimo pabaigoje, kai nekraštinių langelių yra nedaug ar išvis nėra, galima atmesti minų permutacijas kurios nors ir atitinka visas užuominas, bet yra neįmanomos dėl žinomo likusių minų kiekio. Tai leidžia išspręsti papildomas situacija situacijas žaidimo pabaigoje ar bent patikslinti tikimybes rasti minas langeliuose.

4.6.3.2. Minų tikimybių perskaičiavimas atsižvelgiant į likusį minų kiekį

Atskiros kraštinės yra nesusijusios viena su kita per teikiamas užuominas, tačiau gali būti susijusios per likusių minų skaičių. Šiame skyrelyje aprašomas tai išnaudojantis algoritmas, patikslinantis tikimybes rasti minas atsižvelgiant į likusių minų skaičių.

Jau atlikus permutacijų tikrinimą, likusių minų skaičių galima panaudoti dar kartą - iš rastų teisingų permutacijų galima nustatyti vidutinį tikėtiną minų kiekį kraštinėse. Kiekvienos kraštinės kiekvienai patikrintai permutacijai suskaičiuojamas joje esančių minų kiekis, iš to nustatomos minimalus bei maksimalus minų kiekis reikalingas visose kraštinėse. Panašiai kaip aprašyta skyrelyje 4.6.3.1 Filtravimas permutacijų tikrinimo metu, iš naujo apskaičiuojama viršutinė bei apatinė ribos pagal likusių minų skaičių ir neatidarytų langelių skaičių, tačiau šį kartą taip pat atsižvelgiama į apskaičiuotą minimalų bei maksimalų reikalingą minų kiekį kraštinėse, taigi šios ribos tampa tikslesnės ir labiau suvaržančios.

Žiūrint į kiekvieną kraštinę, jei minų kiekis visose teisingose kraštinės permutacijose yra vienodas, tai parodo jog ši kraštinė yra nepriklausoma nuo likusių minų kiekio, taigi toliau nebus svarstoma. Visos kraštinės, kuriose rastas kintantis minų kiekis teisingose permutacijose yra patikrinamos dar kartą, šį kartą bendrai kartu, traktuojant jas kaip vieną bendrą kraštinę. Naujos minų permutacijos negeneruojamos, iš jau patikrintų individualių kraštinių minų permutacijų sugeneruojamos permutacijų kombinacijos, kurios paverčiamos į apjungtos bendros kraštinės minų permutacijas. Šios bendros permutacijos yra tikrinamos pagal suskaičiuotus rėžius. Į žemėlapio užuominas nekreipiama dėmesio, nes jau žinoma kad šios permutacijos teisingos pagal žemėlapio užuominas iš pirminio tikrinimo, tikrinamas tik bendras minų skaičius.

Perskaičiavus kraštinių permutacijas, iš naujo apskaičiuojama tikimybė rasti minas kiekvienos kraštinės langeliuose. Šios tikimybės yra tikslesnės, nes pilnai atsižvelgiama į likusių minų skaičių.

Kelių kraštinių tikrinimas kartu nebūtų įmanomas be pradinio individualių kraštinių tikrinimo, kurio metu atmetama praktiškai visos minų permutacijos. Tačiau vis vien – jaučiamas sulėtėjimas. Šis papildomas tikrinimas yra lygiagretinamas pasitelkiant papildomas procesoriaus gijas siekiant sumažinti sulėtėjimą ir išlaikyti sparčią greitaveiką.

4.6.3.3. Minų tikimybių skaičiavimas ne kraštinėse

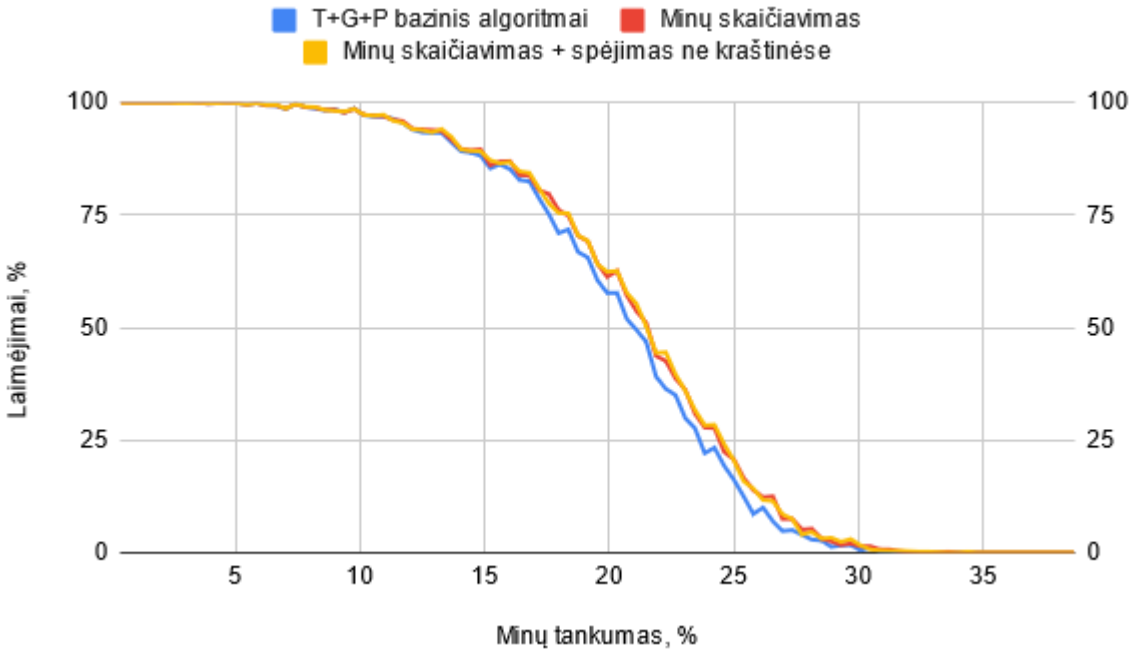
Pertikrinant minų permutacijas visose kraštinėse sukuriama papildomą nauda – gali būti lengvai sužinomas tikėtiniausias vidutinis minų kiekis visose kraštinėse. Nors minų gali egzistuoti tik sveikasis skaičius, šis tikėtiniausias vidutinis minų kiekis nėra sveikasis skaičius, nes yra skaičiuojamos tikimybės.

Žinant tikėtiniausią minų kiekį kraštinėse įmanoma suskaičiuoti tikėtiniausią minų kiekį ne kraštinėse. Kadangi yra pateikiamas likęs minų skaičius, tikėtiniausias minų kiekis ne kraštinėje yra likusių minų skaičius atėmus tikėtiniausią minų skaičių kraštinėje. Žinant šį skaičių, tikimybė rasti miną bet kuriame ne kraštinės langelyje lengvai suskaičiuojama padalinant tikėtiniausią ne kraštinės minų skaičių iš ne kraštinės langelių kiekio.

Žinant tikimybę pataikyti ant ne minos įmanoma nustatyti ar verta iš viso spėti kraštinėje, ar galbūt labiau apsimoka spėti aklai bet kur ne kraštinėje. Vadovaujamasi įprasta logika spėjimui renkantis langelį su mažiausia tikimybe rasti miną.

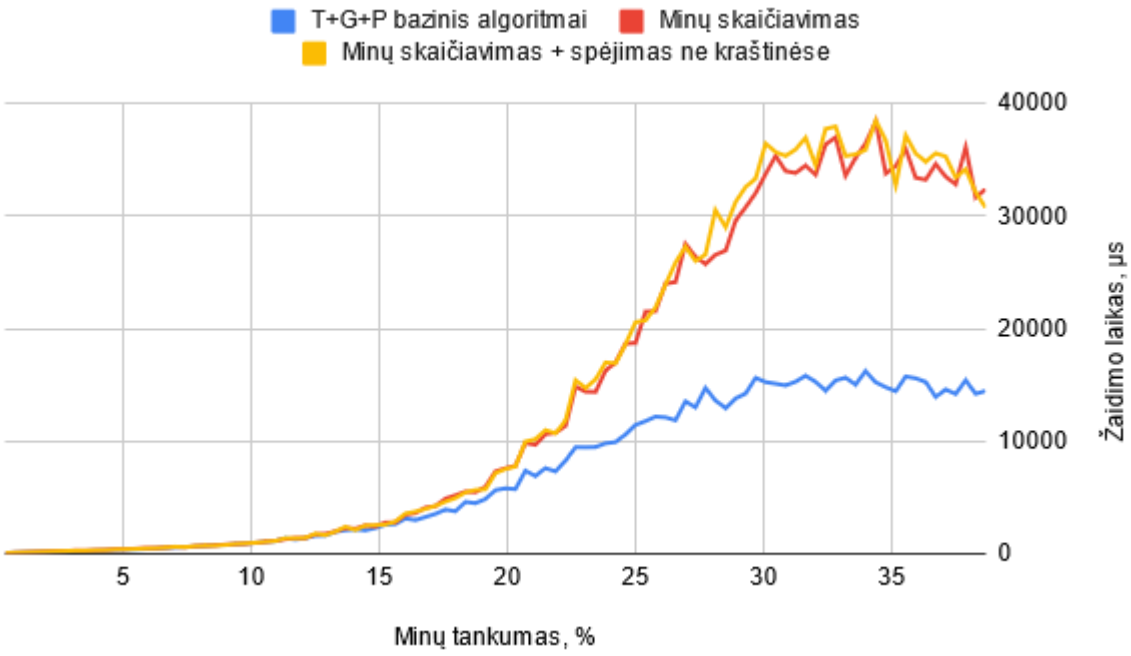
4.6.3.4. Likusių minų skaičiavimo rezultatai

Atlikti standartiniai tyrimai likusių minų skaičiavimo patobulinimams įvertinti.



4.15 pav. Likusių minų skaičiavimo tikslumo tyrimo rezultatai

Likusių minų skaičiavimo tikslumo tyrimo rezultatai vaizduojami **4.15 pav.** Likusių minų skaičiavimo tikslumo tyrimo rezultatai. Matoma, jog minų tikimybių perskaičiavimas pagal minų skaičių duoda akivaizdų tikslumo padidėjimą, tačiau ne kraštinių tikimybių paskaičiavimas yra tik labai nežymiai pranašesnis paėmus bendrai, ir kai kuriais atvejais net veikia prasčiau. Tačiau dėl bendrai paėmus kad ir nežymiai didesnio tikslumo, ši optimizacija laikoma naudinga.



4.16 pav. Likusių minų skaičiavimo greitaveikos tyrimo rezultatai

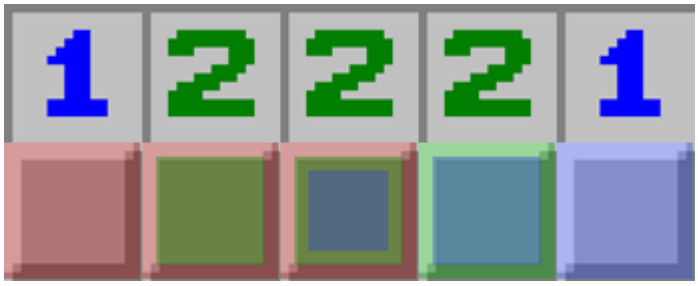
Likusių minų skaičiavimo greitaveikos rezultatai vaizduojami **4.16 pav.** Likusių minų skaičiavimo greitaveikos tyrimo rezultatai. Matoma, jog šie patobulinimai sulėtina greitaveiką apie du su puse karto esant dideliam minų tankumui. Šis sulėtėjimas nėra kritiškas, ir atsižvelgiant į jų suteikiamą tikslumo naudą, tai yra priimtina.

4.6.4. Dalinių kraštinių tikrinimas

Iki šiol taikytas būdas susidoroti su itin ilgomis kraštinėmis – jas ignoruoti ir nevertinti minų permutacijų joms, prireikus – spėti aklai. Sukurtas patobulinimas yra algoritmas dalinai įvertinantis kraštinę sukuriant aproksimaciją tikroms tikimybėms rasti miną kraštinės langelyje.

4.6.4.1. Dalinio kraštinių tikrinimo aprašymas

Aptikus kraštinę kurios ilgis N viršija maksimalią toleruotiną ribą, ši kraštinė išskaidoma į mažesnes kraštines, turinčias fiksuotą ilgį M. Tai taikant, tradicinis kraštinės minų permutacijų kiekis 2^N tampa $(N - M + 1) \cdot 2^M$. Tai sustabdo eksponentinį sudėtingumo augimą, nes eksponentė M yra fiksuoto dydžio skaičius.



4.17 pav. Dalinių kraštinių paskirstymo pavyzdys

Pavyzdys vaizduojamas **4.17 pav.** Dalinių kraštinių paskirstymo pavyzdys. Šiuo atveju 5 langelių ilgio kraštinė skaidoma į tris 3 langelių ilgio kraštines. Vietoj $2^5 = 32$ permutacijų bus tikrinama tik $(5 - 3 + 1) \cdot 2^3 = 3 \cdot 8 = 24$ permutacijos.

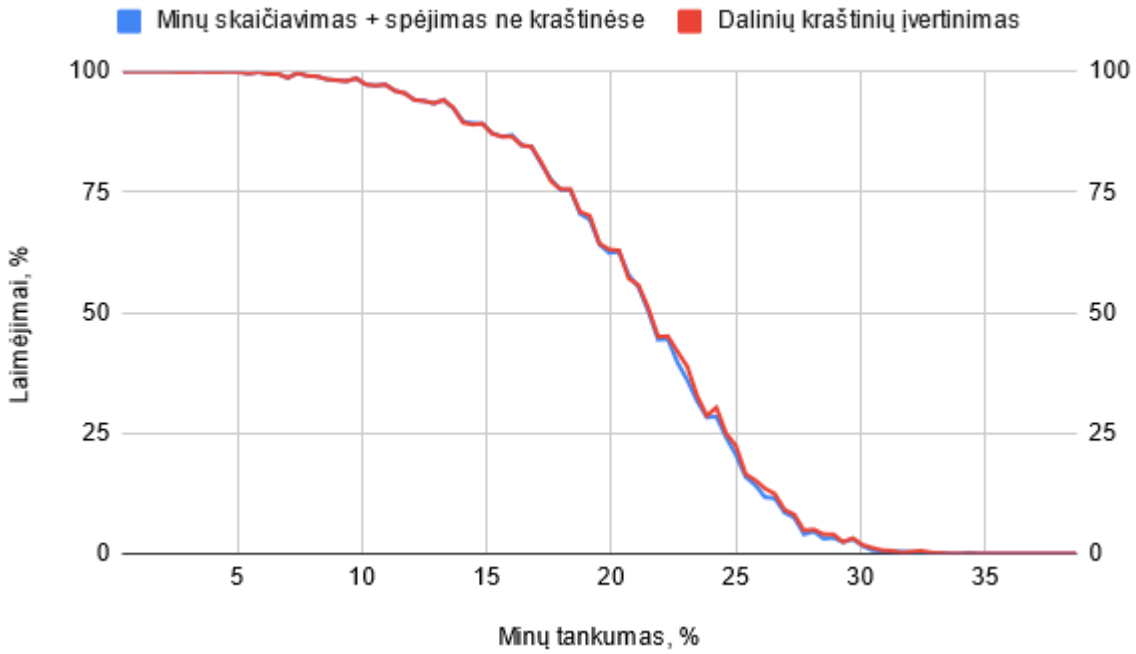
Jei patikrinus visas dalinės kraštinės permutacijas nerandama nė viena permutacija kurios viename ar daugiau langelių nėra minos, ne taip kaip tikrinant įprastas kraštines, nėra garantuota jog šis langelis neturi minos. To negalima garantuoti, nes įmanoma jog minos gali nebūti šalia esančiame langelyje, kuris nepriklauso dalinei kraštinei tačiau yra kaimyninis su dalinės kraštinės permutacijų tikrinimui naudojama užuomina. Taip pat jei nerasta nė viena permutacija kurioje viename ar keliuose langeliuose nebūtų minos, tai negarantuoja jog šiuose langeliuose garantuotai minos.

Garantuotam minų ar saugių langelių radimui naudojamas prieštaringumo atmetimas. Kadangi maksimaliai langelis turi tik 8 kaimyninius langelius, naudojant dalinės kraštinės ilgį didesnį nei 8, yra garantuota jog imant kiekvieną langelį kaip dalinės kraštinės centrą, jei visų dalinių kraštinių permutacijos sutinka šiam langeliui – galima garantuoti minos buvimą ar nebuvimą šiame langelyje, kadangi yra neįmanoma jog naudojama užuomina galėtų paveikti ne dalinėje kraštinėje esantį langelį. Anksčiau tikrintų dalinių kraštinių sprendimai nepamirštami pereinant prie tolimesnių dalinių kraštinių – garantuotam minos buvimo ar nebuvimo nusprendimui įvertinami visų dalinių kraštinių rezultatai, ir verdiktas priimamas tik jei jie visi sutampa. Svarbu paminėti jog tai negarantuoja jog bus rasti visi langeliai garantuotai turintys ar neturintys minų, tai būtų NP-pilnas uždavinys[6] ir

atitiktų dalinių kraštinių nenaudojimą. Užtikrinama tik kad jei algoritmas rado miną turintį ar neturintį langelį, tai yra garantuotai tiesa.

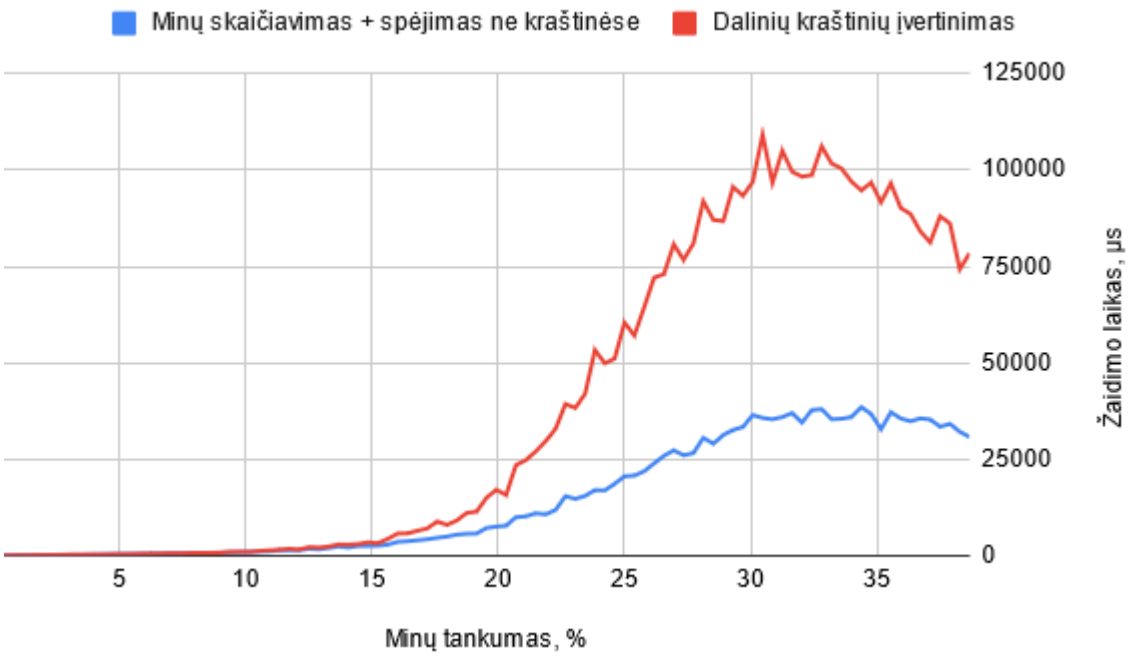
Neužtikrintiems kraštinės langeliams, tai yra langeliams kuriems randama teisingų permutacijų ir su minomis ir be minų, minos radimo tikimybės suskaičiavimas yra tik aproksimacija. Kiekvienai dalinei kraštinei apskaičiuojama tikimybė rasti miną, ir visų skaičiuotų dalinių kraštinių tikimybėms yra suskaičiuojamas vidurkis.

4.6.4.2. Dalinių kraštinių tikrinimo rezultatai



4.18 pav. Dalinių kraštinių tikrinimo tikslumo rezultatai

Dalinių kraštinių tikrinimo tikslumo rezultatai rodomi 4.18 pav. Dalinių kraštinių tikrinimo tikslumo rezultatai. Matoma, jog yra pasiekiamas nežymiai didesnis tikslumas. Kadangi tikslumo padidėjimas toks mažas, tai rodo jog pernelyg ilgų kraštinių sprendimas nėra itin naudingas, tačiau tai vis vien yra tiksliausia išbandyta algoritmų kombinacija.



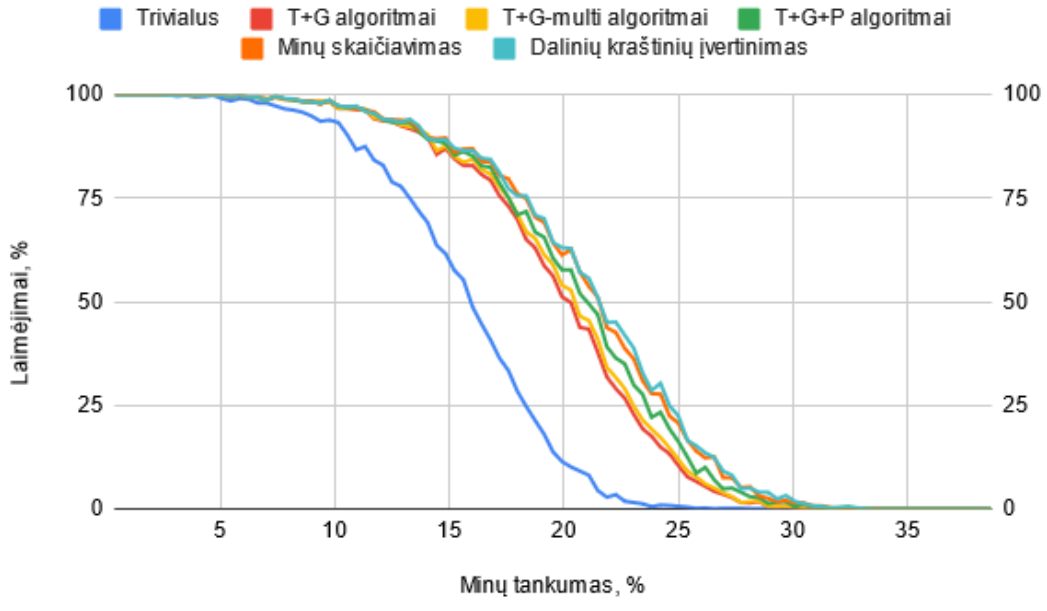
4.19 pav. Dalinių kraštinių tikrinimo greitaveikos rezultatai

Dalinių kraštinių tikrinimo tikslumo rezultatai rodomi **4.19 pav.** Dalinių kraštinių tikrinimo greitaveikos rezultatai. Matoma, jog poveikis greitaveikai ženklus, sprendimas apie 3 kartus lėtesnis. Tai ganėtinai ženklus sulėtėjimas, galintis paveikti vartotojo pasitenkinimą sukurta sistema. Standartiniuose nustatymuose dalinių kraštinių tikrinimas paliekamas įjungtas, su galimybe išjungti siekiant geresnės greitaveikos.

4.7. Tyrimo apibendrinimas

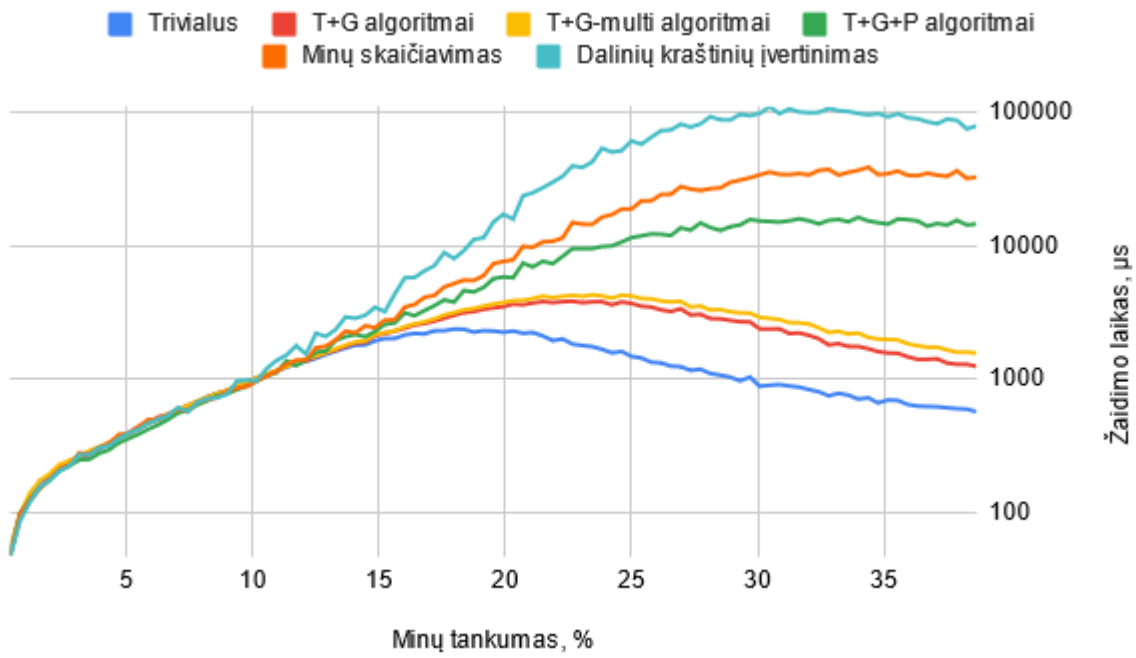
Šiame skyriuje pateikiami visų ankstesnių tyrimų apibendrinti tyrimai, siūloma optimali algoritmų kombinacija, bei standartinių Minesweeper sunkumų tyrimai.

4.7.1. Algoritmų palyginimas



4.20 pav. Algoritmų tikslumo palyginimas

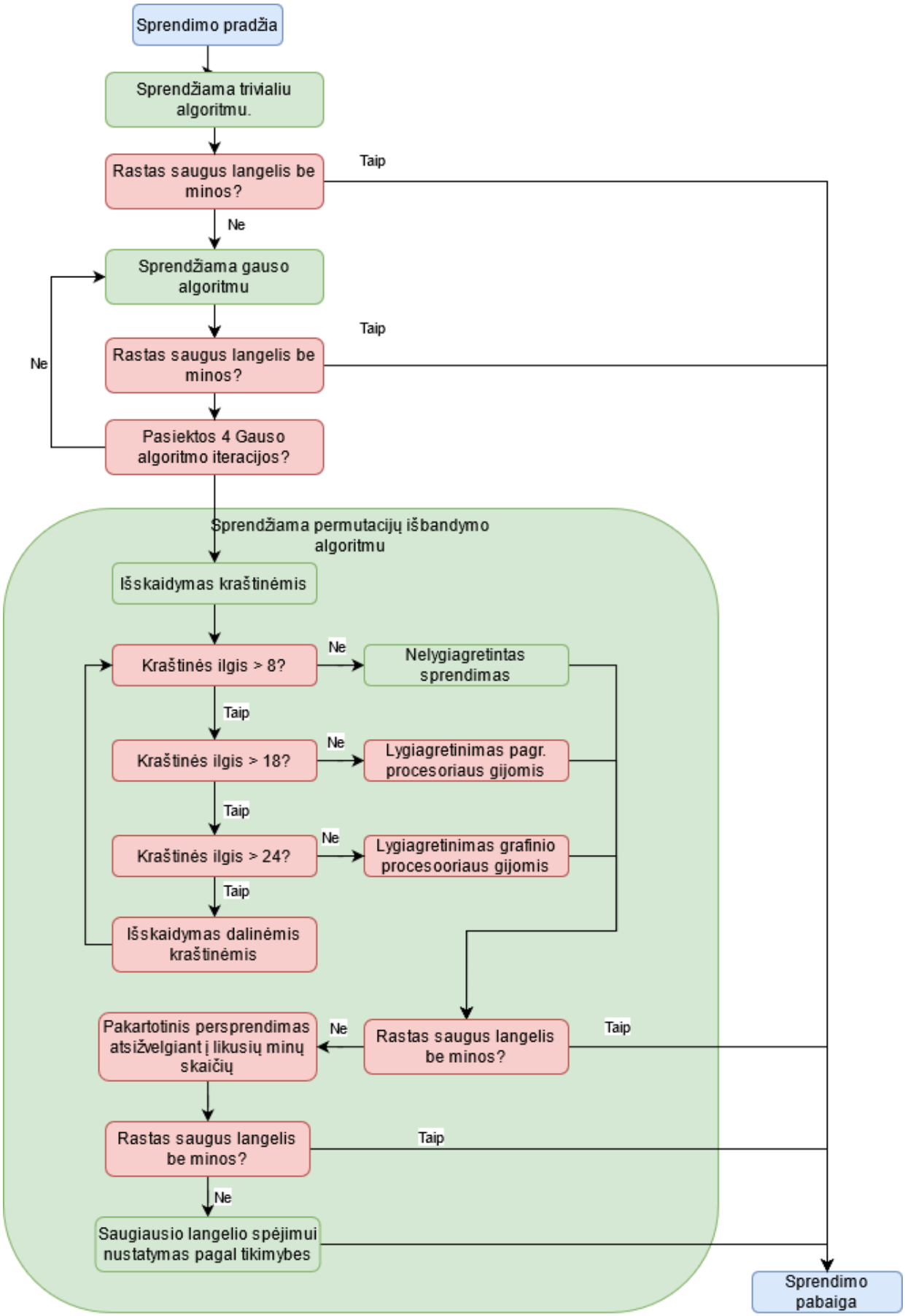
Bendras algoritmų tikslumo palyginimas vaizduojamas **4.20 pav.** Algoritmų tikslumo palyginimas. Algoritmai naudoti kartu, panaudojant tolimesnį patobulinimą po ankstesnių. Matoma, jog panaudojus Gauso algoritmą pasiektas didžiulis tikslumo šuolis, o po to einantys algoritmai tikslumą patobulina ne tiek drastiškai, tačiau juos naudojant vis tiek matomas ženklus tikslumo patobulėjimas.



4.21 pav. Algoritmų greitaveikos palyginimas

Bendras algoritmų greitaveikos palyginimas vaizduojamas **4.21 pav.** Algoritmų greitaveikos palyginimas. Čia vaizduojamos sparčiausios algoritmų versijos, su aprašytais greitaveikos patobulinimais. Grafikas vaizduojamas logaritminėje skalėje. Matoma, jog esant labai mažam minų tankumui, visos algoritmų kombinacijos veikia vienodai greitai, nes trivialus algoritmas dominuoja ir sugeba visas situacijas išspręsti itin greitai. Didėjant minų tankumui, algoritmų greitaveika išsiskiria, kuo daugiau papildomų algoritmų naudojama, tuo lėtesnė greitaveika. Didžiausias reliatyvus šuolis – naudojant permutacijų išbandymo algoritmą. Net ir su visais šiais algoritmais, žaidimo sprendimas turėtų būti pakankamai spartus. Vartotojams paliekama galimybė išjungti norimus algoritmus, siekiant paspartinti greitaveiką toliau.

4.7.2. Optimali algoritmų panaudojimo seka



4.22 pav. Bendras optimalus algoritmų panaudojimas

Bendras sukurtas algoritmų panaudojimas vaizduojamas rodomas **4.22 pav.** Bendras optimalus algoritmų panaudojimas. Dėl greitaveikos iš pradžių sprendžiama trivialiu algoritmu, nepavykus – vykdomos keturios Gauso algoritmo iteracijos. Nepavykus išspręsti Gauso algoritmu, naudojamas permutacijų išbandymo algoritmas. Išskaidoma kraštinėmis, kurios individualiai sprendžiamos pagal kraštinės ilgį nustatomu tinkamiausiu lygiagretinimo metodu, arba išskaidomos dalinėmis kraštinėmis esant per ilgai kraštinei. Nepavykus išspręsti, kraštinės persprendžiamos bendrai, atsižvelgiant į likusių minų skaičių. Jei nė vienam ankstesniam algoritmui nepavyko nustatyti garantuotai saugaus langelio, vykdomas spėjimas langelyje, kuriame mažiausiai tikėtina rasti miną.

Spalvomis nurodoma:

- Mėlyna spalva: pradžios bei išeities taškai.
- Žalia spalva: anksčiau sukurti ir tyrimo metu tik išbandyti algoritmai.
- Raudona spalva: tyrimo metu sukurti bei išbandyti algoritmai ar jų patobulinimai.

4.7.3. Sprendimo standartiniais sunkumais rezultatai

4.3 lentelė. Galutiniai standartinių Minesweeper sunkumų tyrimai.

Pavadinimas	**Plotis**	**Aukštis**	**Plotas**	**Minų kiekis**	**Minų tankumas**	**Sėkmingas žaidimo išsprendimas. %.**	**Vid. žaidimo sprendimo laikas , µs .**
Pradedantysis (angl. *Beginner*)	9	9	81	10	12.3%	96.70%	449
Vidutinis (angl. *Intermediate*)	16	16	256	40	15.6%	87.8%	4102
Ekspertas (angl. *Expert*)	30	16	480	99	20.6%	52.9%	33721

Galutiniai standartinių Minesweeper sunkumų tyrimai vaizduojami **4.3 lentelė.** Galutiniai standartinių Minesweeper sunkumų tyrimai. Matoma, jog net sprendžiant eksprerto sunkumu, žaidimas dažniau išsprendžiamas nei neišsprendžiamas.

5. Išvados

1. Atlikus rinkoje esamų automatinio Minesweeper žaidimo algoritmų analizę, naudojimui pasirinktas trivialus, Gauso, bei permutacijų išbandymo algoritmai.
2. Pasirinktiems algoritmams sukurtas į greitaveiką atsižvelgiantis kodas, sukurti nauji tikslumo bei greitaveikos patobulinimai. Šių patobulinimų dėka, vartotojams itin greitai suteikiami tikslūs žaidimo keliamų užduočių atsakymai.
3. Sukurtas PĮ prototipas leidžiantis vartotojams ir žaisti Minesweeper žaidimą, ir priėjus neaiškią situaciją pasitelkti automatinio sprendimo funkcionalumą.
4. Išbandyti trys baziniai algoritmai – trivialus, Gauso metodo, bei permutacijų išbandymo. Visi šie algoritmai buvo panaudotini siekiant tikslumo ar greitaveikos. Bendrai greičiausiai veikiantis trivialus algoritmas yra mažiausiai tikslus, Gauso algoritmo tikslumas bei greitaveika vidutiniai, o tiksliausiai veikiantis permutacijų išbandymo algoritmas veikia itin lėtai.

   Sukurti šie egzistuojančių algoritmų patobulinimai:

   - Kiekvienoje algoritmų vykdymo žingsnyje patikrinama ar rastas saugus langelis atvėrimui, jį radus galima stabdyti vykdomą iteraciją nes žinoma jog atidarius langelį bus gaunama papildomos informacijos, tai sutaupo nereikalingų skaičiavimų palaukiant papildomos informacijos iš tolimesnių užuominų.
   - Vykdomos kelios Gauso algoritmo iteracijos padidina tikslumą tačiau greitaveikos sulėtėjimas yra neženklus. Esant lėtesniam algoritmui taikomam po Gauso algoritmo, kiekvienas tikslumo pagerinimas yra naudingas sumažinant tolimesnio skaičiavimais intensyvaus algoritmo panaudojimą.
   - Minų permutacijų algoritmo vykdymui sukurtas lygiagretinimas pasitelkiant papildomas procesoriaus gijas bei grafinį procesorių, tai pagreitino algoritmų vykdymą apie 30 kartų.
   - Išskaidymas dalinėmis kraštinėmis yra aproksimacija naudojama kaip alternatyva išankstiniam skaičiavimo nutraukimui, ir pasiekia šiek tiek geresnius rezultatus nei bazinis algoritmas, su trigubai ilgesnio skaičiavimo kaina.
   - Atsižvelgimo į likusias minas patobulinimas ženkliai pagerina pasiekiamą algoritmo tikslumą, tačiau tuo pačiu sulėtina greitaveiką apie du su puse karto.
   - Ne kraštinių įvertinimas yra neženkliai tikslesnis nei tiesiog likusių minų skaičiavimo patobulinimas, tačiau nesulėtina skaičiavimo greitaveikos.
5. Nustatytas optimalus šių algoritmų kombinacijos panaudojimas, suteikiantis ir aukštą tikslumą ir sąlyginai spartų veikimą.

Literatūros sąrašas

1. ADAMATZKY, A. How cellular automaton plays Minesweeper. In *Applied Mathematics and Computation* . 1997. .
2. BECERRA, D.J. Algorithmic Approaches to Playing Minesweeper. In [interaktyvus]. 2015. [žiūrėta 2019-12-16]. . Prieiga per internetą: http://nrs.harvard.edu/urn-3:HUL.InstRepos:14398552.
3. CASTILLO, L.P. - WROBEL, S. Learning minesweeper with multirelational learning. In *IJCAI International Joint Conference on Artificial Intelligence* . 2003. .
4. COOK, S.A. The complexity of theorem-proving procedures. In *Proceedings of the Annual ACM Symposium on Theory of Computing* . 1971. .
5. GEER, D. Industry trends: Chip makers turn to multicore processors. In *Computer* . 2005. Vol. 38, no. 5, p. 11–13. .
6. KAYE, R. Minesweeper is NP-complete. In *The Mathematical Intelligencer* . 2000. .
7. MASSAIOLI, R. Solving Minesweeper with Matrices. In [interaktyvus]. [žiūrėta 2021-05-17]. Prieiga per internetą: https://massaioli.wordpress.com/2013/01/12/solving-minesweeper-with-matricies/comment-page-1/.
8. NAKOV, P. - WEI, Z. Minesweeper,# Minesweeper. In *Unpublished Manuscript, Available at: http://www.minesweeper.info/articles/Minesweeper (Nakov, Wei).pdf* . 2003. .
9. ROBERT, M. Solving Minesweeper with Matrices. In [interaktyvus]. 2013. [žiūrėta 2019-12-16]. Prieiga per internetą: https://massaioli.wordpress.com/2013/01/12/solving-minesweeper-with-matricies/.
10. SCOTT, A. - STEGE, U. - ROOIJ, I. VAN Minesweeper May Not Be NP-Complete but Is Hard Nonetheless. In *Mathematical Intelligencer* . 2011. .
11. SEBAG, M. - TEYTAUD, O. Combining Myopic Optimization and Tree Search: Application to MineSweeper. In *LION6, Learning and Intelligent Optimization* . 2012. .
12. STUDHOLME, C. Minesweeper as a constraint satisfaction problem. In . 2000. .
13. YURICHEV, D. 2021. .
14. C# .NET vs C++ g++ - Which programs are fastest? | Computer Language Benchmarks Game. In [interaktyvus]. [žiūrėta 2021-05-24]. Prieiga per internetą: https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/csharpcore-gpp.html.
15. The most successful game ever: a history of Minesweeper | TechRadar. In [interaktyvus]. [žiūrėta 2021-05-24]. Prieiga per internetą: https://www.techradar.com/news/gaming/the-most-successful-game-ever-a-history-of-minesweeper-596504.

Priedai

1 Sukurto prototipo licencija

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

  Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

  Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such

measures.

  When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no

further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in

source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same

material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the

rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to

sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is

in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single

combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a

copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
{one line to give the program's name and a brief idea of what it does.}
Copyright (C) {year}  {name of author}

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

    {project} Copyright (C) {year} {fullname}
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.