KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Žilvinas Jucius

EURISTINIŲ METODŲ TAIKYMAS “MAHJONG” ŽAIDIMUI

Baigiamasis magistro projektas

Vadovas
Prof. dr. Alfonsas Misevičius

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

EURISTINIŲ METODŲ TAIKYMAS “MAHJONG” ŽAIDIMUI

Baigiamasis magistro projektas
Informatika (621I10003)

Vadovas
(parašas) Prof. dr. Alfonsas Misevičius
(data)

Recenzentas
(parašas) Prof. dr. Vacius Jusas
(data)

Projektą atliko
(parašas) Žilvinas Jucius
(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Žilvinas Jucius

(Studento vardas, pavardė)

Informatika, 621I10003

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Euristinių metodų taikymas „Mahjong“ žaidimui“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Žilvino Juciaus**, baigiamasis projektas tema „Euristinių metodų taikymas „Mahjong“ žaidimui“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Jucius, Žilvinas. Euristinių metodų taikymas „Mahjong“ žaidimui. Magistro baigiamasis projektas / vadovas prof. dr. Alfonsas Misevičius; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Informatika, Dirbtinis intelektas

Reikšminiai žodžiai: *„Mahjong“ žaidimas, euristiniai algoritmai, dirbiniai neuroniniai tinklai*.

Kaunas, 2017. 63 p.

SANTRAUKA

Žaidimai, kurie turi kontroliuojamą aplinką su griežtomis taisyklėmis ir nesudėtingai apskaičiuojamą žaidimo kokybę, kaip pvz. šachmatai ar šaškės, suteikia idealią galimybę juos panaudoti sistemų mokymosi ir metodams išbandyti. Kadangi šiuos žaidimus dažniausiai gali žaisti keli žaidėjai, kompiuteriui varžantis su žmogumi, galima įvertinti šio algoritmo efektyvumą. Kompiuteriai jau seniai yra pripažinti esantys lygūs arba pranokstantys pasaulinio lygio žaidėjus, žaidžiant įvairius žaidimus, kaip šachmatai, nardai ir nesenai pranoko vienu iš sudėtingiausių laikomo žaidimo „Go“ pasaulio čempioną.

Šiame darbe yra nagrinėjamas euristinių metodų naudojimas žaidimui „Mahjong.“ Tarp nagrinėjamų metodų yra euristiniai metodai, dirbtiniai neuroniniai tinklai, suvaržymų patenkinimo uždavinių sprendimo metodai. Šių metodų efektyvumas palygintas tarpusavyje ir iš eksperimentinių rezultatų padarytos išvados.

Jucius, Žilvinas. *Application of Heuristic Methods to Mahjong Game,* Master‘s thesis in Informatics / supervisor Prof. Dr. Alfonsas Misevičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Informatics, Artificial intelligence

Key words: *Mahjong game, heuristic algorithms, artificial neural networks*

Kaunas, 2017. 63 p.

SUMMARY

Games with controlled environment, fixed rules and easy to compute performance, like chess or checkers, allow ideal conditions for testing Machine learning algorithms and techniques. As most games allow multiple players, competing computer versus human player makes it easy to assess the effectiveness of the algorithms. Computers have long equaled or surpassed world class human players in various games ranging from chess to backgammon, and recently it has surpassed Go, a game thought to be one of most complex, world champion.

This work analyses application of heuristic methods to Mahjong game. It is accomplished using heuristic methods, artificial neural networks, constraint satisfaction problem solving methods. The effectiveness of these methods is compared and conclusions are drawn.

Turinys

1. ĮVADAS

„Mahjong“ yra keturių žaidėjų žaidimas, kurį dažnai laisvalaikiu žaidžia Azijoje, ypač Kinijoje ir Japonijoje. Laikui bėgant išsivystė daugybė taisyklių variantų, kurios skiriasi priklausomai nuo šalies ar net regiono. Šiame darbe naudojamos Europos „Mahjong“ asociacijos „Riichi mahjong“ taisyklės [1].

Dauguma sistemų mokymosi metodai yra taikomi dviejų žaidėjų žaidimams, kuriuose žaidėjai keičiasi vienas po kito. Dauguma populiarių žaidimų atitinka dviejų žaidėjų kriterijų, nebent išskyrus pokerį. Tuo tarpu „Mahjong“ žaidime, kurį jau ir taip žaidžia keturi žaidėjai, yra įmanoma žaidėjo ėjimui būti praleistam, kai kiti žaidėjai „pavagia“ išmestas kaladėles. Šios galimos situacijos padaro žaidimą sunkiau modeliuojamą. Taip pat faktas, kad žaidimo „rankos“ vertė kinta tarp žaidimų ar net to pačio žaidimo metu, pasunkina paprasto žaidimo „rankos“ vertės įvertinimo metodo sukūrimą. Tai ypač sudėtinga žaidimo „rankoje“ tik su eilėmis, kuri turi daug apribojimų, kad būtų įmanoma su šia „ranka“ laimėti. Pavyzdžiui, vien tik eilių „ranka“ gali laimėti iš kito žaidėjo išmestos kaladėlės tik tada, kai ši „ranka“ gali laimėti dviem skirtingomis kaladėlėmis. Dėl „rankos“ vertės kintamumo yra sunkiau nustatyti, ar „ranka“ gali laimėti, jos tikrąją vertę.

Taip pat reikia paminėti, kad naujų kaladėlių traukimai yra iš sumaišytų kaladėlių „sienų“, todėl sėkmė turi didelę įtaką žaidimo eigai. Reikia sužaisti didelį kiekį žaidimų, kad būtų įmanoma išsiaiškinti tikslų algoritmo efektyvumo rezultatą. Pavyzdžiui, ar bandyti laimėti naudojant greitai sudaromą mažai vertą „ranką“, ar didesnės vertės lėčiau sudaromą ranką? Maksimalios vertės „ranka“ yra 48 kartus vertingesnė nei mažiausios vertės „ranka“, ir todėl laimingas kaladėlės traukimas gali iškraipyti rezultatus.

2. DARBO TIKSLAS

Šio darbo tikslas yra ištirti euristinius sprendimo metodus „Mahjong“ žaidimui. Siekiama konkrečiai išanalizuoti kompiuteriniu-eksperimentiniu būdu pasiūlytų euristinių metodų efektyvumą žaidžiant „Mahjong“ žaidimą. Efektyvumas šiame darbe įvertinamas apskaičiuojant surinktą žaidėjų taškų kiekį it kitus papildomus rodiklius.

Darbo tikslui pasiekti buvo nagrinėjami „Mahjong“ žaidimui pritaikyti euristinio pobūdžio algoritmai:

- Godus atsitiktinis algoritmas
- Godus euristinis algoritmas
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi
- Giliu Q neuroniniu tinklu pagrįstas algoritmas
- Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstas algoritmas
- Brutalios jėgos metodu pagrįstas algoritmas

3. „MAHJONG“ ŽAIDIMAS

3.1. „Riichi Mahjong“ taisyklės

„Riichi Mahjong“ (toliau – „Mahjong“) yra stalo žaidimas, kurį keturi žaidėjai, pradedami su 25000 taškų, konkuruoja surinkti kuo daugiau taškų. Vieną „Mahjong“ partiją dažniausiai sudaro vienas arba du raundai. Vieną raundą dažniausiai sudaro keturi žaidimai. Žaidėjas gali laimėti žaidimą surinkęs laiminčią „ranką“, kurią sudaro 14 kaladėlių, ir laimėti taškų, atitinkančių „rankos“ vertę. Kiekviename raundo ėjime žaidėjas paima vieną kaladėlę iš kaladėlių „sienos“ (žaidėjui nežinomas kaladėlių rinkinys, kuris nepriklauso nė vienam žaidėjui, atitinka įprastų kortų žaidimų malką), arba pasiima prieš tai ėjusio žaidėjo išmestą kaladėlę, tai vadinama „vagyste“. Žaidėjas tada išmeta kaladėlę arba paskelbia laimėjimą. Kai žaidėjas laimi naudodamas kaladėlę, kurią pats ištraukė iš „sienos“, tai vadinama laimėjimu iš sienos *(ang. winning-from-the-wall; jap. tsumo)*, ir kiti trys žaidėjai dalinasi atsakomybe sumokėti laiminčios „rankos“ taškų vertę. Žaidėjas gali laimėti pavogdamas kito žaidėjo išmestą kaladėlę ir tai vadinama laimėjimu su išmesta kaladėle *(ang. winning-by-a-discard; jap. ron)* ir žaidėjas, kuris išmetė šią kaladėlę sumoka visą laiminčios „rankos“ taškų vertę. Žaidimo pavyzdį galima pamatyti 3.1 pav.

„Mahjong“ žaidime naudojama 136 kaladėles, kurias sudaro 34 skirtingos kaladėlės, kiekvienos skirtingos kaladėlės po keturias. 108 iš šių kaladėlių yra akinės. Kiekviena kaladėlė yra vienos iš trijų rūšių (atitinkamai pavaizduotos *ženklais, rutuliukais, lazdelėmis*) ir turi numerį nuo vieno iki devynių. Likusios 28 kaladėlės yra sudarytos iš *vėjų* (*rytai, pietūs, vakarai, šiaurė*) ir *drakonų* (*balti, raudoni, žali*). Bendrai *vėjai* ir *drakonai* vadinami *kilmingomis kaladėlėmis*. Kaladėlės, kurių akių vertė yra 1 arba 9 vadinamos *terminalais*. Visos kaladėles yra pavaizduotos 3.1 lentelėje.

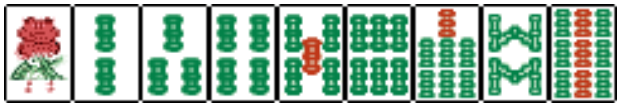
3.1 pav. „Mahjong“ stalas žaidimo metu

Kiekvienas žaidėjas turi jam priskirtą *vėją.* Žaidėjas, kuris pradeda žaidimą yra vadinamas *dalintojas (ang. dealer)*. *Dalintojo vėjas* visada yra *rytai*. Žaidėjui, sėdinčiam *dalintojui* iš dešinės, yra priskirtas *pietų vėjas*. Priešais *dalintoją* sėdinčiam žaidėjui priskiriamas *vakarų vėjas*. Dalintojui iš kairės sėdinčiam žaidėjui priskiriamas *šiaurės vėjas*. Reikia pastebėti, kad vėjai yra atvirkštinės tvarkos negu būtų pagal kompasą. Žaidėjų eiga yra prieš laikrodžio rodyklę.

Po kiekvieno žaidimo žaidėjų *vėjai* pasikeičia. Buvęs *pietų* žaidėjas tampa *dalintoju* ir jo vėjas tampa *rytais.* Atitinkamai pasikeičia ir kiti *vėjai. Dalintojas* (ir tuo pačiu *vėjai*) nesikeičia dviem atvejais – *dalintojas* laimi šį žaidimą, arba lygiųjų atveju *dalintojas* yra *tenpai* būsenoje.

Raundo vėjas yra stabilesnis *vėjas. Raundo vėjas* yra vienodas visiems žaidėjams. *Raundo vėjas* žaidimo pradžioje būna *rytai*. *Raundo vėjas* keičiasi, kai žaidėjas, kuris pirmasis buvo *dalintojas* (ir turėjo *rytų vėją*), vėl tampa *dalintoju*, t.y. kiekvienas žaidėjas pabuvo *dalintoju* bent kartą. *Raundo vėjo* tvarka keičiasi taip: *rytai-pietūs-vakarai-šiaurė. Raundo vėjo* pasikeitimas reiškia ir naujo raundo pradžią.

3.1 lentelė „Mahjong“ kaladėlės

Kaladėlių rūšies pavadinimas	Kaladėlių pavyzdžiai
Rutuliukai *(ang. dots, coins, circles; jap. pinzu)(iš kairės į dešinę nuo 1 iki 9)*	
Lazdelės *(ang. sticks, bamboos; jap. souzu)(iš kairės į dešinę nuo 1 iki 9)*	
Ženklai *(ang. characters; jap. manzu)(iš kairės į dešinę nuo 1 iki 9)*	
Vėjai *(ang. winds; jap kazehai)(iš kairės į dešinę: rytai, pietūs, vakarai šiaurė)*	
Drakonai *(ang. dragons; jap. sangenpai)(iš kairės į dešinę: baltas, žalias, raudonas)*	

Komplektai ir „vagystės“

Kiekvienas žaidėjas bando sudaryti „ranką“, susidedančią iš keturių komplektų *(ang. set, meld; jap. mentsu)* ir poros. Komplektai būna trijų tipų: trys vienodos kaladėlės, vadinamos *pung* arba *pon*, keturios vienodos kaladėlės, vadinamos *kong* arba *kan*, trijų vienos rūšies kaladėlių eilė, vadinama *chow* arba *chi*. Eilė sudaroma tik iš akinių kaladėlių. Eilės nebūna ciklišKos, t.y. kombinacija iš 8, 9, 1 kaladėlių nėra eilė.

Žaidėjas turi teisę „pavogti“ kito žaidėjo ką tik išmestą kaladėlę, jeigu žaidėjas jau turi dvi iš trijų (arba tris iš keturių, jeigu vagiama *kan*) kombinacijai reikalingų kaladėlių. Visa „pavogta“ kombinacija lieka atversta ant stalo iki žaidimo pabaigos, šių kombinacijų jau keisti negalima. Kaladėlės „vogti“, kad susidarytų pora, negalima. Eilę „vogti“ žaidėjui galima tik iš to žaidėjo, kuris yra šiam žaidėjui iš kairės, t. y. prieš jį veiksmą darantis žaidėjas.

Žaidėjas gali pats skelbti *uždarą kan*. Šis žaidėjas turi parodyti keturias vienodas kaladėles ir po to dvi iš jų apverčia nugara aukštyn. Taip parodoma, kad žaidėjas pats surinko visas keturias kaladėles.

Žaidėjui „pavogus“ arba pačiam paskelbus *kan*, kadangi žaidėjas kombinacijai panaudojo keturias kaladėles, žaidėjas iš „mirusios sienos“ ištraukia papildomą kaladėlę. „Mirusi siena“ taip vadinama todėl, kad iš jos galima traukti tik papildomą kaladėlę, kai žaidėjas paskelbė *kan*. Traukimas iš „mirusios sienos“ atliekamas tam, kad žaidėjas turėtų pakankamai kaladėlių sudaryti likusioms kombinacijoms. Kitu atveju žaidėjui pritruktų vienos (ar daugiau, jei žaidėjas paskelbė daugiau *kan*) kaladėlės .

„Pavogęs“ žaidėjas tada tęsia žaidimą lyg jis būtų ištraukęs kaladėlę iš sienos. Dėl kitų žaidėjų „vagysčių“ gali būti praleistas vieno ar kelių žaidėjų ėjimai.

Jeigu kombinacija „pavagiama“ iš kito žaidėjo, tai ir žaidėjo „ranka“ ir kombinacija laikoma atvira. Reikia pažymėti, kad paskelbtas *uždaras kan* nepadaro „rankos“ atviros, kadangi žaidėjas pats surinko visas keturias kaladėles.

Kan dar galima sudaryti prie atviro (t.y. jau „pavogto“) *pon* iš žaidėjo „rankos“ pridedant tokią pačią kaladėlę. Tai gali įvykti, kai kituose ėjimuose žaidėjas pats ištraukia reikalingą kaladėlę. *Kan* pridėjimas laikomas kaladėlės išmetimu ir šią kaladėlę kiti žaidėjai gali „vogti.“

Galimų kombinacijų pavyzdžiai matomi 3.2 lentelėje. 90° kampu pasuktos kaladėles reiškia, iš kurio žaidėjo buvo „pavogta“ kaladėlė.

Taip pat ką tik išmestą kaladėlę galima „vogti“ pergalei. Kitaip negu kombinacijų „vogimo“ atveju, kaladėlę galima „vogti“ ir tada, kai „rankai“ trūksta tik poros.

Žaidimo metu atsitinka momentų, kai vienu metu tą pačią kaladėlę nori „pavogti“ keli žaidėjai. Šie ginčai išsprendžiami pagal šiuos prioritetus: pergalė *(ron), kan/pon, chi*.

3.2 lentelė. „Mahjong“ kombinacijų pavyzdžiai

	Uždara kombinacija	**Atvira kombinacija**
Pung/Pon		
Kong/Kan		
Chow/Chi		

Dora

Kiekvieno žaidimo pradžioje viena iš „mirusios sienos“ kaladėlių yra atverčiama. Šita atversta kaladėlė (vadinama *dora indikatoriumi*) parodo, kurios kaladėlės yra *dora.* Jei atversta kaladėlė yra akinė, tai *dora* yra sekanti tos pačios rūšies kaladėlė, pvz. jei atversta kaladėlė yra rutuliukų 2, tai *dora* yra rutuliukų 3. Jei atversta kaladėlė yra 9 akių, tai *dora* bus tos pačios rūšies 1 akies kaladėlė.

Jei atversta kaladėlė yra *drakonas,* tai *dora* irgi yra *drakonas* pagal šią tvarką: atverstas raudonas *drakonas* nurodo baltą *drakoną,* baltas *drakonas* nurodo žalią *drakoną*, žalias *drakonas* nurodo raudoną *drakoną.* Vėjai atitinkamai nurodomi pagal šią tvarką: *rytai-pietūs-vakarai-šiaurė-rytai.*

Kai kuris nors žaidėjas paskelbia *kan* arba *uždarą kan,* atverčiama po papildomą *dora indikatorių* dar vadinamais *kan dora indikatoriais.* Jeigu jau yra atversta keturi *kan dora indikatoriai,* žaidėjai nebegali skelbti nei *kan,* nei *uždaro kan*, nei *pridėto kan*.

Dora kaladėlės žaidimo pabaigoje yra vertos papildomų taškų.

Tenpai* ir *riichi

Žaidėjas, kuriam iki pergalės trūksta tik vienos kaladėlės yra *laukimo* būsenoje (*jap. tenpai*). Žaidėjas yra *laukimo* būsenoje net ir tada, kai visos žaidėjo laukiamos kaladėlės yra išmestos ant stalo. Žaidėjas nėra *laukimo* būsenoje jei žaidėjas jau turi keturias laukiamų kaladėlių.

Žaidėjas su uždara *laukiančia* „ranka“, savo ėjimo metu gali skelbti *riichi.* Žaidėjas išmeta nereikalingą kaladėlę ir ant stalo padeda 1000 savo taškų. Jeigu išmesta kaladėle kitas žaidėjas laimi, *riichi* paskelbimas nesiskaito ir žaidėjas atsiima savo 1000 padėtų taškų. Žaidėjas negali paskelbti *riichi* jei „sienoje“ liko tik 4 kaladėlės.

Riichi skelbimui panaudotus taškus gauna žaidėjas, kuris laimi šią „ranką“. Jei laimi žaidėjas, kuris paskelbė *riichi* jis atgauna savo taškus. Jeigu žaidimo „ranka“ baigiasi lygiosios, *riichi* skelbimui naudoti taškai lieka ant stalo kitam žaidimui.

Žaidėjas paskelbęs *riichi* nebegali keisti savo „rankos“. Tai reiškia, kad jis negali „vogti“ kitų žaidėjų kaladėlių, nebent tai būtų „vagystė“ pergalei; taip pat turi išmesti ką tik ištrauktą kaladėlę nebent su ištraukta kaladėlė žaidėjas gali laimėti.

Žaidėjas dar gali paskelbti *uždarą kan* tik šiuo atveju - jeigu jis ištraukė ketvirtą tos kombinacijos kaladėlę, jeigu šios kombinacijos paskelbimas nekeičia žaidėjo laukiamų kaladėlių ir jeigu originalioje „rankoje“ skelbtos kaladėlės gali būti interpretuojamos tik kaip *pon*.

Žaidėjas, kuris laimėjo ir yra paskelbęs *riichi*, atverčia kaladėles esančias po *dora indikatoriais.* Šios kaladėlės yra *ura dora indikatoriai*. Šios kaladėlės nurodo *ura dora* lygiai taip, kaip *dora indikatoriai* nurodo *dora*.

Furiten

Jeigu *laukiantis* žaidėjas galėtų laimėti panaudojus vieną iš šio žaidėjo jau išmestų kaladėlių, tai šis žaidėjas yra *furiten* būsenoje ir negali laimėti „vagiant“ kitų žaidėjų kaladėles. Žaidėjas *furiten* būsenoje gali laimėti pats ištraukęs laiminčia kaladėlę.

Žaidėjas, kuris yra *furiten* būsenoje, gali pakeisti savo būseną pakeisdamas savo *laukiamas* kaladėles, nebent žaidėjas yra paskelbęs *riichi*. Žaidėjas, kuris yra *furiten* būsenoje, gali „vogti“ kaladėles kombinacijoms sudaryti, nebent žaidėjas yra paskelbęs *riichi*.

Žaidėjas, kuris yra *furiten* būsenoje, gali skelbti *riichi.* Žaidėjas, kuris nusprendžia nelaimėti su ištraukta kaladėle tampa laikinoje *furiten* būsenoje, nesvarbu ar „ranka“ su išmesta kaladėle nesudarytu *yaku*. Laikina *furiten* būsena tęsiasi iki kito žaidėjo traukimo arba kaladėlės „vogimo“ kombinacijoms užbaigti. Jeigu žaidėjas yra paskelbęs *riichi* laikinas *furiten* tęsiasi iki žaidimo pabaigos. Žaidėjas niekada nėra *furiten* būsenoje pats traukdamas kaladėlę.

Lygiosios

Lygiosios įvyksta, kai nė vienas žaidėjas nepaskelbia pergalės po paskutinės kaladėlės išmetimo. 14 „mirusios sienos“ kaladėlės įprastam žaidimui nenaudojamos.

Po lygiųjų visi žaidėjai, kurie yra *tenpai* būsenoje parodo savo kaladėles. Žaidėjas, kuris negali arba nenori parodyti *tenpai* „rankos“ vadinamas *noten.* Žaidėjai, kurie yra paskelbę *riichi*, privalo parodyti savo „rankas“.

Žaidėjai, kurie yra *noten*, bendrai sumoka 3000 taškų žaidėjams, kurie yra *tenpai*. Jeigu trys žaidėjai yra *tenpai* būsenoje, o tik vienas *noten* tai šis žaidėjas sumoka kiekvienam kitam žaidėjui po 1000 taškų. Jeigu du žaidėjai yra *tenpai* būsenoje, tai šie žaidėjai gauna po 1500 taškų iš *noten* žaidėjų. Jeigu tik vienas žaidėjas yra *tenpai*, tai šis žaidėjas iš *noten* žaidėjų gauna po 1000 taškų. Jeigu visi arba nė vienas žaidėjas yra *tenpai* būsenoje, taškų niekas niekam nemoka.

Įvykus lygiosioms ant stalo padedamas skaitiklis. Jeigu jau ant stalo yra skaitiklių, pridedamas dar vienas skaitiklis.

Chombo

Sunkūs taisyklių pažeidimai yra baudžiami *chombo* taškų bauda. *Chombo* taškų vertė yra priešinga *mangan* ribinės „rankos“ vertei. Jeigu nusižengė *dalintojas*, žaidėjas sumoka kiekvienam kitam žaidėjui po 4000 taškų. Jeigu nusižengė ne *dalintojas*, žaidėjas sumoka *dalintojui* 4000 taškų, kitiems žaidėjams po 2000 taškų. Po baudos sumokėjimo žaidimas yra pradedamas iš naujo.

Priežastys *chombo* baudai:

- Neteisėtai paskelbta pergalė, kai arba „ranka“ nėra užbaigta, arba kai „ranka“ neturi nė vieno *yaku.*
- Žaidėjas paskelbė *riichi*, nors žaidėjo ranka nėra *laukianti.* Tai įmanoma išsiaiškinti tik lygiųjų atveju. Jeigu laimi kitas žaidėjas, negu kuris paskelbė netinkamą *riichi*, bauda nėra skiriama.
- Paskelbtas neteisėtas *uždaras kan*, kai žaidėjas yra paskelbęs *riichi*. Tai įmanoma išsiaiškinti tik lygiųjų arba šio žaidėjo pergalės atveju. Jeigu laimi kitas žaidėjas, negu kuris paskelbė netinkamą *riichi*, bauda nėra skiriama.

Pergalė

Žaidėjui, kuris paskelbė laimėjimą pats ištraukęs laiminčią kaladėlę *(tsumo),* sumoka visi kiti žaidėjai. Žaidėjas, kurio išmesta kaladėle laimėjo kitas žaidėjas *(ron)*, laiminčiam žaidėjui sumoką visą laiminčio žaidėjo „rankos“ vertę pats. *Dalintojo* laimėjimas yra vertingesnis nei kitų žaidėjų su identiškoms „rankų“ vertėm, tačiau *dalintojas* sumoka daugiau jei kitas žaidėjas laimi pats ištraukęs kaladėlę *(tsumo).*

Kai laimi *dalintojas*, ant stalo padedamas skaitiklis. Jeigu jau ant stalo yra skaitiklių, pridedamas dar vienas skaitiklis. Šie skaitikliai reiškia, kad laiminčių „rankų“ vertė yra didesnė po 300 taškų už kiekvieną skaitiklį ant stalo. Skaitikliai nuimami nuo stalo, kai laimi žaidėjas, kuris nėra *dalintojas*.

Jeigu laimi *dalintojas* arba lygiųjų atveju *dalintojas* yra *tenpai* būsenoje, *dalintojas* (ir *vėjai*) nesikeičia. Kitais atvejais *dalintoju* (ir *rytais*) tampa žaidėjas, kurio vėjas buvo *pietūs*. Buvę *vakarai* tampa *pietumis, šiaurė – vakarais, rytai – šiaure.*

Partijos pabaiga

Kai sužaidžiama sutartas raundų kiekis (dažniausiai du), žaidimų partija baigiasi. Žaidėjas, turintis dagiausia taškų yra nugalėtojas. Nėra svarbu, kiek „rankų“ žaidėjas laimėjo – svarbiausia yra taškų kiekis. Įmanoma, kad įvyksta lygiųjų.

Nugalėtojas pasiima visus likusius *riichi* taškus nuo stalo. Lygiųjų atveju taškai padalinami visiems nugalėtojams vienodai. Žaidėjo taškų kiekis gali būti neigiamas. Partijos žaidėjo taškų kiekis neįtakoja. Skaičiuojant galutinius rezultatus, iš žaidėjų taškų atimama 25000 taškų su kuriais žaidėjai pradėjo žaidimą. Tada pridedamos nugalėtojų premijos.

Du geriausiai pasirodę žaidėjai iš prasčiau pasirodžiusių žaidėjų gauna taškų pagal šią tvarką: nugalėtojas gauna 15000 taškų, antros vietos laimėtojas – 5000 taškų, trečios vietos žaidėjas praranda 5000 taškų, ketvirtos vietos žaidėjas praranda 15000 taškų.

Jei yra lygiųjų, tai atitinkamų vietų taškų kiekiai padalinami žaidėjams vienodai. T.y. jeigu į pirmą vietą pretenduoja du žaidėjai, šie žaidėjai abu gauna po 10000 taškų $\left(\frac{15000 + 5000}{2} = 10000\right)$.

„Rankos“ įvertinimas

„Rankos“ vertei suskaičiuoti, pirmiausia reikia suskaičiuoti „rankos“ *han* (kartais naudojamas terminas *fan*). Tai atliekama sudedant „rankos“ *yaku* (bent vienas), *dora* kaladėles ir, jeigu laimintis žaidėjas yra paskelbęs *riichi, ura dora* kaladėles. Šita suma yra „rankos“ *han* vertė.

Tada skaičiuojama „rankos“ bazinė vertė, dar vadinama mini taškais arba *fu*. Suskaičiuotus taškus apvalinama iki didesnės dešimties, t.y. 32 apvalinama į 40. Jeigu „ranka“ sudaryta iš 7 porų, „rankos“ mini taškų vertė yra 25 ir nėra apvalinama. Jeigu „rankos“ *han* vertė yra lygi arba didesnė nei 5, mini taškų skaičiuoti nereikia.

Mini taškų skaičiavimas

Žaidėjas visada gauna mini taškų už laimėjimo būdą:

- Uždara „ranka,“ laiminti panaudojant kito žaidėjo išmesta kaladėle, yra verta 30 mini taškų
- Septynios poros yra visada vertos 25 mini taškų
- Kita (laimėjimas pačio ištraukta kaladėle ar atvira „ranka“) – 20 mini taškų

Tada pridedami mini taškai už „rankoje“ esančias kombinacijas. Eilės neturi mini taškų vertės. Jeigu laimima užbaigiant *pon* kombinaciją, šita kombinacija laikoma uždara, jeigu laimima pačio žaidėjo ištraukta kaladėle, ir atvira, jeigu laimima „vagiant“ kito žaidėjo kaladėlę. Kombinacijų vertes galima pamatyti 3.3 lentelėje.

Taip pat „ranka“ gauna po 2 mini taškus už kiekvieną:

- Porą *drakonų*
- Porą *raundo vėjų*
- Porą *žaidėjo vėjų*
- Pergalę kraštiniu, uždaru ar poros *laukimu*
- Pergalę pačio ištraukta kaladėle (išskyrus *pinfu* „ranką“)
- Atvirą *pinfu* „ranką“

3.3 lentelė. Kombinacijų mini taškų vertės

Kombinacija	**Atvira kombinacijos vertė**	**Uždaros kombinacijos vertė**
pon, pung (2-8 akys)	2	4
pon, pung (terminalai, kilmingos kaladėlės)	4	8
kan, kong (2-8 akys)	8	16
kan, kong (terminalai, kilmingos kaladėlės)	16	32

Kraštinis *laukimas* būna kai 1-2 arba 8-9 akių eilių kombinacijoms trūksta 3 ar 7 akių kaladėlių. Uždaras *laukimas* būna, kai eilei trūksta vidurinės kaladėlės, pvz. 4-6 eilei trūksta 5 akių kaladėlės. Atvira *pinfu* „ranka“ yra atvira „ranka,“ kurios mini taškų vertė (išskyrus 20 taškų už pergalę) yra 0. Ši „ranka“ papildomai įvertinama 2 mini taškais.

Tiksli „rankos“ vertė

„Rankos,“ kurių *han* vertė yra 5 arba daugiau galutinę taškų vertę galima pasižiūrėti ribinių „rankų“ vertės 3.8 lentelėje, 3.9 lentelėje.

„Rankų,“ kurių *han* vertė yra mažesnė nei 5, galutinė vertė apskaičiuojama taip:

Bazinė „rankos“ vertė (suapvalinti mini taškai) yra padvigubinami „rankos“ *han* vertės plius du kartų ($galutinė\ vertė = fu * 2^{han+2}$). Šią sumą moka visi pralaimėję žaidėjai, jeigu laimėtojas laimėjo pačio ištraukta kaladėle. Tačiau *dalintojui* ši suma yra dvigubinama dar kartą. *Dalintojas* laimi dvigubai daugiau, tačiau ir pralaimi dvigubai daugiau jei priešininkas laimėjo pats ištraukdamas kaladėlę. Galutinė vertė suapvalinama iki didesnio šimto, bet niekada neviršija *mangan* ribinės vertės.

Jeigu laimima naudojant kito žaidėjo išmestą kaladėlę, tą kaladėlę išmetęs žaidėjas sumoką visą galutinę sumą pats už visus žaidėjus. T.y. jeigu laimėjo ne *dalintojas*, žaidėjas sumoka tris kartus didesnę sumą, o jeigu laimėjo *dalintojas,* žaidėjas sumoka keturis kartus didesnę sumą. Suma suapvalinama iki didesnio šimto, bet niekada neviršija *mangan* ribinės vertės.

Prie šių verčių dar pridedama taškai už skaitiklius. Vienas skaitiklis vertas 300 taškų. Jeigu laimėta pačio žaidėjo ištraukta kaladėle, tai kiekvienas žaidėjas sumoka 100 taškų daugiau už kiekvieną skaitiklį. Jeigu laimėta „pavogus“ kito žaidėjo kaladėlę, tai pralaimintis žaidėjas sumoka 300 taškų už kiekvieną skaitiklį.

Taip pat laimėtojas susirenka visus ant stalo esančius *riichi* taškus.

Jau paskaičiuotų taškų vertes galima pamatyti 3.4 lentelėje, 3.5 lentelėje, 3.6 lentelėje, 3.7 lentelėje. 3.6 lentelė didesnis skaičius parodo kiek moka *dalintojas*, o mažesnis – kiti žaidėjai.

3.4 lentelė. *Dalintojo „*rankos“ vertė, *dalintojui* laimint pačio ištraukta kaladėle *(tsumo)*

***Han*\Mini taškai**	**20**	**25**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1 *han*			500	700	800	1000	1200	1300	1500	1600
2 *han*	700		1000	1300	1600	2000	2300	2600	2900	3200
3 *han*	1300	1600	2000	2600	3200	3900	4000	4000	4000	4000
4 *han*	2600	3200	3900	4000	4000	4000	4000	4000	4000	4000

3.5 lentelė. *Dalintojo* „rankos“ vertė, *dalintojui* laimint panaudojus kito žaidėjo kaladėlę *(ron)*

***Han*\Mini taškai**	**25**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1 *han*		1500	2000	2400	2900	3400	3900	4400	4800
2 *han*	2400	2900	3900	4800	5800	6800	7700	8700	9600
3 *han*	4800	5800	7700	9600	11600	12000	12000	12000	12000
4 *han*	9600	11600	12000	12000	12000	12000	12000	12000	12000

3.6 lentelė. Kitų žaidėjų „rankos“ vertė, žaidėjui laimint pačio ištraukta kaladėle *(tsumo)*

***Han*\Mini taškai**	**20**	**25**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1 *han*			300 500	400 700	400 800	500 1000	600 1200	700 1300	800 1500	800 1600
2 *han*	400 700		500 1000	700 1300	800 1600	1000 2000	1200 2300	1300 2600	1500 2900	1600 3200
3 *han*	700 1300	800 1600	1000 2000	1300 2600	1600 3200	2000 3900	2000 4000	2000 4000	2000 4000	2000 4000
4 *han*	1300 2600	1600 3200	2000 3900	2000 4000	2000 4000	2000 4000	2000 4000	2000 4000	2000 4000	2000 4000

3.7 lentelė. Kitų žaidėjų „rankos“ vertė, laimint panaudojus kito žaidėjo kaladėlę *(ron)*

***Han*\Mini taškai**	**25**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1 *han*		1000	1300	1600	2000	2300	2600	2900	3200
2 *han*	1600	2000	2600	3200	3900	4500	5200	5800	6400
3 *han*	3200	3900	5200	6400	7700	8000	8000	8000	8000
4 *han*	6400	7700	8000	8000	8000	8000	8000	8000	8000

3.8 lentelė. *Dalintojo* ribinių „rankų“ vertės

Ribinė „ranka“	***Mangan***	***Haneman***	***Baiman***	***Sanbaiman***	***Yakuman***
Han	5	6-7	8-10	11-12	13+
Tsumo	4000	6000	8000	12000	16000
Ron	12000	18000	24000	36000	48000

3.9 lentelė. Kitų žaidėjų ribinių „rankų“ vertės

Ribinė „ranka“	***Mangan***	***Haneman***	***Baiman***	***Sanbaiman***	***Yakuman***
Han	5	6-7	8-10	11-12	13+
Tsumo	2000 4000	3000 6000	4000 8000	6000 12000	8000 16000
Ron	8000	12000	16000	24000	32000

Yaku

Norint žaidėjui laimėti, jo „ranka“ privalo turėti bent vieną *yaku*. „Ranka“ turi *yaku*, kai „ranka“ sudaryta iš tam tikrų kombinacijų rinkinių ar ypatingos „rankos“ būsenos. Dalis kombinacijų rinkinių yra verti daugiau nei vieno *yaku*, ir kai kurie kombinacijų rinkiniai yra suderinami vienas su kitu ir šių *yaku* vertės sudedamos skaičiuojant „rankos“ *han* vertę.

Kai kurie *yaku* reikalauja, kad „ranka“ būtų uždara. Uždarą ranką galima laimėti „vagiant“ iš kito žaidėjo. Jeigu „vagiama“ kaladėlė, užbaigia *pon* kombinaciją, ši kombinacija laikoma atvira skaičiuojant mini taškus, bet „ranka“ vis dar laikoma uždara.

„Rankos“ susumuota *han* vertė negali būti daugiau nei trylika, nes ši „ranka“ tampa ribine *yakuman* vertės „ranka.“ *Yakuman* vertės „rankos“ nesisumuoja.

Rankų kombinacijų pavyzdžių galima rasti lentelėje 3.11. Visą *yaku* sąrašą galima pamatyti lentelėje 3.10.

3.10 lentelė. *Yaku* aprašymai

Lietuviškas *yaku* vertimas	**Japoniškas *yaku* pavadinimas**	***Yaku* vertė**	**Trumpas aprašymas**
Riichi	*Riichi*	1	Uždara „ranka“, žaidėjo paskelbtas *riichi*
Dvigubas *riichi*	*Daburu riichi*	2	*Riichi* paskelbtas žaidėjo pirmo ėjimo metu
Pirmu bandymu	*Ippatsu*	1	Pergalė per vieną ėjimų ratą aplink stalą nuo žaidėjo *riichi* paskelbimo
Savas traukimas	*Mentsumo*	1	Uždara ranka, pergalė traukimu iš sienos *(tsumo)*
Visos paprastos	*Tanyao*	1	Visos kombinacijos sudarytos tik iš 2–8 akinių kaladėlių
Visos eilės	*Pinfu*	1	Visos kombinacijos yra eilės, pora iš akinių kaladėlių
Vienodos eilės	*Iipeikou*	1	Uždara „ranka“, dvi kombinacijos turi būti identiškos (ir rūšys, ir akys turi būti vienodos) eilės
Eilė	*Ittsu*	2 (1)	Trys tos pačios rūšies eilių kombinacijos nuo 1 iki 9 (123, 456, 789), atvira „ranka“ verta 1 *yaku*
Vertingos kombinacijos	*Fanpai/Yakuhai*	1	Kombinacija sudaryta iš *drakonų, raundo vėjo ar žaidėjo vėjo*
Trys spalvotos eilės	*Sanshoku doujun*	2 (1)	Trys skirtingų rūšių tų pačių akių eilės, atvira „ranka“ verta 1 *yaku*
Trys spalvoti trynukai	*Sanshoku doukou*	2	Trys skirtingų rūšių tų pačių akių *pon*
Visi trynukai	*Toitoi*	2	Visos kombinacijos yra *pon*
Trys uždari trynukai	*Sanankou*	2	Trys kombinacijos yra uždaros *pon*
Trys ketvertukai	*Sankantsu*	2	Trys kombinacijos yra *kan*

Terminalai ar kilmingos kaladėlės	*Chanta*	2 (1)	Visose kombinacijose yra bent po vieną *terminalą ar kilmingą kaladėlę*, bent viena eilė, atvira „ranka“ verta 1 *yaku*
Terminalai visose kombinacijose	*Junchan*	3 (2)	Visose kombinacijose yra bent po vieną *terminalą,* bent viena eilė, atvira „ranka“ verta 2 *yaku*
Dvigubos vienodos eilės	*Ryanpeikou*	3	Uždara „ranka“, du vienodos eilės (*Iipeikou*) *yaku*
Trys maži drakonai	*Shousangen*	4 (2)	Dvi kombinacijos yra *drakonai,* pora taip pat yra *drakonai*; kadangi drakonai ir taip verti po *yaku* – „rankos“ vertė yra 4
Tik terminalai arba kilmingos kaladėlės	*Honroutou*	4 (2)	Visos kombinacijos sudarytos tik iš *terminalų* ar *kilmingų kaladėlių*
Pusiau viena rūšis	*Honitsu*	3 (2)	Visos kombinacijos yra sudarytos iš vienos rūšies arba *kilmingų kaladėlių*, atvira „ranka“ verta 2 *yaku*
Viena rūšis	*Chinitsu*	6 (5)	Visos kombinacijos yra sudarytos iš vienos rūšies kaladėlių, atvira „ranka“ verta 5 *yaku*
Žmogiška ranka	*Renhou*	5	Pergalė „pavagiant“ per pirmą žaidimo ratą
Septynios poros	*Chiitoitsu*	2	Uždara, unikali „ranka“ sudaryta iš septynių skirtingų porų
„Mirusios sienos“ traukimas	*Rinshan*	1	Pergalė traukiant iš „mirusios sienos“
Pergalė paskutine kaladėle	*Haitei/Houtei*	1	Pergalė ištraukiant arba pavagiant paskutinę kaladėlę
Apvagiant ketvertuką	*Chankan*	1	Pergalė „apvagiant“, kai kitas žaidėjas prideda kaladėlę prie *pon*, kad sudarytų *kan*
Suskaičiuotas *yakuman*	*Kazoe yakuman*	*Yakuman*	„Ranka,“ kurios *han* vertė yra 13 arba daugiau
Trylika našlaičių	*Kokushi*	*Yakuman*	Uždara, unikali ranka sudaryta iš trylikos skirtingų *terminalų ir kilmingų* kaladėlių ir papildomos vienos *terminalo ar kilmingos* kaladėlė.
Keturi uždari trynukai	*Suuankou*	*Yakuman*	Keturios kombinacijos yra uždaros *pon*
Trys dideli drakonai	*Daisangen*	*Yakuman*	Trys kombinacijos yra *drakonai*
Keturi maži vėjai	*Shousuushii*	*Yakuman*	Trys kombinacijos yra *vėjai,* pora taip pat yra *vėjai*
Keturi dideli vėjai	*Daisuushii*	*Yakuman*	Keturios kombinacijos yra *vėjai*
Visos kilmingos kaladėlės	*Tsuuiisou*	*Yakuman*	Visa „ranka“ sudaryta tik iš *kilmingų* kaladėlių
Visi terminalai	*Chinroutou*	*Yakuman*	Visa „ranka“ sudaryta tik iš *terminalų*

Žalia ranka	*Ryuuiisou*	*Yakuman*	Visa „ranka“ sudaryta tik iš žalių kaladėlių (*rutuliukų 2, 3, 4, 6, 8 ir žalių drakonų*)
Devyni vartai	*Chuuren poutou*	*Yakuman*	Uždara „ranka“ sudaryta iš vienos rūšies 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 9 ir dar vienos tos pačios rūšies kaladėlės
Dieviška ranka Žemiška ranka	*Tenhou / Chiihou*	*Yakuman*	Uždara ranka, laiminti pirma žaidėjo ištraukta kaladėle

Yaku, kuriems reikalingi tikslesni paaiškinimai:

Riichi – uždara *laukianti* ranka, paskelbtas *riichi* kartu su 1000 taškų padėjimu ant stalo.

Ippatsu (pirmu bandymu)– prie *riichi* pridedamas vieno *han* vertės papildomas *yaku*, jei žaidėjas laimi per nepertrauktą pirmą ratą aplink stalą įskaitant ir to pačio žaidėjo traukimą. Jeigu žaidimo ratas yra pertraukiamas kaladėlių „vogimu“ ar *uždaro kan, pridedamo kan* paskelbimu, *ippatsu* užsidirbti nebegalima.

Daburu riichi (dvigubas *riichi*) – prie *riichi* pridedamas vieno *han* vertės papildomas *yaku,* jei žaidėjas paskelbia *riichi* pirmu savo ėjimu. Prieš tai buvę ėjimai turi būti buvę nepertraukti, t.y. neturi būti įvykęs kaladėlių „vogimas,“ *uždaro kan* ar *pridedamo kan* paskelbimas.

Pinfu (visos eilės) – uždara „ranka,“ kurią sudaro vien eilės ir mini taškų neverta pora. Laiminti kaladėlė turi užbaigti eilę su dvipusiu laukimu, pvz. 2-3 akių eilė laukia 1 akies arba 4 akių kaladėlių. „Rankos“ mini taškų vertę sudaro tik pergalės mini taškai (20 mini taškų už pergalę pačio ištraukta kaladėle, 30 mini taškų laimint panaudojant priešininko kaladėlę).

Rinshan („mirusios sienos“ traukimas) – „ranka“ laiminti ištraukus pakeitimo kaladėlę po *kan* paskelbimo. Laikoma laimėjimu, kai pats žaidėjas ištraukė kaladėlę *(tsumo).*

Chankan (apvagiant ketvertuką) – „ranka“ laiminti „pavagiant“ kaladėlę, kurią kitas žaidėjas išmetė skelbdamas *pridėtą kan*. Kadangi *kan* nėra paskelbiamas, papildomų *dora indikatorių* atversti negalima. „Vagystė“ laikoma, kaip „pavogimas.“ Šį *yaku* galima užsidirbti ir kai *laukianti* ranka yra *trylika našlaičių* ir priešininkas skelbia *uždarą kan*. *Uždarą kan* „apvogti“ galima tik kai žaidėjas turi *trylika našlaičių* „ranką.“

Haitei (pergalė paskutine kaladėle) – „ranka“ laiminti ištraukus paskutinę kaladėlę. Nesisumuoja su *Rinshan.*

Houtei (pergalė paskutine kaladėle) – „ranka,“ kuri laimi „vagiant“ paskutinę išmestą kaladėlę, kai „siena“ jau tuščia. Šią kaladėlę galima „vogti“ tik pergalei. *Kan, pon, chi* skelbti negalima.

Chi toitsu (septynios poros) – uždara „ranka,“ sudaryta iš septynių porų. „Rankoje“ negali būti dviejų vienodų porų. *Chi toitsu* mini taškų vertė yra 25. Šie mini taškai neapvalinami ir „ranka“ negauna papildomų taškų už poras ir *laukimus*.

Sanankou (trys uždari trynukai) – „ranka,“ kurios sudėtyje yra trys uždari *pon* arba *kan*. „Ranka“ neprivalo būti uždara.

Ryanpeikou (dvigubos vienodos eilės) – uždara „ranka“, kurią sudaro dvi po dvi vienodos eilės. Nesisumuoja su *Iipeikou.*

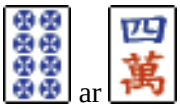
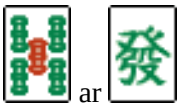
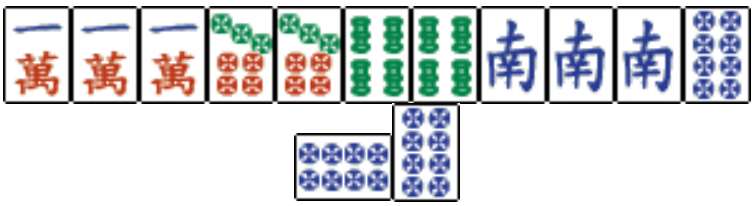
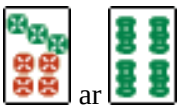
Renhou (žmogiška ranka) – „ranka,“ kuri laimi „pavogus“ kaladėlę pirmą žaidimo ratą. Jeigu žaidimo ratas buvo pertrauktas kaladėlių „vogimu“, ar *uždaro kan* skelbimu, šis *yaku* nesiskaičiuoja. *Yaku* nesisumuoja su kitais *yaku* ar *dora.*

Suuankou (keturi uždari trynukai) – keturi uždari *pon* ar *kan*. Laimėti „vagiant“ kaladėlę įmanoma tik *laukiant* poros.

Tenhou (dieviška ranka) – „ranka,“ kai *dalintojas* laimi pirmo traukimo metu. *Uždarų kan* skelbti negalima.

Chiihou (žemiška ranka) – „ranka,“ kai žaidėjas laimi pirmo traukimo metu. Prieš tai buvęs žaidimo ratas negali būti buvęs pertrauktas kaladėlių „vogimu“, ar *uždaro kan* skelbimu. Laimintis žaidėjas taip pat negali skelbti *uždaro kan.*

3.11 lentelė. Rankų kombinacijų pavyzdžiai

Rankos *yaku* ir *yaku* vertės	**„Rankos“ pavyzdys**	**„Ranka“ gali laimėti šiomis kaladėlėmis**
Visos paprastos (1)		
Galimi vertingi trejetai (*rytai*, *drakonai*) (po 1)		
Visi trynukai (2), galimas vertingas trejetas (*vakarai*) (1)		

„Mahjong“ žaidimas yra kombinatorinio pobūdžio, susijęs su kombinatorinių variantų skaičiavimu. Žaidimo variantų skaičius žaidimo eigoje didėja eksponentiškai.

3.2. Galimos „Riichi Mahjong“ žaidėjų strategijos

Kaladėlių efektyvumas yra „rankos“ vystymosi procesas, kai žaidėjas stengiasi maksimizuoti ištrauktų ir/ar išmestų kaladėlių panaudojimą, kad užbaigtų „ranką“ kaip įmanoma greičiau. Tai priklauso nuo riboto skaičiaus ištraukiamų, išmetamų kaladėlių ir žaidimo „rankų“ būsenų, kurias žaidėjas gali bandyti laimėti.

Kitaip negu pokeryje ar kituose kortų žaidimuose, „Mahjong“ „rankos“ vertė nenulemia pergalės. Užuot laimi tas žaidėjas, kuris pirmas išvysto laimėjimą „ranką.“ Žinoma, gynybiniais tikslais žaidėjas gali nepaisyti kaladėlių efektyvumo ir paaukoti to žaidimo „ranką.“

Žaidėjas gali būti *laukimo* būsenoje ir kai žaidėjo „ranka“ neturi *yaku*. Tokiu atveju žaidėjo „ranka“ privalo gauti bent vieną *yaku*, kai žaidėjas ištraukią arba „pavagia“ kaladėlę pergalei. Tarp „rankos“ *laukiamų* kaladėlių gali būti ir tokių, kurios „rankai“ šiomis kaladėlėmis laimėjus, „rankai“ nesuteikia *yaku*. Žaidėjas turi būti atsargus, kad neprarastų taškų neteisingai paskelbdamas laimėjimą.

Žaidėjas, kad neatskleistų priešininkams informacijos, gali *laukti* laimėjimo kaladėlės turėdamas uždarą „ranką.“ Tokią strategiją galima pavadinti paslėptu *tenpai*. Kadangi *riichi* šaukimas nėra privalomas, žaidėjas gali *laukti* su uždara *tenpai* ranka tikėdamasis laimėti be *riichi*. Dažniausiai tokią strategiją naudojantis žaidėjas „rankoje“ jau turi *yaku*, nes kitu atveju žaidėjas gali laimėti tik pats ištraukdamas laimėjimą kaladėlę (neskaitant ypatingų atvejų).

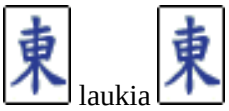
Bet kuriuo žaidimo metu, žaidėjai laukia tam tikrų kaladėlių, kad galėtų vystyti ar užbaigti savo „ranką.“ Tam tikras *laukiamų* kaladėlių išsidėstymas atsiranda, kai „rankoje“ esančios kaladėlėms trūksta kitų kaladėlių kombinacijoms ar „rankai“ užbaigti. *Laukiamos* kaladėlės ypatingai svarbu, kai

žaidėjas būna *tenpai* būsenoje. Žaidėjas turi gerai žinoti savo *laukiamas* kaladėles, nes kitaip žaidėjas gali praleisti laimėjimą kaladėlę.

Yra penki pagrindiniai dažniausiai pasitaikantys *laukiamų* kaladėlių išsidėstymai (3.12 lentelė). Pagrindiniai *laukiamų* kaladėlių išsidėstymai nepriklauso nuo kitų išsidėstymų, ir šių išsidėstymų kombinacijos sudaro sudėtingesnius *laukiamų* kaladėlių išsidėstymus. Tai ypatingai aktualu „rankoms," kuriose yra daug tos pačios rūšies kaladėlių.

Laukiančių kaladėlių išsidėstymus galima apsakyti dviem skaičiais: *n-pusių laukimas* ir *n-prieinamų* kaladėlių. Šie skaičiai skaičiuoja ir reikalingų kaladėlių tipus ir pačių kaladėlių kiekį. *N-pusių laukimai* ieško kiek skirtingų kaladėlių užbaigtų „ranką." Didžiausias *n-pusių laukimas* yra 13 trylikos našlaičių *yaku* „rankai." Mažiausias *n-pusių laukimas* yra 1. *N-prieinamų* kaladėlių skaičius priima, kad visų *laukiamų* kaladėlių yra po 4, išskyrus kaladėles, kurios jau yra „rankoje." Kitų žaidėjų „rankose," išmestos ar *dora indikatoriai* į šį skaičių neįskaičiuojami. Žaidėjai, žaidimo metu patys turi pasiskaičiuoti *prieinamų* kaladėlių kiekį.

3.12 lentelė. Pagrindiniai *laukiamų* kaladėlių išsidėstymai

Išsidėstymo pavadinimas	**Išsidėstymo aprašymas**	**Pavyzdys**
Atviras *laukimas*	Eilei trūksta šoninių kaladėlių	 laukia arba
Kraštinis *laukimas*	Kraštinei eilei (1-2-3 arba 7-8-9) trūksta 3 arba 7	 laukia
Dvigubos poros *laukimas*	Dviem poroms trūksta kaladėlės	 laukia arba
Uždaras *laukimas*, vidurinis *laukimas*	Eilei trūksta vidurinės kaladėlės	 laukia
Poros *laukimas*	Trūksta vienos kaladėlės iki poros	 laukia

Teisinga *riichi* skelbimo strategija turi daug aplinkybių. Kai kurie žaidėjai be jokių svarstymų aklai skelbia *riichi*. Kartais tokiems žaidėjams pasiseka, kartais jie būna nubausti. Nors *riichi* paskelbimas leidžia žaidėjams pasiekti didesnės vertės „rankas," žaidėjas turi apsvarstyti galimas rizikas ir priimti žaidėjui priimtiniausią sprendimą. Kartais *riichi* skelbimas nėra būtinas.

Riichi privalumai:

- Bet kokia uždara „ranka" gali laimėti panaudojant kito žaidėjo kaladėlę
- *Riichi* yra *yaku* ir po pergalės priskaičiuojamas prie „rankos" *han* vertės
- Pergalė pačio ištraukta kaladėle visada verta bent 2 *han*
- *Iipatsu* sudaro galimybę „rankos" vertei padidėti dar vienu *han*
- Po pergalės žaidėjui suteikia priėjimą prie *ura dora*
- Bet kurio žaidėjo *kan, riichi* paskelbusiam žaidėjui suteikia papildomą *dora*
- Priverčia kitus žaidėjui žaisti atsargiau ir gynybiškiau, potencialiai išardant žaidėjų „rankas"

Riichi trūkumai:

- „Ranka“ tampa nepakeičiama, todėl neįmanoma pakeisti „rankos“ sudėties tam, kad padidintum „rankos“ vertę
- Žaidėjas negali gintis ir turi išmesti ištrauktą kaladėlę
- Kiti žaidėjai gali keisti ir tobulinti savo „rankas,“ ypač jei kiti žaidėjai nusprendžia nesiginti
- Kiti žaidėjai, kurie nusprendė gintis, mažiau tikėtini, kad išmes žaidėjui reikalingą kaladėlę
- Rekomenduojama laimėti pirma pasitaikiusia laiminčia kaladėle, net jeigu šios kaladėlės vertė „rankai“ yra maža, kadangi „ranka“ gali tapti *furiten*
- Žaidėjas sumoka 1000 taškų, kad skelbtų *riichi*, tikėdamasis, kad taškus atsiloš. Jeigu žaidimą laimi kiti žaidėjai, šie *riichi* taškai atitenka kitam žaidėjui

Didžiausia *riichi* silpnybė kyla iš negalėjimo pakeisti „rankos“ būsenos po *riichi* paskelbimo. Taip žaidėjas tampa pažeidžiamu kintančioms žaidimo sąlygoms. Kartais kiti žaidėjai gali nesiginti prieš žaidėjo paskelbtą *riichi*, ir vis tiek pasiekti *tenpai* būseną išvengdami žaidėjo *riichi* „rankos.“ Taip pat *riichi* paskelbimas gali baigtis kaladėlės sužaidimu į priešininko „ranką.“ Žaidimo taškų kiekiai gali įtakoti žaidėjų pasirinkimus dėl *riichi* skelbimo. Tai dažniausiai pasitaiko partijos pabaigoje, kai žaidėjas turi daugiau nei kiti žaidėjai taškų, ir gali sau leisti neskelbti *riichi*.

Riichi ne visada padidina „rankos“ vertę. Jeigu „ranka“ verta 6 *han*, tai ar žaidėjas paskelbs *riichi* ar ne, „rankos“ galutinė vertė vis tiek bus *haneman*. Tokiais atvejais skelbti *riichi* yra nenaudinga vien dėl to, kad žaidėjas „rankos“ nebegalėtų keisti ir pasunkintų žaidėjo gynybinį žaidimą.

Taip pat žaidėjas gali neskelbti *riichi* tam, kad kiti žaidėjai išmestų žaidėjui reikalingą kaladėlę, kadangi žaidėjui paskelbus *riichi*, kiti žaidėjai žino, kad žaidėjas yra *tenpai* būsenoje, ir žaidžia atsargiau.

Gynybinis žaidimas yra remiasi viena svarbia idėja – kitų žaidėjų „rankų“ išvengimas neišmetant kito žaidėjo laiminčių kaladėlių. Taip pat žaidėjui svarbu yra neleisti kitiems žaidėjams „pavogti“ žaidėjo išmestų kaladėlių kombinacijoms sudaryti. Šita žaidimo dalis yra žymiai labiau pabrėžta po kitų žaidėjų *riichi* paskelbimo ir/ar egzistuojant didelės vertės „rankos“ grėsmei. Bet kokiu atveju žaidėjui yra svarbu prarasti kuo mažiau taškų tiesiogiai kitiems žaidėjams. Žaidėjai, kurie rečiau išmeta kitiems žaidėjams reikalingas kaladėles, laimi dažniau.

Gynybinis žaidimas yra „Mahjong“ žaidimo mokymosi kreivės dalis. Dažniausiai pradedantieji žaidėjai nesuvokia gynybinio žaidimo prasmės, kadangi pradedantiesiems žaidėjams yra svarbiau išmokti taisyklingai vystyti „ranką“ ir išmokti *yaku* kombinacijas. Nesvarbu kaip gerai žaidėjas ginasi, kad būtų pirmas , žaidėjas turi laimėti bent kelias „rankas.“ Bet nėra privaloma laimėti kiekvieną „ranką.“

Ilgainiui žaidėjai išmoksta taškų nepraradimo svarbą. Ypatingai kai taškai pralošiami tiesiogiai kitam žaidėjui. Taškų praradimas, dėl kitų žaidėjų pačių ištrauktų kaladėlių, kai taškų praradimas yra padalinamas tarp nelaiminčių žaidėjų, ar lygiųjų yra mažesnis, nei taškų praradimas sužaidžiant kitam žaidėjui į „ranką.“ Dažnai laimėti taškai, o ne pralaimėti taškai nulemia partijos baigtį.

„Mahjong“ žaidimas žaidėjams suteikia galimybę nustatyti ir išmastyti saugias kaladėles. Tai padaryti padeda žaidėjų išmestų kaladėlių rinkiniai, žaidėjų *riichi* paskelbimas, ar atviros „rankos,“ kurių „pavogtos“ kombinacijos nurodo didelę „rankos“ vertę. Saugios kaladėlės yra tokios kaladėlės, kurių kiti žaidėjai negali „pavogti“ pergalei. Žaidėjai, kurie gerai ginasi, atsižvelgia į visas matomas kaladėles ir yra gerai perpratę *furiten* taisyklę.

Svarbiausia ir paprasčiausia gynybinė strategija apima kaladėles, kurios yra garantuotai saugios. Tam naudojama *furiten* taisyklė. Bet kokia kaladėlė, kurią kitas žaidėjas yra išmetęs, yra saugi kaladėlė prieš tą žaidėją, kadangi kitas žaidėjas būtų *furiten* būsenoje, jeigu jis *lauktų* šios kaladėlės. Laikino *furiten* būsenos taisyklė užtikrina, kad tame žaidimo rate išmestos kaladėlės yra garantuotai saugios prieš žaidėjus, kurie dar neišmetė kaladėlės. Pagal šią taisyklę prieš tai ėjusio žaidėjo išmesta kaladėlė yra laikinai saugi, nebent su ta kaladėle kuris kitas žaidėjas jau laimėjo. Žaidėjo kaladėlės, kurios buvo išmestos po žaidėjo *riichi* paskelbimo, yra taip pat garantuotai saugios.

Kartais yra svarbu išmesti kaladėles, kol jos dar yra saugios. Su kiekviena ištraukta ir išmesta kaladėle žaidimo būsena keičiasi. Dažnai žaidėjai savo „rankose“ turi kitų žaidėjų laiminčias kaladėles. Šios kaladėlės turi būti panaudotos „rankos“ vystymui, išmestos laiku, kad kiti žaidėjai jų nepavogtu, arba išlaikomos rankoje, nors tai trukdo žaidėjui laimėti.

Kilmingos kaladėlės dažnai yra saugios, nes jų *laukti* yra sunkiau. Kaladėlės, kurios jau buvo išmestos yra saugesnės, nei kaladėlės, kurių ant stalo dar nėra. Ne *yaku vėjai* yra saugesni ir mažiau žalojantys taškų kiekio atžvilgiu pralaimėjimo atveju, nei kaladėlės, kurios tinka *yakuhai yaku*. *Yakuhai yaku* kaladėlės, kurių dar nėra ant stalo, yra gana pavojingos. Jei bent trys vieno tipo *kilmingos kaladėlės* jau yra ant stalo, ketvirtą galima laisvai išmesti, kadangi *kilmingų kaladėlių* negalima panaudoti eilėms sudaryti. Viena išimtis yra trylikos našlaičių „ranka,“ bet ši „ranka“ yra labai reta, todėl tokios rizikos galima nepaisyti.

Kita gynybinė strategija yra naudoja *furiten* taisyklę ir šios taisyklės panaudojimą „Mahjong“ „intervaluose.“ Kadangi eilė sudaryta iš trijų kaladėlių, jeigu žaidėjas išmetė 4 akių kaladėlę, tai galima nuspręsti, kad 1 ir 7 akių kaladėlės yra sąlyginai saugios, nes žaidėjas *laukiantis* atviru *laukimu* 2-3 arba 5-6 akių kaladėles būtų *furiten* būsenoje. 7 akių kaladėlė yra mažiau saugi, kadangi 1 akies kaladėlės *laukti* gali tik porai arba trynukui, o 7 akių kaladėlės papildomai gali *laukti* kraštiniam ar uždaram *laukimui*. Išmesta 8 akių kaladėlė nereiškia, kad 5 akių kaladėlė yra saugi, bet jeigu yra išmesta ir 2 akių kaladėlė, tada 5 yra sąlyginai saugi. Kaladėlės pagal saugumą intervalais išsidėsto taip: 1, 9 – dvigubas 4, 5, 6 intervalas – 2, 8 – 3, 7.

Iš kitos pusės ši strategija gali būti panaudota žaidėjams paspęsti spąstus. Dažniausiai kaladėlės, kurių išmetimo metu skelbiamas *riichi*, yra pavojingiausios ir naudojamos, kaip spąstai, iš priešininkų reikalingoms kaladėlėms išgauti. Kai kurie žaidėjai pasirenka blogus *laukimus* ir prieš tokius žaidėjus šita strategija yra bevertė. Intervalų strategija reikėtų naudoti, tik kai baigiasi kitos saugios kaladėlės.

Panaši į intervalų taktiką naudoja kaladėlių „sienas.“ Jeigu kokios nors kaladėlės jau išmestos 4, tai su ta kaladėle eilių sudaryti neįmanoma. „Sienų“ taktika nėra ypatingai patikima ir turėtų būti naudojama atsargiai.

Shanten yra kaladėlių skaičius, kurių reikia, kad „ranka“ pasiektų *tenpai* būseną [2]. Šis skaičius taip pat nurodo minimalų kaladėlių traukimų ar „vogimo“ kiekį, kad „ranka“ pasiektų *tenpai* būseną. Kuo mažesnis *shanten* skaičius, tuo arčiau *tenpai* būsenos yra „ranka.“ Blogiausio atvejo *Shanten* skaičių galima paskaičiuoti patikrinant kiek nenaudingų kaladėlių (vienišos *akinės* kaladėlės bent per tris akis nuo kitų kaladėlių, vienišos *kilmingos kaladėlės*) „rankoje“ yra:

$$Shanten = \min(nenaudingų\ kaladėlių\ kiekis, 6 - porų\ rankoje\ kiekis)\,, (1)$$

Tikslų *shanten* skaičių galima paskaičiuoti iš „rankos“ naiviai išimant kombinacijas, tada paskaičiuojant poras ir trumpas eiles (pora kaladėlių, kurioms trūksta vienos kaladėlės iki eilės). Iš „rankos“ skirtingai išimant kombinacijas, gali gautis skirtingas *shanten* rezultatas. Iš 1123456 akių galima išimti 345 kombinaciją, bet išimti kombinacijas 123 ir 456 yra optimaliau. Sudėtingose „rankose“ kartais gali reikėti išbandyti visas išimamas kombinacijas, kad būtų gautas tikslus rezultatas.

$$\begin{aligned} Shanten = \min\Big(&8 - 2 * kombinacijos \\ &- \max\Big(poros + trumpos\ eilės, floor\left(\frac{rankos\ dydis}{3}\right) - kombinacijos\Big) \\ &- \min\big(1, \max\big(0, poros + trumpos\ eilės - (4 - kombinacijos)\big)\big), 6 - poros, 13 \\ &- skirtingi\ terminalai\ ir\ kilmingos\ kaladėlės \\ &- \max(1, terminalų\ ir\ kilmingų\ kaladėlių\ poros)\Big), (2) \end{aligned}$$

4. ŽAIDIMO METODŲ APŽVALGA

4.1. Egzistuojantys metodai

Prognozuoti priešininkų veiksmus ir slaptas būsenas yra labai svarbu nepilnos informacijos žaidimuose. Naoki Mizukami ir Yoshimasa Tsuruoka straipsnis [3] aprašo „Mahjong“ žaidimo programos kūrimo metodą, kuris modeliuoja priešininkus ir atlieka Monte Karlo simuliacijas su šiais priešininkų modeliais. Straipsnyje priešininkų žaidimo veiksmai išskaidomi į tris elementus – *laukiamas* kaladėles, laiminčias kaladėles ir laimėjimų vertes, ir apmokomi prognozavimo modeliai šiems trims elementams naudojant patyrusių žaidėjų žaidimų rezultatus. Priešininkų veiksmai Monte Karlo simuliacijose yra nustatomi pagal priešininkų modelių tikimybių paskirstymus. Straipsnyje aprašyta programa buvo išbandyta populiariame „Mahjong“ puslapyje „Tenhou“ ir pasiekė 1718 reitingo taškų. Šis reitingas yra žymiai geresnis negu vidutinio žmogiško žaidėjo.

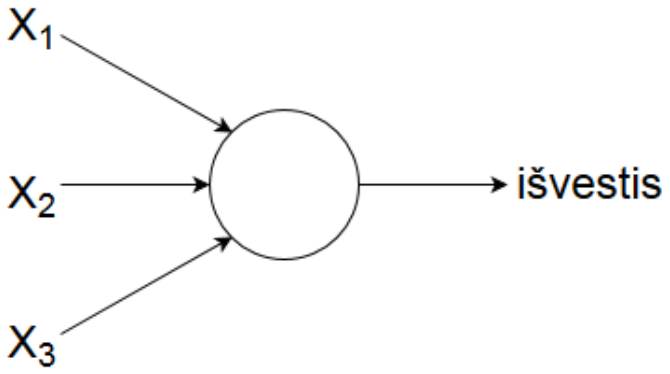
Dviejų žaidėjų „Texas Hold‘em“ žaidimo variantas, kuris yra vienas populiaraus pokerio variantų pasaulyje, nesenai buvo išspręstas apskaičiuojant apytikslią Nešo pusiausvyros strategiją [4]. Deja toks būdas ne visada galimas, kadangi Nešo pusiausvyros strategijos apskaičiavimas kai kuriuose žaidimuose gali būti brangus apskaičiavimo atžvilgiu. Kitas kompiuterinių žaidėjų kūrimo požiūris nepilnos informacijos žaidimuose yra priešininkų modeliavimas. Van der Kleij [5] išskaido žaidėjus pagal žaidėjų žaidimo stilių remiantis žaidimo rezultatais. „Rankos“ kategorinis paskirstymas yra atnaujinamas žaidimo eigoje [6]. Kompiuteriniame „Skat“ variante, Buro ir kiti [7] siūlo metodą, kuris įvertina hipotetinį pasaulį, kai žaidėjai nusprendžia savo veiksmus esamoje būsenoje naudojant žaidimų rezultatus. Hipotetinis pasaulis sudaromas paskirstant kortas tarp nežinomų žaidėjų „rankų.“ Panaudojus vidutinių realių žaidėjų žaidimų duomenis ir padarius aukšto lygio išvadas apie pasaulių ypatybes, „Skat“ žaidžianti programa žaidimą žaidžia ekspertų lygiu.

Mnih ir kitų [8] straipsnyje aprašytas gilaus mokymosi modelis, kuris sėkmingai mokosi valdymo strategiją tiesiogiai iš daugia-dimensinių įvesties duomenų naudojant pastiprinimo mokymo metodus. Modelis yra sąsūkinis neuroninis tinklas, apmokytas Q-mokymo variantu, kurio įvesties duomenys yra neapdoroti pikselių duomenys ir kurio išvesties duomenys yra vertės funkcija, kuri įvertina ateities atlygius. Šis mokymo modelis pritaikomas septyniems „Atari 2600“ žaidimams be architektūros ar mokymo algoritmų koregavimų. Šis mokymo metodas lenkia prieš tai naudotus mokymo būdus šešiuose žaidimuose ir lenkia žmones ekspertus trijuose žaidimuose.

4.2. Neuroniniai tinklai

Neuroninis tinklas yra informacijos apdorojimo modelis, įkvėptas biologinių nervų sistemų, pvz. smegenys. Pagrindinis šio modelio elementas yra neįprasta informacijos apdorojimo sistema. Ši sistema susideda iš daug tarpusavyje sujungtų kartu veikiančių apdorojimo elementų (neuronų).

Perceptronai [9] priima kelias įvestis x_1, x_2, ..., ir išveda vieną išvestį. 4.1 pav.eikslėlyje matomas perceptronas priima tris dvejataines įvestis x_1,x_2,x_3. Apskritai, neuronai gali priimti ir daugiau ir mažiau įvesčių. Įvestys turi atitinkančius svorius w_1, w_2, ..., kurie yra realūs skaičiai nurodantys atitinkančios įvesties svarbą išvesčiai. Neurono išvestis, 0 arba 1, priklauso nuo to, ar suma $\sum_j w_j x_j$ yra mažesnė ar didesnė nei tam tikra ribinė vertė b. Kaip ir kiti parametrai, perceptrono ribinė vertė yra realus skaičius.



4.1 pav. Perceptronas

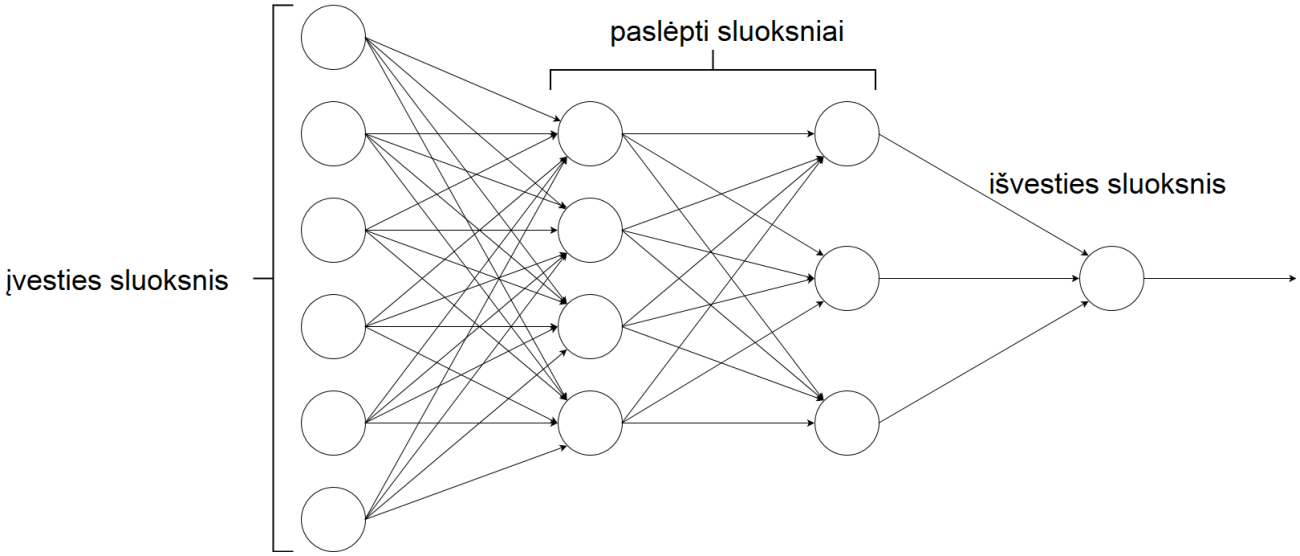
Perceptronų silpnybė yra tai, kad nedaug pasikeitus vienai įvesčiai, rezultatas gali pasikeisti iš 0 į 1 arba atvirkščiai. Modernūs neuroniniai tinklai nenaudoja perceptronų, o kitų rūšių neuronus. Viena iš šių rūšių yra sigmoidinis neuronas. Sigmoidinis neuronas, kaip ir perceptronas, turi įvestis x_1, x_2, ..., kurios kitaip nei perceptronuose, yra realūs skaičiai ir gali būti tarp 0 ir 1. Ir kaip ir perceptronas, sigmoidinis neuronas turi svorius w_1, w_2, ..., ir ribinę vertę b. Užtat sigmoidinio neurono išvestis yra paskaičiuojama pagal $\sigma(wx + b)$, kur σ yra vadinama sigmoidinė funkcija ir yra aprašyta:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}. (3)$$

Pilnai aprašyti sigmoidinio neurono išvestį pagal įvestis x_1, x_2, ..., svorius w_1, w_2, ..., ribinę vertę b galima taip:

$$\frac{1}{1-\exp(-\sum_j w_j x_j - b)}. (4)$$

Neuronai būna sujungti į sluoksniuotus tinklus (4.2 pav.). Pirmas slenkstis vadinamas įvesties sluoksniu. Paskutinis sluoksnis vadinamas išvesties sluoksniu. Tarp įvesties ir išvesties esantys sluoksniai vadinami paslėptais sluoksniais.



4.2 pav. Kelių sluoksnių neuroninio tinklo pavyzdys

Taip pat egzistuoja neuroninių tinklų modeliai, kurie naudoja grįžtamąjį ryšį. Tokie neuroniniai tinklai vadinami grįžtamaisiais neuroniniais tinklais. Šių modelių pagrindinė idėja yra neuronai, kurie aktyvuojami tam tikram trumpam laiko tarpui, o po to vėl nurimsta. Šių neuronų aktyvavimas kiek vėliau gali aktyvuoti kitus neuronus. Šie neuronai gali aktyvuoti dar kitus ir t.t.

4.3. Gilus Q neuroninis tinklas

4.3.1. Q funkcija

Esama užduotis, kurioje agentas bendrauja su aplinka Ɛ, šiuo atveju „Mahjong“ žaidimu, veiksmų, pastabų ir atlygių seka. Kiekvieno laiko momentu agentas pasirenka veiksmą a_t iš legalių žaidimo veiksmų rinkinio $A = \{1, ..., K,\}$. Šis veiksmas yra perduodamas žaidimui ir pakeičia žaidimo vidinę būseną ir žaidimo rezultatą. Ɛ gali būti stochastinė. Žaidimo vidinė būsena agento nėra matoma, agentas mato tik būseną, kurią matytų realus žaidėjas. Taip pat agentas gauna atlygį r_t, kuris nurodo žaidimo rezultato pasikeitimą. Reikia pastebėti, kad žaidimo rezultatas priklauso nuo kelių prieš tai įvykusių veiksmų ir pastabų. Atlikto veiksmo atgalinis ryšys gali būti gautas tik po daugybės laiko momentų.

Agento tikslas yra bendrauti su žaidimu parenkant veiksmus tokiu būdu, kuris maksimizuoja ateities atlygius. Priimami standartinė prielaida, kad ateities atlygiai yra atpiginti faktoriumi γ per laiko momentą, ir apibrėžiama ateities atpiginti atlygis laiku t taip

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, (5)$$

kur T yra laiko momentas, kada baigiasi žaidimas. Apibrėžiama optimalios veiksmo vertės funkcija *Q*(s, a)* kaip maksimalius pasiekiamas atlygis pasiekiamas naudojant bet kokią strategiją, pamačius tam tikrą būseną *s* ir atliekant veiksmą *a*,

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = t, \pi\right], (6)$$

kur π yra strategija, kuri susieja būsenas su veiksmais.

Optimali veiksmo vertės funkcija paklusta svarbiai savybei žinomai kaip Belmano lygybė. Tai išplaukia iš šios intuicijos – jeigu optimali būsenos *s'* vertė *Q*(s', a')* kitame laiko momente yra žinoma visiems galimiems *a'*, tada optimali strategija yra parinkti veiksmą *a'*, maksimizuojant numatomą vertę $r + \gamma Q^*(s', a')$,

$$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s', a') \middle| s, a\right]. (7)$$

Pastiprinimo mokymo algoritmų bazinė idėja yra įvertinti veiksmų vertės funkciją naudojant Belmano lygybę kaip iteracinį atnaujinimą

$$Q_{i+1}(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q_i(s', a') | s, a]. (8)$$

Tokie vertės iteraciniai algoritmai konverguoja į optimalią veiksmo vertės funkciją $Q_i \to Q^*$ kai $i \to \infty$. Praktiškai, toks bazinis būdas yra visiškai nepraktiškas, kadangi veiksmo vertės funkcija yra įvertinama atskirai kiekvienai būsenai be jokio apibendrinimo. Vietoj to, dažnai naudojama funkcijos aproksimatorius veiksmo vertės funkcijos įvertinimui

$$Q(s,a,\theta) \approx Q^*(s,a). (9)$$

Tam dažniausiai naudojama tiesinė funkcija, bet kartais naudojama ir netiesinis funkcijos aproksimatorius, pvz. neuroninis tinklas. Neuroninio tinklo aproksimatorius su svoriais θ vadinamas Q-tinklu. Q-tinklas gali būti apmokomas minimizuojant būsenos praradimo funkciją $L_i(\theta_i)$, kuri keičiasi kiekvienos *i* iteracijos metu,

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[\left(y_i - Q(s,a,\theta_i)\right)^2\right]. (10)$$

Čia

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a] (11)$$

yra iteracijos *i* tikslas ir *p(s,a)* yra būsenų *s* ir veiksmų *a* tikimybių distribucija, dar vadinama elgesio distribucija. Buvusios iteracijos θ_{i-1} parametrai yra pastovūs optimizuojant praradimo funkciją $L_i(\theta_i)$. Reikia paminėti, kad tikslai priklauso nuo tinklo svorių. Tai skiriasi nuo prižiūrimo mokymo metodų, kur tikslai yra nustatomi prieš mokymo pradžią. Diferencijuojant praradimo funkciją svorių požiūriu gauname gradientą

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]. (12)$$

Vietoj to, kad apskaičiuotume visus gradiento lūkesčius, dažniausiai vertingiau optimizuoti praradimo funkciją naudojant stochastinį gradiento nusileidimą. Jeigu svoriai yra atnaujiname po kiekvieno laiko momento ir lūkesčiai atitinkamai pakeičiami mėginiais iš elgesio distribucijos *p* ir žaidimo Ɛ, tada priartėjame prie Q-mokymosi algoritmo.

Šitas algoritmas yra nepriristas prie modelių – algoritmas sprendžia mokymosi užduotį naudodamas mėginius iš žaidimo Ɛ ir nekonstruoja aiškaus Ɛ modelio. Algoritmas taip pat mokosi godžia strategija

$$a = \max_a Q(s, a; \theta) \quad (13)$$

sekdamas elgesio distribuciją, kuri užtikrina adekvatų būsenų erdvės tyrinėjimą. Dažniausiai elgesio distribucija pasirenkama pagar ε-godžią strategiją, kuri atitinka godžiai strategijai su tikimybe $1 - \varepsilon$, ir parenka atsitiktinį veiksmą su tikimybe ε.

4.3.2. Gilus Q neuroninis tinklas

Tesauro TD-Gammon [10] architektūra siūlo pradinį tašką gilaus Q tinklo kūrimui. Ši architektūra atnaujina tinklo, kuris įvertina vertės funkciją, parametrus tiesiogiai iš patirties mėginių s_t, a_t, r_t, s_{t+1}, a_{t+1}, kurie gaunami iš algoritmo bendravimo su aplinka (arba savarankiško žaidimo Backgammon atveju). Kadangi šis būdas prieš 20 metų leido Backgammon programai aplošti geriausius žaidėjus, natūralu, kad įdomu pasvajoti kokį progresą gali atnešti dvi dešimtys technikos tobulėjimo kartu su moderniomis gilaus neuroninio tinklo architektūromis.

Palyginus su TD-Gammon ir panašiais būdais, šiame darbe naudojamas metodas, žinomas kaip patirties pakartojimas, kai agento patirtis $e_t = (s_t, a_t, r_t, s_{t+1})$ išsaugoma kiekvieną laiko momentą duomenų rinkinyje $D = e_1, \ldots, e_n$, sukauptą per daug laiko momentų. Vidiniame algoritmo cikle, pritaikoma Q-mokymo atnaujinimai, kitaip vadinami mini paketiniu atnaujinimu, naudodami patirties mėginius $e \sim D$, atsitiktinai ištrauktus iš sukauptų duomenų. Po patirties pakartojimo, agentas pasirenka ir įvykdo veiksmą pagal ε-godžią strategiją. Kadangi naudoti neapibrėžto dydžio istorijos įvestis neuroniniam tinklui gali būti sudėtinga, Q-funkcija naudoja pastovaus dydžio mėginius gaunamus panaudojus funkciją φ. Pilną algoritmą, kuris vadinamas giliu Q-mokymu, pamatyti galima 4.3 pav.

Šis būdas turi keletą privalumų palyginus su standartinius Q-mokymo algoritmu. Kiekviena patirtis potencialiai naudojama daugelį svorių atnaujinimo kartų. Tai padidina mokymo duomenų efektyvumą. Mokymasis iš nuoseklių mėginių nėra efektyvus, dėl stiprių ryšių tarp mėginių. Atsitiktinis mėginių parinkimas nutraukia šiuos ryšius ir todėl padidina duomenų variantiškumą. Naudojant įprastą mokymo metodą, dabartiniai parametrai nusako sekantį mėginį, kuriuo apmokomi dabartiniai parametrai. Taip gali iškilti nepageidaujami atgalinio ryšio ciklai ir parametrai įstrigti į prastą lokalų minimumą ar net katastrofiškai diverguoti. Naudojant patirties pakartojimą, elgesio paskirstymas yra išrenkamas iš daugybės buvusių būsenų, taip išlyginant mokymąsi ir išvengiant parametrų virpesių ar divergacijos.

Praktiškai algoritmas saugo tik paskutinius N patirties rinkinius ir tolygiai atsitiktinai parenka mėginius iš rinkinio D. Šis būdas yra dalinai apribojantis, kadangi ribota atmintis nediferencijuoja svarbius mėginius nuo nesvarbių ir panaikina seniausius mėginius naujais.

Inicijuojamas N dydžio pakartojimo duomenų rinkinys D
Inicijuojama veiksmo vertės funkcija Q su atsitiktiniais dydžiais
for episodas nuo 1 iki M **do**
    inicijuojama būsena s_1 ir apdorojama funkcija $\varphi_1=\varphi(s_1)$
    for t nuo 1 iki T **do**
        su tikimybe Ɛ atsitiktinai parenkamas veiksmas a_t
        kitu atveju parenkamas veiksmas $a_t = \max_a Q^*(\varphi(s_t), a; \theta)$
        žaidime įvykdomas veiksmas a_t ir išsaugomas atlygis r_t ir būsena s_{t+1}
        apdorojama funkcija $\varphi_{t+1}=\varphi(s_{t+1})$
        išsaugoma patirtis (φ_t, a_t, r_t, φ_{t+1}) duomenų rinkinyje D
        parenkamas atsitiktinis mėginių patirčių (φ_j, a_j, r_j, φ_{j+1}) mini paketas iš rinkinio D
        nustatomas $y_j = \begin{cases} r_j, jeigu\ \varphi_{j+1}\ yra\ galutinė\ žaidimo\ būsena \\ r_j + \max_{a'} Q^*(\varphi_{j+1}, a'; \theta), jei\ \varphi_{j+1}\ nėra\ galutinė\ žaidimo\ būsena \end{cases}$
        atliekamas $(y_j - Q(\varphi_t, a_t; \theta))^2$ gradiento nusileidimo žingsnis
    end for
end for

4.3 pav. Gilaus Q-mokymo su patirties pakartojimu algoritmo pseudokodas

4.4. Suvaržymų patenkinimo uždavinys

Suvaržymų patenkinimo uždavinys (toliau SPU) yra apibrėžta kintamųjų $X_1,X_2,...,X_n$ ir apribojimų $C_1,C_2,...,C_n$ rinkiniais [11]. Kiekvienas kintamasis x_i turi galimų verčių sritį D_i. Kiekvienas apribojimas c_i apima dalį verčių ir nurodo to verčių poaibio galimas kombinacijas. Uždavinio būsena yra apibrėžiama priskiriant vertes keliems ar visiems kintamiesiems, $\{X_i = v_i, X_j = v_j, ...\}$. Priskyrimas, kuris nepažeidžia nė vieno apribojimo, yra vadinamas pastoviu arba legaliu priskyrimu. Pilnas priskyrimas yra priskyrimas, kuriame paminimi visi kintamieji, o SPU sprendimas yra pilnas priskyrimas, kuris patenkina visus apribojimus. Kai kurie SPU reikalauja sprendimo, kuris maksimizuoja tam tikrą funkciją.

SPU yra nesunku aprašyti kaip standartinės paieškos uždavinio formuluotę:

- Pradinė būsena – tuščias priskyrimas, kuriame visi kintamieji yra nepriskirti
- Funkcija – vertė gali būti priskirta bet kokiam kintamajam, jeigu naujas priskyrimas neprieštarauja apribojimams
- Patikrinimas – esantis priskyrimas yra pilnas priskyrimas
- Paieškos kainos funkcija – kiekvieno žingsnio kaina pastovi

Kiekvienas sprendinys yra turi būti pilnas priskyrimas ir pasirodo gylyje *n*, jeigu yra *n* kintamųjų. Taip pat paieškos medis tęsiasi tik iki *n* gylio. Dėl šių priežasčių *paieškos į gylį* algoritmai yra populiarūs SPU sprendimui. Kelias, kuriuo randamas sprendimas nėra svarbus. Todėl galima naudoti pilnos būsenos formuluotę, kurioje kiekviena būsena yra pilnas priskyrimas, kuris gali ir patenkinti ir nepatenkinti apribojimus. Lokalūs paieškos metodai gerai veikia šiai formuluotei.

Paprasčiausi SPU yra sudaryti iš kintamųjų, kurie yra diskretus ir turi baigtines sritis. Jeigu bet kurio kintamojo maksimalus srities dydis yra *d*, tada galimų pilnų priskyrimų kiekis yra $O(d^n)$ – eksponentiškai pagal kintamųjų kiekį. Blogiausiu atveju, baigtinių sričių SPU galima išspręsti tik per eksponentinį laiką. Dažniausiai SPU algoritmai gali išspręsti žymiai didesnius uždavinius nei paprasti paieškos algoritmai.

Diskretūs kintamieji gali turėti begalines sritis. Su begalinėmis sritimis nebeįmanoma aprašyti apribojimų išvardinant visas galimas verčių kombinacijas, o reikia naudoti apribojimų kalbas. Taip pat neįmanoma išspręsti tokių uždavinių išvardinant visus galimus priskyrimus, nes priskyrimų yra begalė daug. Egzistuoja specialūs algoritmai skirti tiesiniams apribojimams. Įrodyta, kad neegzistuoja bendras algoritmas, kuris išspręstų uždavinį su netiesiniais apribojimais. Kartais įmanoma apriboti uždavinį į baigtinės srities uždavinį, apribojant visų kintamųjų galimas vertes.

SPU su tęstinėmis sritimis yra labai dažni realiame pasaulyje. Geriausiai žinoma tęstinių sričių SPU kategorija yra tiesinio programavimo uždavinio, kur apribojimai turi būti tiesinės nelygybės formuojančios išgaubtą regioną. Tiesinio programavimo uždaviniai išsprendžiami polinominiu laiku pagal kintamųjų kiekį.

Paprasčiausi apribojimai yra vienetiniai apribojimai, kurie apriboja vieną kintamąjį. Vienetinius apribojimus galima panaikinti apdorojant apribotų kintamųjų sritis panaikinant vertes, kurios neatitinka apribojimo. Dvimatis apribojimas yra apribojimas, kuris apriboja du kintamuosius. Aukštesnio lygio apribojimai apriboja tris ar daugiau kintamųjų.

Visi iki šiol aprašyti apribojimai yra absoliutūs apribojimai, kurių pažeidimai atmeta potencialų sprendimą. Daug realaus pasaulio SPU įtraukia ir pirmenybės apribojimus, kurie nurodo, kuriems sprendimams teikiama pirmenybė. Pirmenybės apribojimai gali būti užrašomi kaip individualių kintamųjų priskyrimo kainos. Su šia formuluote, SPU su pirmenybės apribojimais galima išspręsti naudojant optimizacijos paieškos algoritmus.

4.5. Minimalių konfliktų metodas

Lokalūs paieškos algoritmai yra efektyvūs sprendžiant įvairius SPU. Jie naudoja pilnos būsenos formuluotę – pradinė būsena kiekvienam kintamajam priskiria tam tikrą vertę ir kiti žingsniai dažniausiai keičia vieno kintamojo vertę po vieną.

Parenkant naują kintamojo vertę akivaizdžiausia euristika yra pasirinkti tokią vertę, turi mažiausiai konfliktuoja su jau parinktais kintamaisiais – minimalių konfliktų metodas. Minimalių konfliktų metodas yra stebėtinai efektyvus sprendžiant daugybę SPU, ypač kai parenkama tinkama pradinė būsena. Pavyzdžiui, *n*-karalienių uždavinį, neskaičiuojant pradinių karalienių išdėstymų, minimalių konfliktų metodas išsprendžia nepriklausomai nuo uždavinio dydžio. Šis metodas išsprendžia milijono karalienių uždavinį vidutiniškai per 50 žingsnių, neskaičiuojant pradinio karalienių išdėstymo.

Kitas lokalios paieškos privalumas yra, kad paieška gali būti panaudojama iškart, net uždaviniui kintant. Tai ypač svarbu planavimo, tvarkaraščių uždaviniams spręsti. Pavyzdžiui, oro linijų savaitinis tvarkaraštis būna sudarytas iš šimtų skrydžių ir tūkstantinių personalo skyrimo, bet blogas oras viename oro uoste gali padaryti tvarkaraštį neįvykdomu. Paprasčiausias būdas pataisyti tvarkaraštį su kuo mažiau pakeitimų yra naudoti dabartinį tvarkaraštį kaip lokalios paieškos pradinę būseną. Kiti algoritmai gali užtrukti žymiai ilgiau ir gražinti sprendimus su didesniu kiekiu pakeitimų.

Minimalių konfliktų metodo algoritmo pseudokodą galima pamatyti 4.4 pav.

funkcija min-konfliktai (*spu, maks_žingsnių_kiekis*) **gražina** sprendinį arba nesėkmę
    įvestis: *spu* – suvaržymo patenkinimo uždavinys
        maks_žingsnių_kiekis – žingsnių kiekis, po kurio algoritmas pasiduoda
einamasis <– pradinis pilnas *spu* priskyrimas
for *i* nuo 1 iki *maks_žingsnių_kiekis* **do**
    jeigu *einamasis* yra *spu* sprendinys **gražinti** *einamasis*
    kint <- atsitiktinai parinktas konfliktuojantis kintamasis iš Kintamieji[*spu*]
    ver <- kintamojo *kint* vertė *v*, kuri minimizuoja Konfliktai(*kint, v, einamasis, spu*)
    pakečiama *einamojo* kintamojo *kint* vertė į vertę *ver*
end for
gražinti *nesekmė*

4.4 pav. Minimalių konfliktų metodo pseudokodas

5. PROJEKTINĖ DALIS

5.1. Dirbtiniai „Mahjong“ žaidėjai

1. Žaidėjo sugebėjimas iš esamos „rankos“ nustatyti, kokios vertės „rankos“ siekti

Šis reikalavimas yra pats svarbiausias. Turint kaladėlių rinkinį, reikia nustatyti, kokios vertės „rankos“ siekti. „Mahjong“ žaidime yra daug įvairių vertės „rankų“ ir tinkamiausios „rankos“ padidina šansus laimėti žaidimo partiją. „Rankos“ tinkamumą galima nustatyti įvairiais būdais, iš kurių vienas yra išsiaiškinti, kiek „rankai“ trūksta kaladėlių iki pergalės. Kitas būdas – išsiaiškinti, kiek iš žaidime likusių kaladėlių pagerintų „rankos“ būseną. „Rankai“ gali trūkti tik vienos kaladėlės iki pergalės, bet jeigu jau visos keturios šios kaladėlės yra išmestos arba panaudotos, šios kaladėlės laukimas yra bevertis. Deja, tai yra sunku apskaičiuoti, kadangi kuo „ranka“ yra arčiau pergalės, tuo mažiau

skirtingų kaladėlių pagerins „rankos“ būseną. „Rankos“ vertė taip pat yra labai svarbi. Kadangi geras žaidėjas nori laimėti daugiau, reikia maksimizuoti ir „rankos“ vertę. 20 % tikimybė, kad „ranka“ laimės 8000 taškų, yra geriau negu 50 % tikimybė, kad „ranka“ laimės 1000 taškų. Tačiau negalima persistengti, kadangi žaidimų skaičius yra ribotas, neverta siekti „rankų“, kurių tikimybės yra minimalios.

2. Žaidėjo sugebėjimas pasirinkti, kada „vogti“ išmestas kaladėles

Kaladėlių traukimai yra atsitiktiniai ir priklauso nuo sėkmės, bet kitų žaidėjų išmestų kaladėlių šaukimas leidžia pasisavinti reikiamas kaladėles iš kitų žaidėjų. Kaladėlių kombinacijos, kurios suteikia žaidėjui galimybę „vogti“, iš esmės padidina žaidėjo šansus gauti reikiamas kaladėles. Tačiau kaladėlių „vogimas“ suteikia informacijos kitiems žaidėjams ir apriboja kitus žaidėjo pasirinkimus, todėl yra svarbu tiksliai apsvarstyti šiuos kompromisus.

3. Žaidėjo sugebėjimas pasirinkti kaladėlę išmetimui

Turint „rankos“ įvertinimo galimybes, gali atrodyti, kad pasirinkti kaladėlę išmetimui būtų paprasta. Tiesiog pasirenkam vieną kaladėlę iš nereikalingų „rankos“ formavimui ir ją išmetam. Deja, ne viskas taip paprasta – kai kurias kaladėles yra naudingiau išmesti nei kitas. Žaidimo partijos pradžioje dažnai pasitaiko, kad yra verta siekti keleto iš dalies perspektyvių „rankų“. Taip padidinama tikimybė laimėti žaidimą. Dažniausiai verta išmesti kaladėles, kurios netrukdo perspektyvioms „rankoms“, ir atidėti siekiamos „rankos“ sprendimo pasirinkimą vėlesniems ėjimams. Kitų sprendimų metu žaidimo būsena jau bus pasikeitus ir viena iš „rankų“ gali būti nebeperspektyvi.

Taip pat svarbu neišmesti kaladėlių, kurių šaukimu kiti žaidėjai priartėtų arčiau laimėjimo ar laimėtų, kadangi žaidėjas, su kurio išmesta kaladėle kitas žaidėjas laimi, sumoka visą „rankos“ vertę pats. Dėl šių taisyklių, žaidėjui kartais yra vertingiau paaukoti šansus į pergalę ir žaisti dėl lygiųjų, negu sumokėti visą priešininko „rankos“ vertę.

4. Žaidėjo sugebėjimas įvertinti kitų žaidėjų „rankas“

Kitų žaidėjų išmestos kaladėlės gali daug pasakyti apie kitų žaidėjų „rankų“ būseną: kokią „ranką“ šis žaidėjas siekia surinkti; kiek toli nuo pergalės šis žaidėjas yra. Jei žaidėjas nuo žaidimo pradžios neišmetė vienos rūšies kaladėlių, o žaidimui artėjant į pabaigą pradėjo tos rūšies kaladėles išmetinėti, galima manyti, kad šis žaidėjas yra arti pergalės ir kitiems žaidėjams išmesti tos rūšies kaladėles yra pavojinga.

5.2. Dirbtinių „Mahjong“ žaidėjų projektavimas

„Mahjong“ žaidime žaidėjas savo ėjimo metu turi pasirinkimą atlikti vieną iš penkių veiksmų:

- Laimėti pačio ištraukta kaladėle
- Išmesti kaladėlę
- Paskelbti *riichi*
- Paskelbti *uždarą kan*
- Paskelbti *pridėtą kan*

Kitam žaidėjui išmetus kaladėlę žaidėjas turi pasirinkimą atlikti vieną iš penkių veiksmų:

- Laimėti kito žaidėjo išmesta kaladėle
- Paskelbti *kan*
- Paskelbti *pon*
- Paskelbti *chi*
- Pasuoti

Išmetant kaladėlę, žaidėjas turi pasirinkimą, kokią kaladėlę išmesti iš rankos, nebent žaidėjas yra paskelbęs *riichi*. Teisingas kaladėlės pasirinkimas yra labai svarbus veiksmas „rankos“ vystymuisi. *Riichi* paskelbimas priklauso nuo dabartinės „rankos“ vertės. *Riichi* skelbti labiausiai apsimoka, kai „rankos“ vertė yra maža. Kombinacijų „vogimas“ priklauso nuo „rankos,“ kurią žaidėjas bando surinkti. Kai kurios „rankos“ neleidžia „vogti“ iš viso ar tik tam tikrų kombinacijų.

5.3. Pasiūlyti euristiniai „Mahjong“ žaidimo algoritmai

Šiame darbe analizuojami šeši skirtingi algoritmai:

- Godus atsitiktinis algoritmas

- Godus euristinis algoritmas
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi
- Giliu Q neuroniniu tinklu pagrįstas algoritmas
- Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstas algoritmas
- Brutalios jėgos metodu pagrįstas algoritmas

5.3.1. Godus atsitiktinis algoritmas

Godaus algoritmo strategija parenka lokaliai optimalų pasirinkimą kiekvienu laiko momentu tikėdamasi pasiekti globalų optimumą. Daugelyje uždavinių godus algoritmas dažniausiai nepasiekia optimalaus sprendimo, bet visgi godi euristika gali duoti lokaliai optimalius sprendimus, kurie aproksimuoja globalų optimumą per priimamą laiko periodą.

Godus atsitiktinis algoritmas žaidėjo ėjimo metu veiksmus pasirenka iš tuo metu prieinamų veiksmų pagal šį prioritetų sąrašą:

1. Pergalė savo ištraukta kaladėle
2. Paskelbti *uždarą kan*
3. Paskelbti *pridėta kan*
4. Paskelbti *riichi*
5. Išmesti atsitiktinę kaladėlę

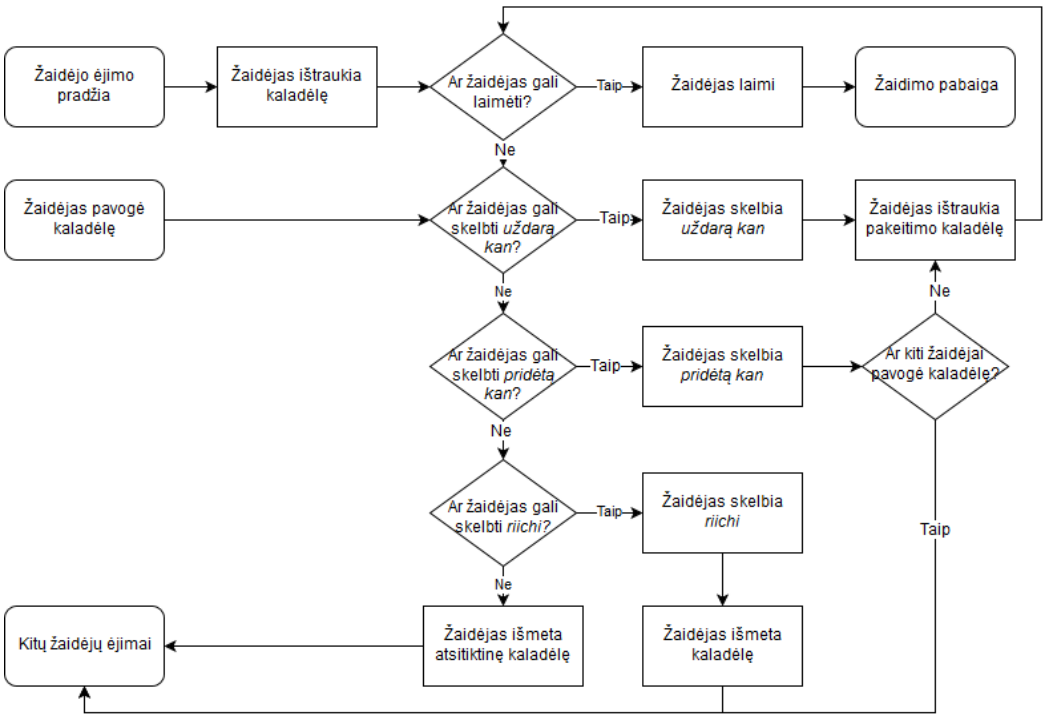
Nors pergalės paskelbimas ne visada geriausias globalus sprendimas, kadangi pergalė su kita kaladėle galėtų uždirbti daugiau taškų, lokaliai pergalė yra geriausias prieinamas sprendimas. *Kan* skelbimas žaidėjui ištraukia papildomą kaladėlę iš „mirusios sienos,“ todėl žaidėjas turi didesnį pasirinkimą kaladėlių išmetimui. *Riichi* paskelbimas užtikrina, kad jeigu žaidėjas laimės žaidėjo „ranka“ turės bent vieną *yaku*. Atsitiktinis kaladėlės išmetimas nustato šį algoritmą, kaip kontrolinį, su kuriuo būtų galima palyginti kitų algoritmų veikimą, kadangi kaladėlės pasirinkime nėra jokių kitų skaičiavimų.

Godus atsitiktinis algoritmas kito žaidėjo ėjimo metu veiksmus pasirenka iš tuo metu prieinamų veiksmų pagal šį prioritetų sąrašą:

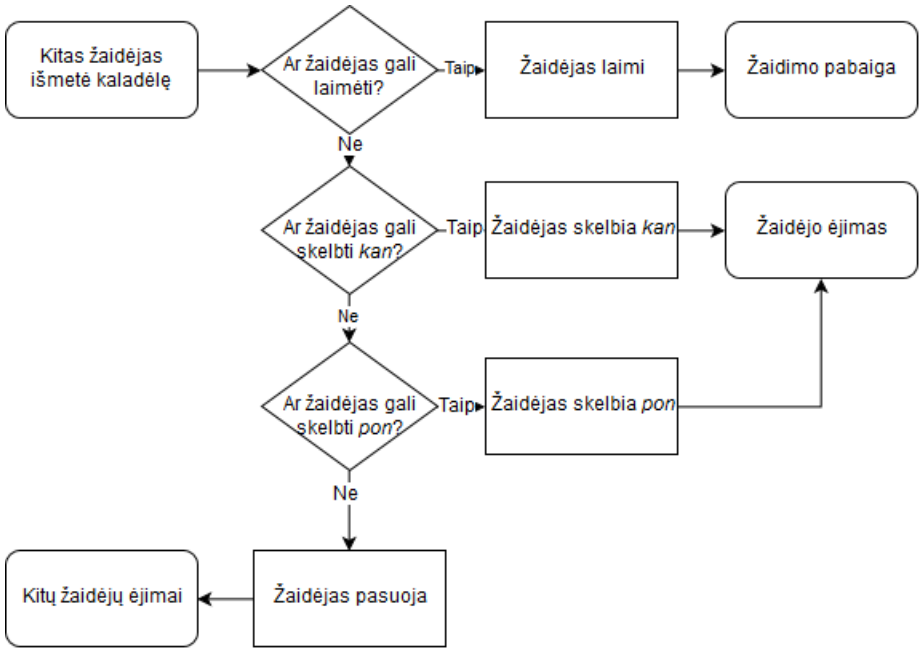
1. Pergalė naudojant kito žaidėjo išmestą kaladėle
2. Paskelbti *kan*
3. Paskelbti *pon*
4. Pasuoti

Pon paskelbimas potencialiai priartina žaidėjo „ranką“ prie pergalės, kadangi žaidėjas užbaigia kombinaciją. Reikia pastebėti, kad šis algoritmas neskelbia *chi*, kadangi neribotas *chi* skelbimas nukreipia „rankos“ vystymąsi į akligatvį. Dauguma „rankų“ kombinacijų, kuriose leidžiamas *chi*, *chi* kombinacijos būna apribotos kitaip, pvz. dvi vienodos *chi* kombinacijos ar trijų *chi* kombinacijų eilė.

Veiksmo pasirinkimo algoritmą galima pamatyti 5.1 pav. ir 5.2 pav.



5.1 pav. Godaus atsitiktinio žaidėjo ėjimo veiksmų algoritmo blokinė schema



5.2 pav. Godaus atsitiktinio žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė

5.3.2. Godus euristinis algoritmas

Godus euristinis algoritmas elgiasi identiškai godžiam atsitiktiniam algoritmui, su viena išimtimi. Žaidėjas pasirenka išmetamą kaladėlę pagal euristinį algoritmą, kuris kaladėlėms priskiria pagal žaidimo būsena apskaičiuotas vertes. Tada žaidėjas išmeta kaladėles, kurių apskaičiuota vertė yra mažiausia.

Kaladėlių vertės apskaičiuojamos pagal šį algoritmą, kurį galima pamatyti 5.3 pav.: Suskaičiuojamos visų skirtingų matomų kaladėlių kiekiai (žaidėjo „ranka,“ visų žaidėjų „pavogtos“ kombinacijos, žaidėjų išmestų kaladėlių rinkiniai). Tada apskaičiuojamos žaidėjo „rankos“ kaladėlių vertės. Jeigu „rankoje“ yra *drakonai, raundo ir žaidėjo vėjų* kaladėlės, šių kaladėlių vertės padidinamos 1 už kiekvieną tokią kaladėlę. Pvz. jeigu „rankoje“ yra pora *žalių drakonų, žalių*

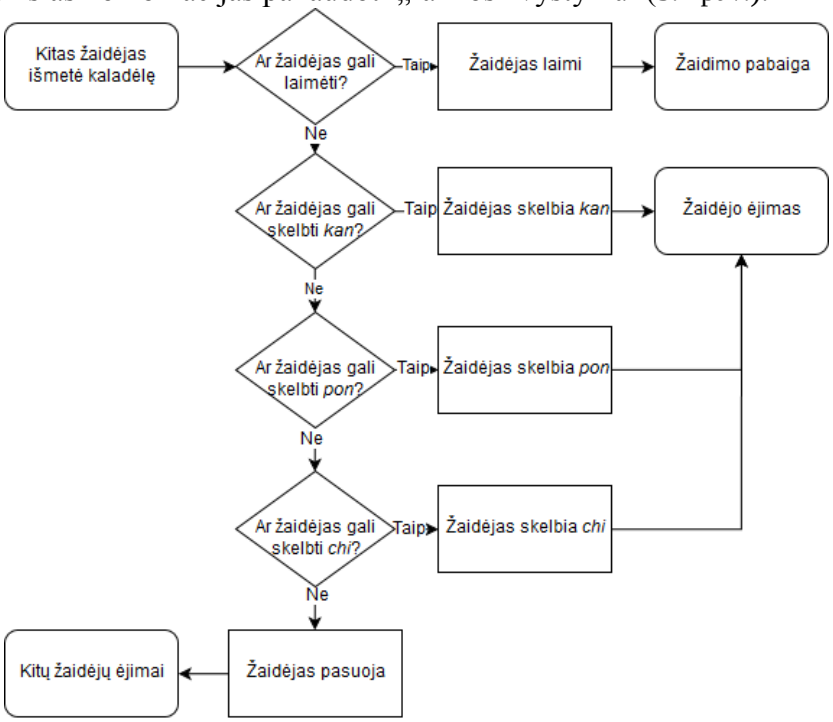
drakonų vertė yra 2. Jeigu „rankoje“ esančios kaladėlės yra *akinės*, tos kaladėlės vertė yra padidinama 0,5, o kaladėlių vertės, kurios skiriasi per dvi akis nuo tikrinamos kaladėlės, padidinamos irgi po 0,5. Tai atliekama tam, kad žaidėjas neišmestų „rankoje“ esančių eilių.

Tada iš apskaičiuotų kaladėlių verčių atimamos suskaičiuoti matomų kaladėlių kiekiai. Taip gaunamos vertės, kurios įvertina jau išmestas ar kitaip panaudotas kaladėles. Tada galutines vertes surūšiuojamos didėjimo tvarka. Tada „rankoje“ surandama kaladėlė su mažiausia verte ir ši kaladėlė parenkama išmetimui.

```
Inicijuojamas 34 skaičių masyvas matomoms kaladėlėms skaičiuoti
Prie matomų kaladėlių masyvo pridedama „rankoje“ esančios kaladėlės
Prie matomų kaladėlių masyvo pridedamos žaidėjų „vogtos“ kombinacijos, žaidėjų išmestų kaladėlių rinkiniai
Inicijuojamas 34 skaičių masyvas kaladėlių vertėm apskaičiuoti
for kaladėlė žaidėjo „rankoje“ do
    jeigu kaladėlė yra drakonas, žaidėjo ar raundo vėjas,
        šios kaladėlės vertė kaladėlių verčių masyve padidinama 1
    jeigu kaladėlė yra akinė, šios kaladėlės vertė kaladėlių verčių masyve padidinama 0,5
        taip pat padidinamos kaladėlių, kurios skirias per 2 akis, vertės po 0,5
end for
Iš kaladėlių verčių masyvo atimamas matomų kaladėlių masyvas
Surūšiuojamas kaladėlių verčių masyvas didėjimo tvarka
for kaladėlė verčių masyve do
    jeigu kaladėlė „rankoje,“ parenkama ši kaladėlė išmetimui ir užbaigiamas algoritmo darbas
end for
```

5.3 pav. Kaladėlės verčių apskaičiavimo algoritmo pseudokodas

Reikia pastebėti, kad šis algoritmas gali „apvogti“ *chi*, kadangi algoritmas naiviai supranta eilių kombinacijas ir gali šias kombinacijas panaudoti „rankos“ vystymui (5.4 pav.).



5.4 pav. Godaus euristinio žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė schema

5.3.3. Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi

Godus euristinias algoritmas pritaikytas greitam „rankos“ vystymuisi yra modifikuotas godaus euristinio algoritmo variantas. Modifikuotas algoritmas naudoja kitokį kaladėlių verčių apskaičiavimo algoritmą (5.5 pav.). Taip pat algoritmas naudoja kitokią sprendimo metodiką kitų žaidėjų kaladėlių „vagystei“ nuspręsti (5.6 pav.).

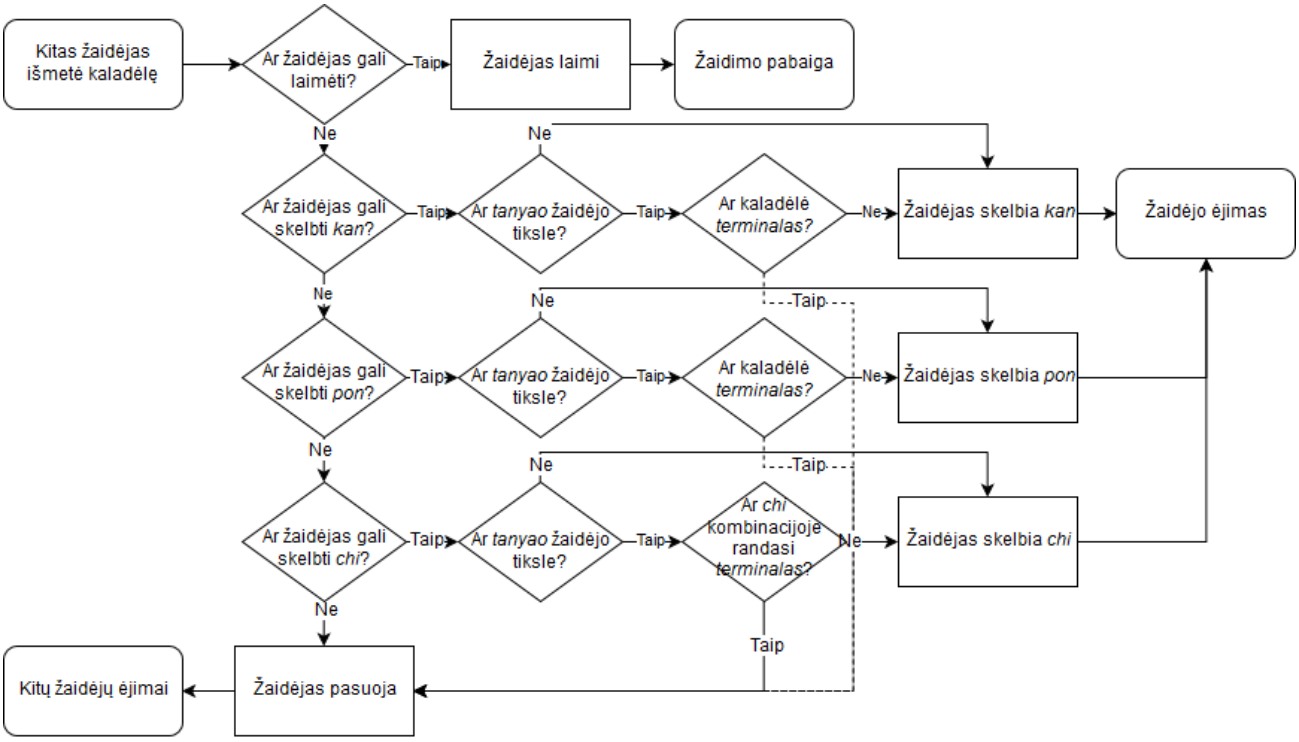
Modifikuotas algoritmas tikrina prieš skaičiuodamas kaladėlių vertes patikrina ar žaidėjas turi vertingų kaladėlių komplektą ar galimybę šį komplektą surinkti. Jeigu „rankoje“ yra vertingų kaladėlių kombinacija arba galimybė šį komplektą surinkti, į žaidėjo nusitaikytų „rankų“ sąrašą įrašomas vertingų kaladėlių (*yakuhai*) *yaku*. Tada tikrinama ar gali nesudėtingai sudaryti trynukų kombinacijas. Jeigu „rankoje“ randama bent keturios poros ir žaidėjas nėra „pavogęs“ eilės, į žaidėjo nusitaikytų „rankų“ sąrašą įrašomas keturių trynukų (*toitoi*) *yaku*. Tada tikrinama ar žaidėjas yra „pavogęs“ *terminalų ar vertingų* kaladėlių. Neigiamu atveju į žaidėjo nusitaikytų „rankų“ sąrašą įrašomas paprastų kaladėlių (*tanyao*) *yaku*.

Tada apskaičiuojamos kaladėlių vertės. Jeigu žaidėjas taikosi į arba turi *vertingų kaladėlių* komplektą, žaidėjas seka godaus euristinio modelio nustatytą tvarką. Jeigu žaidėjas taikosi į keturis trynukus, *akinių* kaladėlės neduoda papildomos vertės aplink šią kaladėlę esančioms kitoms kaladėlėms. Jeigu žaidėjas taikosi į paprastų kaladėlių „ranką,“ vertė neduodama *terminalams ar kilmingoms kaladėlėms,* lyg ranka jų neturėtų. Likusi kaladėlių vertės algoritmo dalis atitinka godaus euristinio metodo algoritmą.

Algoritmas, kuris nusprendžia ar „vogti“ kaladėles iš kitų žaidėjų, tikrina ar išmesta kaladėlė yra terminalas ir ar „rankos“ taikinys yra paprastos kaladėlės (*tanyao*). Jeigu abi sąlygos tenkinamos, tada kaladėlė „nevagiama,“ kadangi naujai „pavogta“ kombinacija nebeleistų „rankai“ laimėti naudojant paprastų kaladėlių (*tanyao*) *yaku*.

```
inicijuojamas 34 skaičių masyvas matomoms kaladėlėms skaičiuoti
prie matomų kaladėlių masyvo pridedama „rankoje“ esančios kaladėlės
prie matomų kaladėlių masyvo pridedamos žaidėjų „vogtos“ kombinacijos, žaidėjų išmestų kaladėlių
    rinkiniai
jeigu „rankoje“ yra vertingų kaladėlių kombinacija arba galimybė ją pasiekti, į „rankos“ taikinių sąrašą
    įrašoma yakuhai yaku
jeigu „rankoje“ yra bent keturios poros ir žaidėjas nėra „pavogęs“ eilės, į „rankos“ taikinių sąrašą įrašoma
    toitoi yaku
jeigu žaidėjas nėra pavogęs nepaprastų kaladėlių, į „rankos“ taikinių sąrašą įrašoma tanyao yaku
inicijuojamas 34 skaičių masyvas kaladėlių vertėm apskaičiuoti
for kaladėlė žaidėjo „rankoje“ do
    jeigu kaladėlė yra drakonas, žaidėjo ar raundo vėjas
        jeigu yakuhai arba toitoi yra „rankos“ taikinių sąraše, šios kaladėlės vertė kaladėlių verčių masyve
            padidinama 1
    jeigu kaladėlė yra akinė
        jeigu yakuhai yra „rankos“ taikinių sąraše, šios kaladėlės vertė kaladėlių verčių masyve padidinama
            0,5
            taip pat padidinamos kaladėlių, kurios skirias per 2 akis, vertės po 0,5
        jeigu tanyao yra „rankos“ taikinių sąraše ir kaladėlė nėra terminalas, šios kaladėlės vertė kaladėlių
            verčių masyve padidinama 0,5, taip pat padidinamos kaladėlių, kurios skirias per 2 akis
            išskyrus terminalus, vertės po 0,5
end for
iš kaladėlių verčių masyvo atimamas matomų kaladėlių masyvas
surūšiuojamas kaladėlių verčių masyvas didėjimo tvarka
for kaladėlė verčių masyve do
    jeigu kaladėlė „rankoje,“ parenkama ši kaladėlė išmetimui ir užbaigiamas algoritmo darbas
end for
```

5.5 pav. Modifikuotas kaladėlės verčių apskaičiavimo algoritmo pseudokodas



5.6 pav. Modifikuoto godaus euristinio žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė schema

Šis algoritmas naudojamas greitai išvystyti „ranką," kadangi visi trys parinkti „rankų" tipai yra greičiau surenkami palyginus su kitais didesnės vertės „rankų" tipais. Žinoma toks algoritmas gali dažniau laimėti, bet vidutinė pergalės vertė bus mažesnė nei metodų, kurie naudoja visų tipų „rankas."

5.3.4. Giliu Q neuroniniu tinklu pagrįstas algoritmas

Giliu Q neuroniniu tinklu pagrįstas algoritmas naudoja neuroninius tinklus, kaip Q funkciją, veiksmų vertėms apskaičiuoti. Šis metodas naudoja du atskirus vienodos konfigūracijos neuroninius tinklus. Vienas neuroninis tinklas skirtas žaidėjo veiksmams nustatyti, kitas – žaidėjo išmestoms kaladėlėms, kadangi kaladėlių išmetimas yra vienas svarbiausių žaidimo veiksmų. Abu neuroniniai tinklai priima 384 informacijos įvestis:

- Žaidėjo „rankos" kaladėlių histograma – 34 sveiki skaičiai
- Visų žaidėjų „pavogtų" kaladėlių histogramos – 4*34 sveiki skaičiai
- Visų žaidėjų išmestų kaladėlių histogramos – 4*34 sveiki skaičiai
- Ką tik išmestos arba žaidėjo ištrauktos kaladėlės histograma – 34 sveiki skaičiai
- Žaidimo *dora indikatorių* histograma – 34 sveiki skaičiai
- Žaidimo būsena (ar žaidėjas ištraukęs kaladėlę, ar kiti žaidėjai išmetę kaladėlę) – 1 sveikas skaičius
- Žaidimo ėjimo numeris – 1 sveikas skaičius
- Žaidėjų *riichi* paskelbimo būsena – 4 sveiki skaičiai
- Žaidėjų turimi taškai – 4 sveiki skaičiai

Žaidėjo veiksmų neuroninio tinklo išvestis sudaryta iš 10 realių skaičių, kurie nurodo atitinkamo veiksmo potencialų ateities atlygį. Žaidėjo išmetamų kaladėlių neuroninio tinklo išvestis sudaryta iš 34 realių skaičių, kurie potencialų ateities atlygį išmetus atitinkamą kaladėlę. Abu neuroniniai tinklai sudaryti iš dviejų slaptų sluoksnių, kurių pirmame yra 512 neuronų, o antrame 256.

Žaidimo metu, atėjus neuroninio tinklo žaidėjo ėjimui, žaidimas perduoda žaidimo būseną s_t neuroniniam tinklui būsenos įvertinimui. Neuroninis tinklas gražina dešimties skaičių masyvą a. Masyvas a surūšiuojamas mažėjimo tvarka arba su ε tikimybe yra atsitiktinai sumaišomas, ir tikrinama ar pirmą masyvo veiksmą a_t galima atlikti. Jeigu taip, tada veiksmas a_t įvykdomas ir išsaugoma žaidimo būsena s_t bei žaidimo atlygis r_t. Atlygis lygus žaidėjo taškų pokyčiui padalinus iš 1000. Jeigu

veiksmas a_t nėra galimas, tada šis veiksmas pašalinamas iš veiksmų masyvo *a* ir neuroninis tinklas apmokomas, kad veiksmas yra paskutinis, kadangi šis veiksmas neatitiktų žaidimo taisyklių. Šio algoritmo pseudokodą galima pamatyti 5.7 pav.

Ateities neuroninio tinklo ėjimo metu išsaugota sena būsena kartu su atlygiu ir nauja būsena siunčiami neuroniniam tinklui apmokyti.

Apmokymai vyksta pagal modifikuotą gilaus Q tinklo apmokymo algoritmą. Kadangi žaidimą žaidžia keturi žaidėjai, algoritmas modifikuotas taip: žaidimo pradžioje inicijuojami pakartojimo duomenų rinkiniai *D* ir veiksmo vertės funkciją atstojantis neuroninis tinklas *Q*. Tada žaidimo metu gaunant žaidimo būsena s_{t-1}, veiksmą a_{t-1}, atlygį r_{t-1}, būseną s_t, ši informacija išsaugoma pakartojimo duomenų rinkinyje *D*. Iš šio rinkinio parenkama atsitiktinis mėginių (s_j, a_j, r_j, s_{j+1}) mini paketas, apskaičiuojama mini paketo atlygių vertės, priklausomai nuo to ar būsena s_{j+1} yra galutinė žaidimo būsena. Šio algoritmo pseudokodą galima pamatyti 5.8 pav.

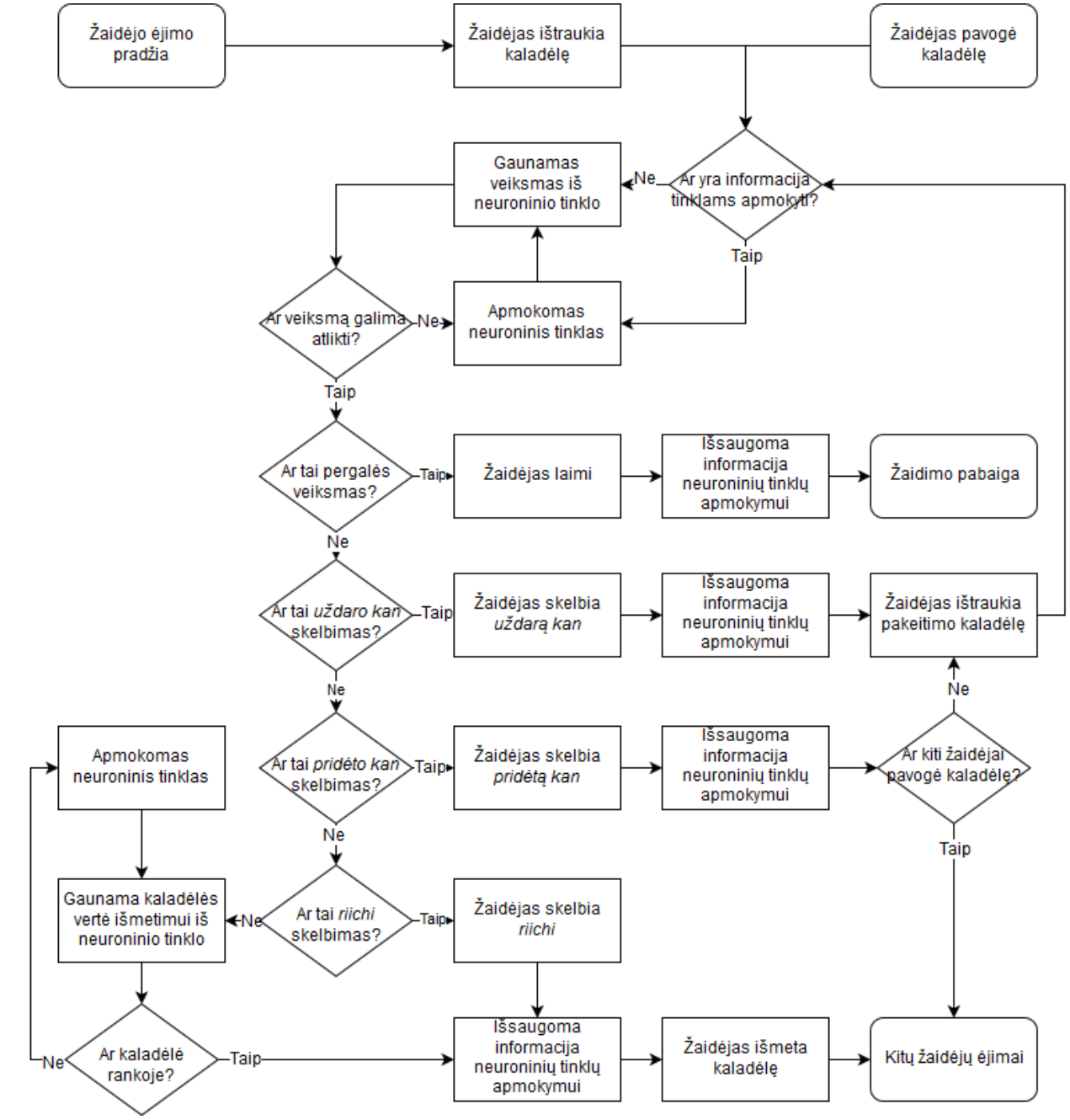
Neuroninių tinklų gradiento nusileidimo žingsniui optimizuoti naudojamas Adam stochastinio optimizavimo metodas.

```
iš žaidimo gaunama žaidimo būsena s_t
jeigu yra išsaugota būsena s_{t-1}, veiksmas a_{t-1}, atlygis r_{t-1}
        apmokoma Q funkcija pagal būsena s_{t-1}, veiksmą a_{t-1}, atlygį r_{t-1}, būseną s_t
įvertinama funkcija Q(s_t) ir gaunamas verčių masyvas a
su tikimybe Ɛ masyvas a atsitiktinai sumaišomas
kitu atveju a surūšiuojamas mažėjimo tvarka
for veiksmai a masyve do
        jeigu galima atlikti pirmą a masyvo veiksmą a_t
                atliekamas veiksmas a_t ir gaunamas atlygis r_t
                išsaugoma būsena s_t, veiksmas a_t, atlygis r_t
                baigiamas šio algoritmo darbas
        kitu atveju apmokoma Q funkcija pagal būseną s_t, veiksmą a_t, atlygį r_b, ir baigtinę būseną s_e
                šalinamas a_t iš a masyvo
end for
```

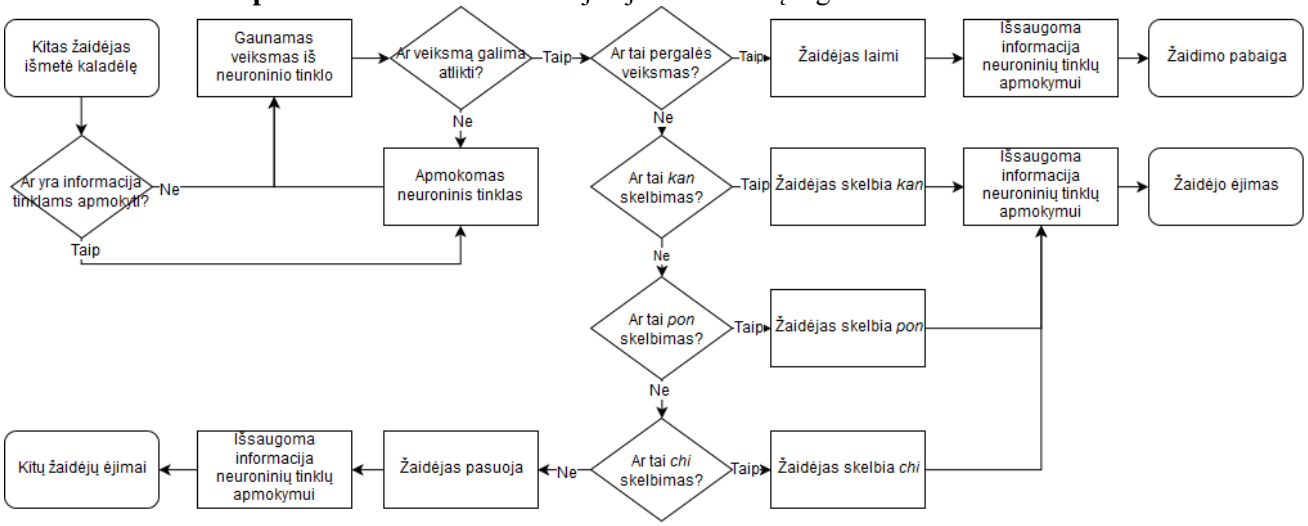
5.7 pav. Neuroninio tinklo veiksmo atlikimo algoritmo pseudokodas

```
žaidimo pradžioje inicijuojamas N dydžio pakartojimo duomenų rinkinys D
        inicijuojama veiksmo vertės funkcija Q su atsitiktiniais dydžiais
žaidimo metu
        iš žaidimo gaunama būsena s_{t-1}, veiksmas a_{t-1}, atlygis r_{t-1,} būsena s_t
        išsaugoma patirtis (s_{t-1}, a_{t-1}, r_{t-1}, s_t) duomenų rinkinyje D
        parenkamas atsitiktinis mėginių patirčių (s_j, a_j, r_j, s_{j+1}) mini paketas iš rinkinio D
        nustatomas y_j = { r_j, jeigu s_{j+1} yra galutinė žaidimo būsena
                        { r_j + max Q*(s_{j+1}), jei s_{j+1} nėra galutinė žaidimo būsena
        atliekamas (y_j – Q (s_t))^2 gradiento nusileidimo žingsnis
```

5.8 pav. Neuroninio tinklo apmokymo algoritmo pseudokodas



5.9 pav. Neuroninio tinko žaidėjo ėjimo veiksmų algoritmo blokinė schema



5.10 pav. Neuroninio tinko žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė schema

5.3.5. Suvaržymų patenkinimo uždaviniu minimalių konfliktų metodu pagrįstas algoritmas

Suvaržymų patenkinimo uždaviniu minimalių konfliktų metodu pagrįstas algoritmas remiasi tuo, kad nustatymas, ar „ranka“ yra laiminti, yra suvaržymų patenkinimo uždavinys.

Algoritmo veikimo pradžioje yra aprašomi visų galimų rankų *yaku* apribojimai. Kiekvienam *yaku* yra sukuriama naujas uždavinys pagal to *yaku* aprašymą. Pvz. trys trynukai *yaku* apribojami keturiais *pon* ir vienu poros apribojimais. Visi skirtingi apribojimai randami 5.1 lentelė. Apribojimai gali būti pritaikomi vienai kaladėlei, kelioms kaladėlėms ar visoms „rankos“ kaladėlėms. Jeigu 5.1 lentelė apribojimo apribotų kaladėlių kiekis pažymėtas brūkšneliu, reiškia šis apribojimas gali būti pritaikytas neribotam kiekiui kaladėlių. Vienintelė išimtis yra uždaros rankos apribojimas, kuris yra naudojamas visai „rankai.“

Minimalių konfliktų sprendėjas (toliau – sprendėjas), tai algoritmas, kuris priima uždavinį ir bando jį išspręsti.

Algoritmas iš žaidimo gavęs žaidėjo „ranką,“ prie „rankos“ prideda vieną „džiokerinę“ kaladėlę, kuri gali būti bet kokia kita kaladėle. „Džiokerinę“ kaladėlę galima panaudoti daugiau nei vieną kartą. „Džiokerinė“ kaladėlė naudojama tam, kad būtų įmanoma išspręsti uždavinį neturint visų „rankai“ reikalingų kaladėlių. Sprendėjas sprendžiant uždavinį bando minimizuoti „džiokerinių“ kaladėlių kiekį, nes kitu atveju gautą sprendimą visą laiką sudarys tik „džiokerinės“ kaladėlės.

Algoritmas patikrina ar uždavinys ir „ranka“ yra uždari. Jeigu uždavinys turi uždaros „rankos“ apribojimą ir žaidėjas yra „pavogęs“ kaladėlių, šis uždavinys nesprendžiamas ir algoritmas bando spręsti kitą uždavinį. Algoritmas perduoda sprendėjui žaidėjo „ranką“ ir uždavinį, ir sprendėjui pabaigus darbą gauna rezultatą. Rezultatas yra kaladėlės, kurios labiausiai atitinka sprendžiamą uždavinį, t.y. kiek „rankoje“ esančių kaladėlių atitinka tam tikrą *yaku*. Tada iš realios (be „džiokerinių“ kaladėlių) „rankos“ išimamos kaladėlės gautos iš sprendėjo. Taip gaunamas „rankai“ nereikalingų kaladėlių sąrašas tam tikram *yaku* siekti. Šis sąrašas kartu su skaičiumi, kuris nurodo kiek kaladėlių atitinka uždavinį, bei uždavinio *yaku* verte yra išsaugomas. Tada bandoma spręsti kitą uždavinį.

Algoritmui išsprendus visus uždavinius, gautas rezultatų sąrašas apdorojamas. Iš rezultatų sąrašo paimami rezultatai, kurių atitinkančių kaladėlių skaičius plius uždavinio *yaku* yra didžiausi. Taip įvertinama ir „rankų“ *yaku* vertė ir kiekis kaladėlių, kiek trūksta iki pergalės. Prie išrinkto sąrašo tada pridėdami matomų kaladėlių kiekiai, panašiai kaip ir godžiuose euristiniuose algoritmuose. Taip algoritmas naiviai įvertina jau išmestų kaladėlių prieinamumą. Tada iš sąrašo parenkama didžiausios vertės kaladėlė išmetimui. Algoritmo galutiniai rezultatai yra maksimali „rankos“ vertė ir kaladėlė išmetimui. Algoritmo pseudokodą galima pamatyti 5.11 pav.

Visi uždavyniai aprašytos 5.2 lentelėje.

Reikia pastebėti, kad tam tikrų *yaku* apribojimų nėra sudaryta dėl šių priežasčių:

- Trylika našlaičių – ypatingai retai pasitaikantis *yaku,* kuriam būtų reikalinga 13 papildomų skirtingų apribojimų
- Trys paslėpti trynukai, keturi paslėpti trynukai – suvaržymų uždavinio sprendėjas nežino, kurios kaladėlės gali būti paslėptos, t.y. pačios kaladėlės neturi informacijos apie jų slaptumą
- Trys ketvertai – kadangi ketvertų paskelbimas suteikia žaidėjui pakeitimo kaladėlė, *kan* skelbimas efektyviai padidina „rankos“ kaladėlių kiekį, o didesnis „rankos“ kaladėlių skaičius netinka suvaržymų uždavinio sprendėjui, kadangi padidėja sprendžiamų kintamųjų kiekis, o apribojimai yra statiniai

Žaidimo metu šio algoritmo žaidėjas kaladėles išmetimui parenka pagal šį algoritmą ir išsaugo gautą „rankos“ vertę. Šią „rankos“ vertę žaidėjas naudoja, kai tikrinama ar žaidėjui apsimoka „vogti“ priešininkų kaladėles. Žaidėjas algoritmui paduoda „ranką“ plius potencialią vogimui kaladėlę ir patikrina ar gauta nauja „rankos“ vertė yra didesnė nei per savo ėjimą gauta „rankos“ vertė. Jeigu gaunamas teigiamas rezultatas, kaladėlė „vagiama.“ Šiuos algoritmus galima pamatyti 5.12 pav.ir 5.13 pav.

5.1 lentelė. Visų apribojimų aprašymas

Apribojimo pavadinimas	**Apribotų kaladėlių kiekis**	**Apribojimo aprašymas**
uždara „ranka“	-	„Ranka“ turi būti uždara
pon	3	Kaladėlės turi sudaryti *pon*
chi	3	Kaladėlės turi sudaryti *chi*
pora	2	Kaladėlės turi sudaryti porą
kombinacija	3	Kaladėlės turi sudaryti kombinaciją (*pon, kan, chi*)
dviguba eilė	6	Kaladėlės turi sudaryti dvi identiškas eiles
paprastos kaladėlės	-	Kaladėlės turi būti paprastos (*ne drakonai, vėjai ar terminalai*)
drakonai	-	Kaladėlės turi būti *drakonai*
vėjai	-	Kaladėlės turi būti *vėjai*
vertingos kaladėlės	-	Kaladėlės turi būti *drakonai, raundo ar žaidėjo vėjai*
kilmingos kaladėlės	-	Kaladėlės turi būti *kilmingos kaladėlės*
terminalai	-	Kaladėlės turi būti *terminalai*
žalios kaladėlės	-	Kaladėlės turi būti žalios
chanta kombinacija	3	Kombinacijoje turi būti bent viena *kilminga kaladėlė arba terminalas*
chanta pora	2	Pora turi būti iš *kilmingų kaladėlių arba terminalų*
chanta „ranka“	-	„Rankoje“ turi būti bent po vieną *kilmingą kaladėlę, terminalą* ir paprastą kaladėlė
junchan kombinacija	3	Kombinacijoje turi būti bent vienas *terminalas*
sanshiki	9	Kaladėlės turi sudaryti 3 tris skirtingų rūšių vienodų akių eiles
ikkitsuukan	9	Kaladėlės turi sudaryti 3 vienos rūšies *chi* su akimis nuo 1 iki 9
sanshoku	9	Kaladėlės turi sudaryti 3 skirtingų rūšių vienodų akių *pon*
chiitoitsu	-	„Ranka“ turi būti sudaryta iš 7 skirtingų porų
honroutou	-	Kaladėlės turi būti tik iš *kilmingos kaladėlės arba terminalai*
honiisou	-	Kaladėlės turi būti tik vienos rūšies arba *kilmingos kaladėlės*
chinitsu	-	Kaladėlės turi būti tos pačios rūšies

5.2 lentelė. Visų *yaku* apribojimai

Apribojimas	**Apribojamos kaladėlės**
Visos paprastos *yaku*	
kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
paprastos kaladėlės	visos kaladėlės atskirai
Visos eilės *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3
chi	kaladėlė 4, kaladėlė 5, kaladėlė 6
chi	kaladėlė 7, kaladėlė 8, kaladėlė 9
chi	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
uždara „ranka“	-
Vienodos eilės *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3
chi	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
dviguba eilė	kaladėlė 1, kaladėlė 2, kaladėlė 3, kaladėlė 4, kaladėlė 5, kaladėlė 6
uždara „ranka“	-
Septynios poros *yaku*	
pora	kaladėlė 1, kaladėlė 2
pora	kaladėlė 3, kaladėlė 4
pora	kaladėlė 5, kaladėlė 6
pora	kaladėlė 7, kaladėlė 8
pora	kaladėlė 9, kaladėlė 10
pora	kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
chiitoitsu	visos kaladėlės

uždara „ranka“	-
Dvi vienodos eilės *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3
chi	kaladėlė 4, kaladėlė 5, kaladėlė 6
chi	kaladėlė 7, kaladėlė 8, kaladėlė 9
chi	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
dviguba eilė	kaladėlė 1, kaladėlė 2, kaladėlė 3, kaladėlė 4, kaladėlė 5, kaladėlė 6
dviguba eilė	kaladėlė 7, kaladėlė 8, kaladėlė 9, kaladėlė 10 kaladėlė 11, kaladėlė 12
uždara „ranka“	-
Vertingos kaladėlės *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
vertingos kaladėlės	kaladėlė 1, kaladėlė 2, kaladėlė 3
Terminalai* arba *kilmingos kaladėlės visose kombinacijose yaku	
chanta kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
chanta kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
chanta kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
chanta kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
chanta pora	kaladėlė 13, kaladėlė 14
chanta „ranka“	visos kaladėlės
Trys spalvotos eilės *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3
chi	kaladėlė 4, kaladėlė 5, kaladėlė 6
chi	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
sanshiki	kaladėlė 1, kaladėlė 2, kaladėlė 3, kaladėlė 4, kaladėlė 5, kaladėlė 6, kaladėlė 7, kaladėlė 8, kaladėlė 9
Ėilė *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3

chi	kaladėlė 4, kaladėlė 5, kaladėlė 6
chi	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
ikkitsuukan	kaladėlė 1, kaladėlė 2, kaladėlė 3, kaladėlė 4, kaladėlė 5, kaladėlė 6, kaladėlė 7, kaladėlė 8, kaladėlė 9
Visi trynukai *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
pon	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
Trys spalvoti trynukai *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
sanshoku	kaladėlė 1, kaladėlė 2, kaladėlė 3, kaladėlė 4, kaladėlė 5, kaladėlė 6, kaladėlė 7, kaladėlė 8, kaladėlė 9
Terminalai ir kilmingos kaladėlės yaku	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
pon	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
honroutou	visos kaladėlės atskirai
Trys maži drakonai *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
drakonai	kaladėlė 1, kaladėlė 2, kaladėlė 3
drakonai	kaladėlė 4, kaladėlė 5, kaladėlė 6

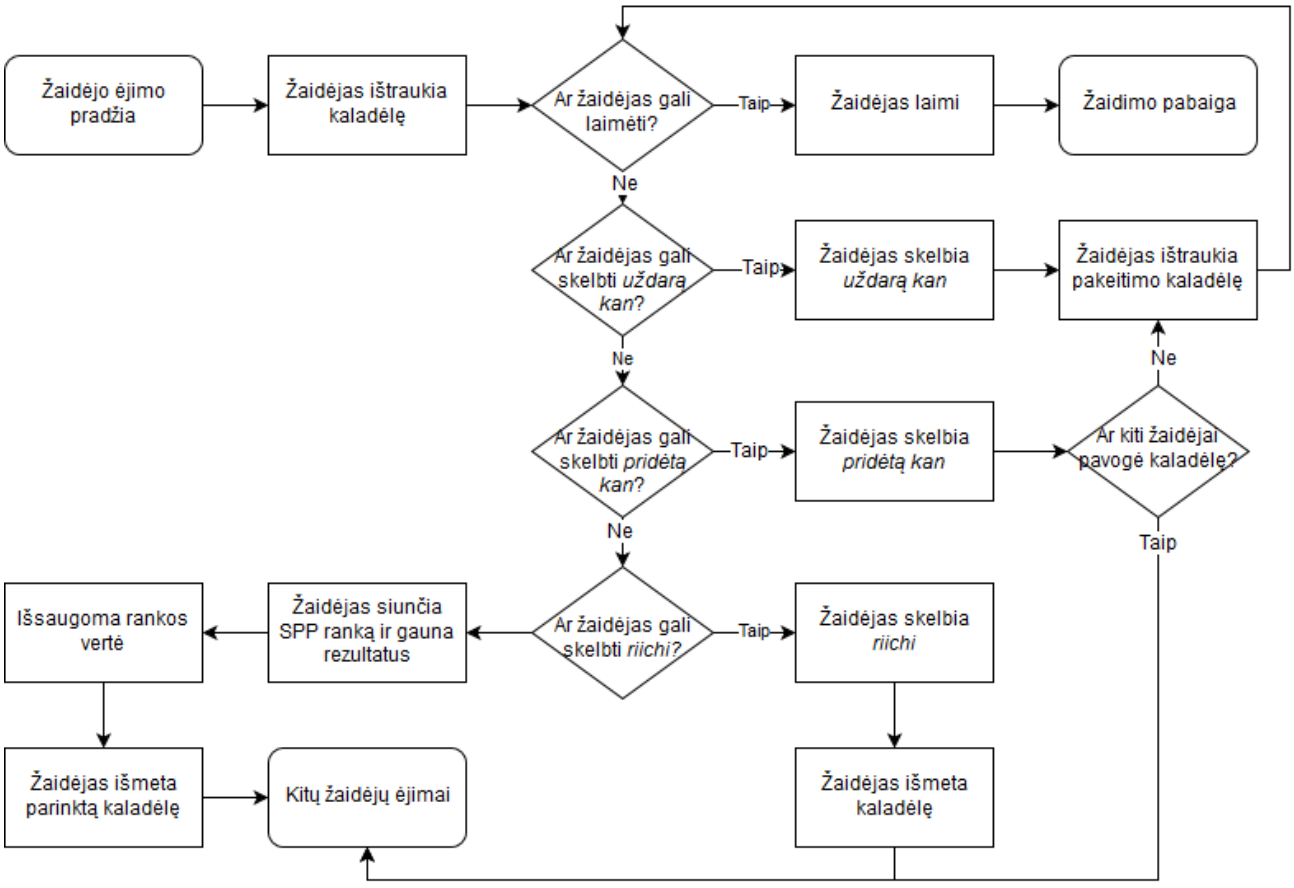
drakonai	kaladėlė 13, kaladėlė 14
Pusiau viena rūšis *yaku*	
kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
honiisou	visos kaladėlės
Terminalai kiekvienoje kombinacijoje *yaku*	
chi	kaladėlė 1, kaladėlė 2, kaladėlė 3
junchan kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
junchan kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
junchan kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
junchan kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
junchan pora	kaladėlė 13, kaladėlė 14
Viena rūšis *yaku*	
kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
chinitsu	visos kaladėlės
Trys dideli drakonai *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
drakonai	kaladėlė 1, kaladėlė 2, kaladėlė 3
drakonai	kaladėlė 4, kaladėlė 5, kaladėlė 6
drakonai	kaladėlė 7, kaladėlė 8, kaladėlė 9
Keturi maži *vėjai yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3

pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
vėjai	kaladėlė 1, kaladėlė 2, kaladėlė 3
vėjai	kaladėlė 4, kaladėlė 5, kaladėlė 6
vėjai	kaladėlė 7, kaladėlė 8, kaladėlė 9
vėjai	kaladėlė 13, kaladėlė 14
Keturi dideli *vėjai yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
pon	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
vėjai	kaladėlė 1, kaladėlė 2, kaladėlė 3
vėjai	kaladėlė 4, kaladėlė 5, kaladėlė 6
vėjai	kaladėlė 7, kaladėlė 8, kaladėlė 9
vėjai	kaladėlė 10, kaladėlė 11, kaladėlė 12
Visos kilmingos kaladėlės *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
pon	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
kilmingos kaladėlės	visos kaladėlės atskirai
Visi terminalai *yaku*	
pon	kaladėlė 1, kaladėlė 2, kaladėlė 3
pon	kaladėlė 4, kaladėlė 5, kaladėlė 6
pon	kaladėlė 7, kaladėlė 8, kaladėlė 9
pon	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
terminalai	visos kaladėlės atskirai
Visos žalios kaladėlės *yaku*	

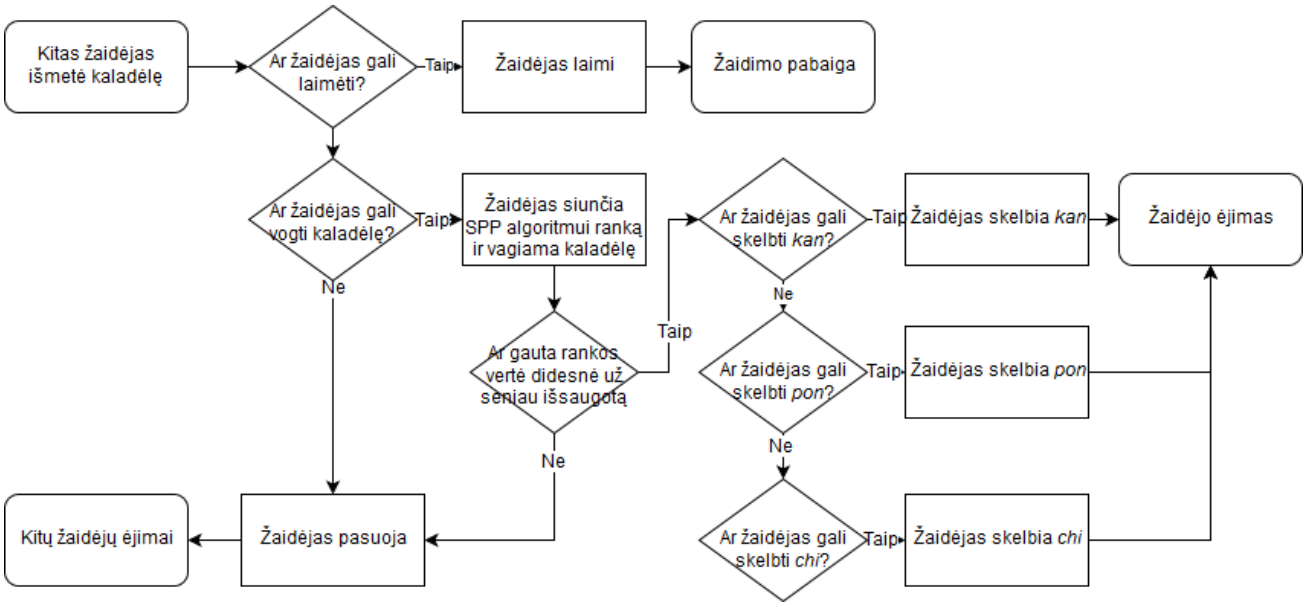
kombinacija	kaladėlė 1, kaladėlė 2, kaladėlė 3
kombinacija	kaladėlė 4, kaladėlė 5, kaladėlė 6
kombinacija	kaladėlė 7, kaladėlė 8, kaladėlė 9
kombinacija	kaladėlė 10, kaladėlė 11, kaladėlė 12
pora	kaladėlė 13, kaladėlė 14
žalios kaladėlės	visos kaladėlės atskirai

iš žaidimo gaunama žaidėjo „ranka“
inicijuojami rezultatų sąrašas *rez*
inicijuojamos uždaviniai pagal *yaku*
for uždavinys uždaviniuose **do**
    jeigu uždavinys uždaras, o „ranka“ atvira tęsiamas kito ciklo darbas
    sprendžiamas uždavinys ir gaunami rezultatai r_p
    paskaičiuojamas kaladėlių kiekis r_k, uždavinio „rankos“ vertė r_v, nereikalingos kaladėlės r_t
    r_k r_v r_t išsaugomi rezultatų sąraše *rez*
end for
iš rezultatų sąrašo išrenkami rezultatai su didžiausia $r_k + r_v$ verte
suskaičiuojami išrinktų rezultatų kaladėlių pasikartojimų kiekiai *k*
prie *k* pridedamos visų žaidėjų išmestų ir „pavogtų“ kaladėlių kiekiai
iš *k* išrenkama kaladėlė su didžiausia verte
gražinami rezultatai $r_k + r_v$ vertė ir išrinkta kaladėlė

5.11 pav. Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįsto algoritmo pseudokodas



5.12 pav. SPU žaidėjo ėjimo veiksmų algoritmo blokinė schema



5.13 pav. SPU žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė schema

5.3.6. Brutalios jėgos metodu pagrįstas algoritmas

Brutalios jėgos metodu pagrįstas algoritmas yra panašus į suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstą algoritmą. Brutalios jėgos metodo algoritmas naudoja „kibirus," kuriuose saugoma informacija apie kombinacijas. Vienas „kibiras" atitinka vieną kaladėlių kombinaciją. Šie „kibirai," kaip ir suvaržymų patenkinimo uždaviniai, turi apribojimus, kurie nusako kokios kaladėlės gali į kibirą įkristi. Vieną žaidimo *yaku* atitinkantis „kibirų" rinkinys dažniausiai sudarytas iš penkių „kibirų" – keturi „kibirai" kombinacijoms talpinti ir vienas „kibiras" porai. „Kibirų" apribojimai taip pat gali būti susiję tarpusavyje. Pvz. vienos rūšies *yaku* „kibirų" bendras apribojimas yra ta pati rūšis. Kaladėlėms įkrentant į „kibirą," „kibiro" ir su juo susijusių kitų „kibirų" apribojimai yra atnaujinami. Pvz. į tuščią „kibirą" be apribojimų įkritus *drakonui*, nauji „kibiro" apribojimai tampa *drakonai*, ir *pon*, kadangi eilių iš *drakonų* sudaryti negalima. Taip apribojamos kitos kaladėlės, kurios gali įkristi į „kibirą." „Kibirų" apribojimus galima rasti 5.3 lentelėje.

5.3 lentelė. „Kibirų" apribojimų aprašymas

„Kibiro" apibojimas	**Apribojimo aprašymas**
pon	„Kibire" gali būti tik *pon* kombinacija
chi	„Kibire" gali būti tik *chi* kombinacija
pora	„Kibire" gali būti tik pora
123	„Kibire" gali būti tik kaladėlės, kurių akių vertė yra 1, arba 2, arba 3
456	„Kibire" gali būti tik kaladėlės, kurių akių vertė yra 4, arba 5, arba 6
789	„Kibire" gali būti tik kaladėlės, kurių akių vertė yra 7, arba 8, arba 9
drakonai	„Kibire" gali būti tik *drakonai*
vėjai	„Kibire" gali būti tik *vėjai*
rutuliukai	„Kibire" gali būti tik *rutuliukai*
lazdelės	„Kibire" gali būti tik *lazdelės*

ženklai	„Kibire“ gali būti tik *ženklai*
ne *kilmingos kaladėlės*	„Kibire“ negali būti *kilmingų kaladėlių*
kilmingos kaladėlės	„Kibire“ gali būti tik *kilmingos kaladėlės*
vertingos kaladėlės	„Kibire“ negali būti vertingos kaladėlės
ne *terminalai*	„Kibire“ negali būti *terminalų*
terminalai	„Kibire“ gali būti tik *terminalai*
žalios kaladėlės	„Kibire“ gali būti tik žalios kaladėlės
rūšis arba *kilmingos kaladėlės*	„Kibire“ gali būti tik rūšis arba *kilmingos kaladėlės*
rutuliukai arba *kilmingos kaladėlės*	„Kibire“ gali būti tik *rutuliukai* arba *kilmingos kaladėlės*
lazdelės arba *kilmingos kaladėlės*	„Kibire“ gali būti tik *lazdelės* arba *kilmingos kaladėlės*
ženklai arba *kilmingos kaladėlės*	„Kibire“ gali būti tik *ženklai* arba *kilmingos kaladėlės*
kilmingos kaladėlės ar terminalai	„Kibire“ gali būti tik *kilmingos kaladėlės ar terminalai*
rūšies apribojimas susijęs su kitais „kibirais“	Visų susijusių „kibirų“ kaladėlės turi būti tik tos vienos rūšies
akių apribojimas susijęs su kitais „kibirais“	Visi susijusių „kibirų“ kaladėlės turi būti vienodų akių
skirtingų kaladėlių apribojimas susijęs su kitais „kibirais“	Visi susijusių „kibirų“ kaladėlės turi skirtis

Algoritmas pradedamas tuščių pagal *yaku* aprašymą apribotų „kibirų“ sudarymu. Tuščių kibirų sąrašą galima rasti 5.4 lentelė. Šie tušti „kibirai“ pridedami prie apdorojimui skirto „kibirų“ sąrašo. Tada algoritmas iš žaidimo gauna žaidėjo „ranką“ ir bando užpildyti „kibirus.“ Jeigu algoritmas randa kaladėlę, kuri telpa į „kibirą,“ algoritmas padaro tuo metu apdorojamų „kibirų“ kopijas ir į naują „kibirą“ patalpina kaladėlę. Po kaladėlės „kibire“ patalpinimo to „kibiro“ apribojimai yra atnaujinami. Nauji „kibirai“ pridedami prie apdorojimui skirto „kibirų“ sąrašo. Algoritmas toliau tęsia savo darbą bandydamas į „kibirus“ patalpinti kitas „rankos“ kaladėles.

Algoritmui baigus talpinti kaladėles, gautas apdorojimui skirtas „kibirų“ sąrašas apdorojamas. Iš šio sąrašo paimami rezultatai, kurių „kibiruose“ esančių kaladėlių skaičius plius *yaku* vertė, kurio atžvilgiu sudaryti „kibirai,“ yra didžiausi. Taip įvertinama ir „rankų“ *yaku* vertė ir kiekis kaladėlių, kiek trūksta iki pergalės. Prie išrinkto sąrašo tada pridėdami matomų kaladėlių kiekiai, panašiai kaip ir godžiuose euristiniuose algoritmuose. Taip algoritmas naiviai įvertina jau išmestų kaladėlių prieinamumą. Tada iš sąrašo parenkama didžiausios vertės kaladėlė išmetimui. Algoritmo galutiniai rezultatai yra maksimali „rankos“ vertė ir kaladėlė išmetimui. Šiuos metodus galima pamatyti 5.14 pav. ir 5.15 pav.

5.4 lentelė. „Kibirų“ pagal *yaku* aprašymus apribojimai

„Kibirai“	**„Kibiro“ apribojimas**
Visos paprastos *yaku*	
„Kibiras“ 1	ne *kilmingos kaladėlės,* ne *terminalai*
„Kibiras“ 2	ne *kilmingos kaladėlės,* ne *terminalai*
„Kibiras“ 3	ne *kilmingos kaladėlės,* ne *terminalai*
„Kibiras“ 4	ne *kilmingos kaladėlės,* ne *terminalai*

„Kibiras“ 5	ne *kilmingos kaladėlės,* ne *terminalai,* pora
Visos eilės *yaku*	
„Kibiras“ 1	*chi*
„Kibiras“ 2	*chi*
„Kibiras“ 3	*chi*
„Kibiras“ 4	*chi*
„Kibiras“ 5	pora
Vienodos eilės *yaku*	
„Kibiras“ 1	*chi*
„Kibiras“ 2	*chi*
„Kibiras“ 3	-
„Kibiras“ 4	-
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2
Susietų „kibirų“ apribojimai	rūšis, akys
Septynios poros *yaku*	
„Kibiras“ 1	pora
„Kibiras“ 2	pora
„Kibiras“ 3	pora
„Kibiras“ 4	pora
„Kibiras“ 5	pora
„Kibiras“ 6	pora
„Kibiras“ 7	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3, „Kibiras“ 4, „Kibiras“ 5, „Kibiras“ 6, „Kibiras“ 7
Susietų „kibirų“ apribojimai	rūšis, skirtingos kaladėlės
Dvi vienodos eilės *yaku*	
„Kibiras“ 1	*chi*
„Kibiras“ 2	*chi*
„Kibiras“ 3	*chi*
„Kibiras“ 4	*chi*
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2
Susietų „kibirų“ apribojimai	rūšis, akys

Susieti „kibirai“	„Kibiras“ 3, „Kibiras“ 4
Susietų „kibirų“ apribojimai	rūšis, akys
Vertingos kaladėlės *yaku*	
„Kibiras“ 1	*pon,* vertingos kaladėlės
„Kibiras“ 2	-
„Kibiras“ 3	-
„Kibiras“ 4	-
„Kibiras“ 5	pora
Trys spalvotos eilės *yaku*	
„Kibiras“ 1	*chi, rutuliukai*
„Kibiras“ 2	*chi, lazdelės*
„Kibiras“ 3	*chi, ženklai*
„Kibiras“ 4	-
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3
Susietų „kibirų“ apribojimai	akys
Ėilė *yaku*	
„Kibiras“ 1	*chi,* 123
„Kibiras“ 2	*chi,* 456
„Kibiras“ 3	*chi,* 789
„Kibiras“ 4	-
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3
Susietų „kibirų“ apribojimai	rūšis
Visi trynukai *yaku*	
„Kibiras“ 1	*pon*
„Kibiras“ 2	*pon*
„Kibiras“ 3	*pon*
„Kibiras“ 4	*pon*
„Kibiras“ 5	pora
Trys spalvoti trynukai *yaku*	
„Kibiras“ 1	*pon, rutuliukai*
„Kibiras“ 2	*pon, lazdelės*

„Kibiras“ 3	*pon, ženklai*
„Kibiras“ 4	-
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3
Susietų „kibirų“ apribojimai	akys
Terminalai ir kilmingos kaladėlės yaku	
„Kibiras“ 1	*pon, kilmingos kaladėlės ar terminalai*
„Kibiras“ 2	*pon, kilmingos kaladėlės ar terminalai*
„Kibiras“ 3	*pon, kilmingos kaladėlės ar terminalai*
„Kibiras“ 4	*pon, kilmingos kaladėlės ar terminalai*
„Kibiras“ 5	pora*, kilmingos kaladėlės ar terminalai*
Trys maži drakonai *yaku*	
„Kibiras“ 1	*pon, drakonai*
„Kibiras“ 2	*pon, drakonai*
„Kibiras“ 3	-
„Kibiras“ 4	-
„Kibiras“ 5	pora, *drakonai*
Pusiau viena rūšis *yaku*	
„Kibiras“ 1	rūšis arba *kilmingos kaladėlės*
„Kibiras“ 2	rūšis arba *kilmingos kaladėlės*
„Kibiras“ 3	rūšis arba *kilmingos kaladėlės*
„Kibiras“ 4	rūšis arba *kilmingos kaladėlės*
„Kibiras“ 5	pora, rūšis arba *kilmingos kaladėlės*
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3, „Kibiras“ 4, „Kibiras“ 5
Susietų „kibirų“ apribojimai	rūšis
Viena rūšis *yaku*	
„Kibiras“ 1	-
„Kibiras“ 2	-
„Kibiras“ 3	-
„Kibiras“ 4	-
„Kibiras“ 5	pora
Susieti „kibirai“	„Kibiras“ 1, „Kibiras“ 2, „Kibiras“ 3, „Kibiras“ 4, „Kibiras“ 5
Susietų „kibirų“ apribojimai	rūšis

Trys dideli drakonai *yaku*	
„Kibiras“ 1	*pon, drakonai*
„Kibiras“ 2	*pon, drakonai*
„Kibiras“ 3	*pon, drakonai*
„Kibiras“ 4	-
„Kibiras“ 5	pora
Keturi maži *vėjai yaku*	
„Kibiras“ 1	*pon, vėjai*
„Kibiras“ 2	*pon, vėjai*
„Kibiras“ 3	*pon, vėjai*
„Kibiras“ 4	-
„Kibiras“ 5	pora, *vėjai*
Keturi dideli *vėjai yaku*	
„Kibiras“ 1	*pon, vėjai*
„Kibiras“ 2	*pon, vėjai*
„Kibiras“ 3	*pon, vėjai*
„Kibiras“ 4	*pon, vėjai*
„Kibiras“ 5	pora
Visos kilmingos kaladėlės *yaku*	
„Kibiras“ 1	*pon, kilmingos kaladėlės*
„Kibiras“ 2	*pon, kilmingos kaladėlės*
„Kibiras“ 3	*pon, kilmingos kaladėlės*
„Kibiras“ 4	*pon, kilmingos kaladėlės*
„Kibiras“ 5	pora, *kilmingos kaladėlės*
Visi terminalai *yaku*	
„Kibiras“ 1	*pon, terminalai*
„Kibiras“ 2	*pon, terminalai*
„Kibiras“ 3	*pon, terminalai*
„Kibiras“ 4	*pon, terminalai*
„Kibiras“ 5	pora, *terminalai*
Visos žalios kaladėlės *yaku*	
„Kibiras“ 1	žalios kaladėlės
„Kibiras“ 2	žalios kaladėlės

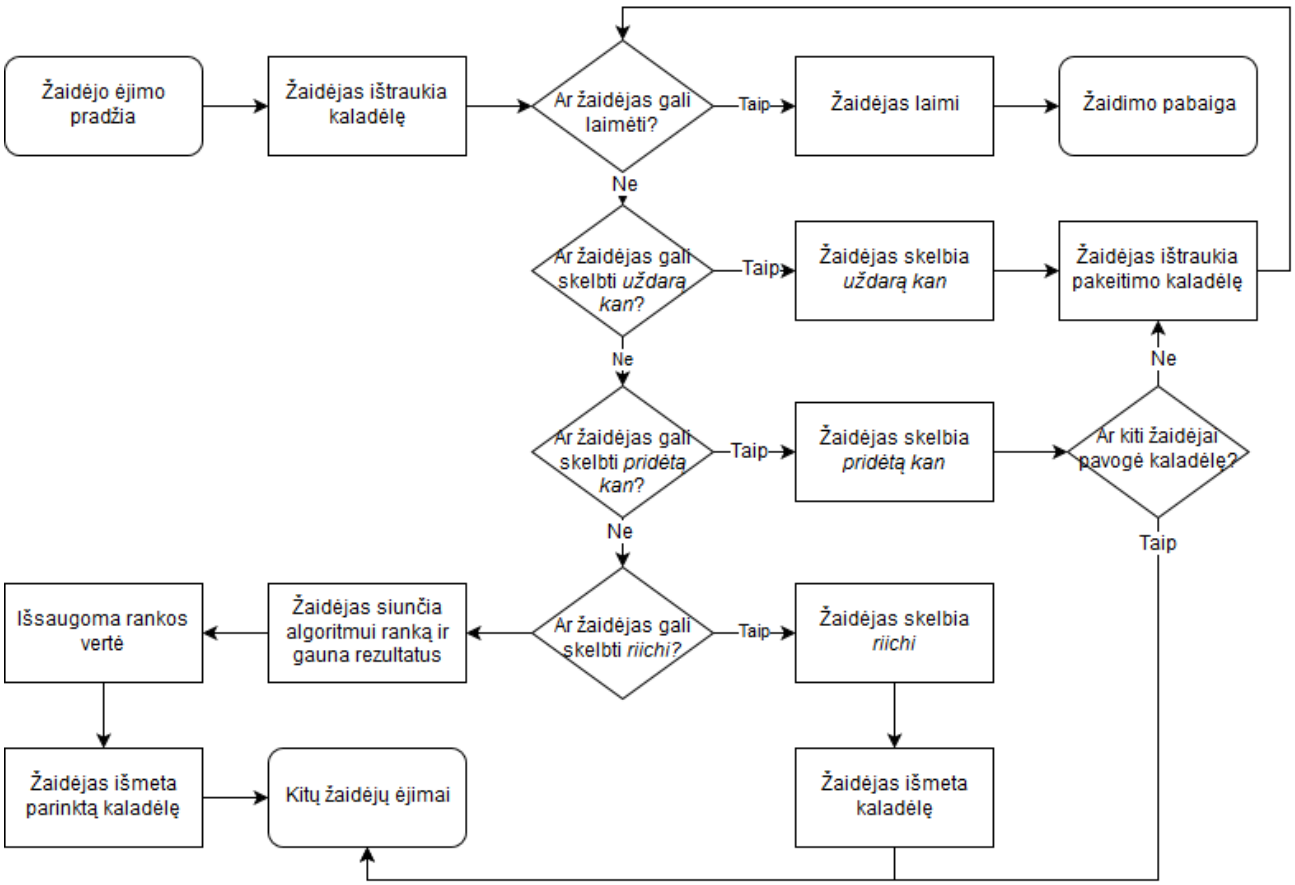
„Kibiras“ 3	žalios kaladėlės
„Kibiras“ 4	žalios kaladėlės
„Kibiras“ 5	pora, žalios kaladėlės

```
jeigu „ranka“ tuščia rezultatų sąraše išsaugoti gautus „kibirus“
nukopijuojame gautus „kibirus“ į naujus „kibirus“
iš „rankos“ paimame pirmą kaladėlę
for „kibiras“ najuose „kibiruose“ do
        jeigu kaladėlė telpa „kibire“
                į „kibira“ patalpiname kaladėlę
                rekursyviai kreipiamės į šitą algoritmą su „ranka,“ naujais „kibirais,“ rezultatų sąrašu
end for
rekursyviai kreipiamės į šitą algoritmą su „ranka,“ naujais „kibirais,“ rezultatų sąrašu
```

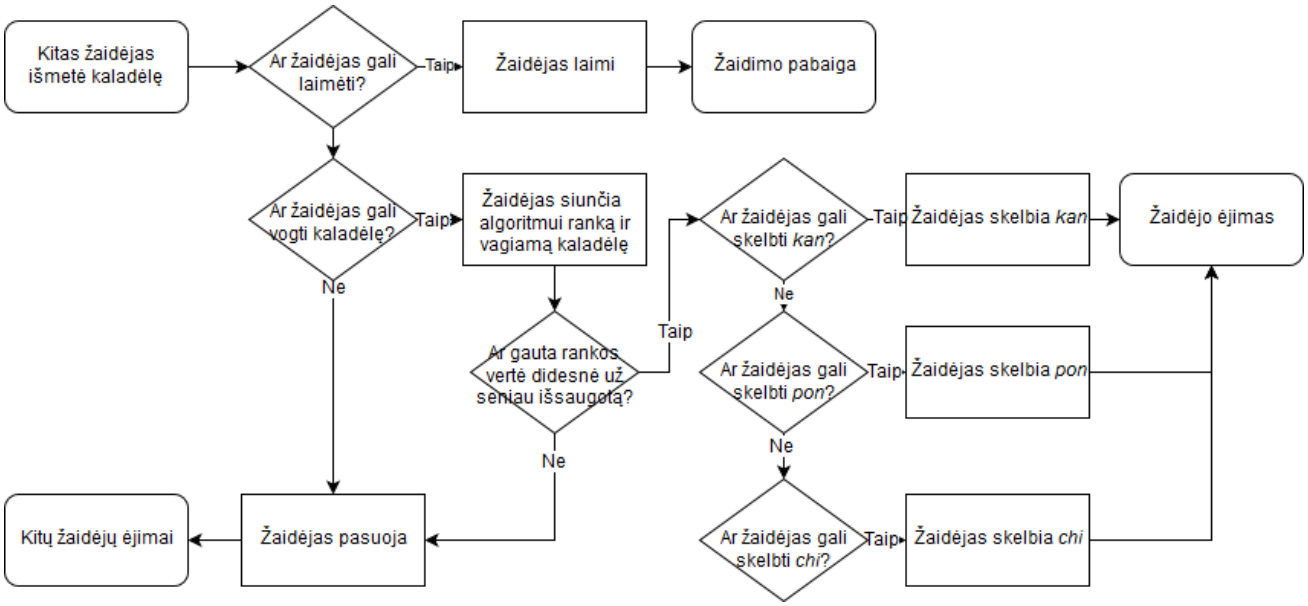
5.14 pav. Brutalios jėgos metodo „kibirų“ pildymo algoritmo pseudokodas

```
inicijuojami tušti pagal yaku aprašyti „kibirų“ rinkiniai
iš žaidimo gaunama žaidėjo „ranka“
inicijuojami rezultatų sąrašas rez
for kibirai „kibirų“ rinkiniai do
        siunčiame „rankos“, kibirų, rezultatų duomenis „kibirų“ pildymo algoritmui
        gaunami rezultatai r_p
        paskaičiuojamas kaladėlių kiekis r_k, „rankos“ vertė r_v, nereikalingos kaladėlės r_t
        r_k r_v r_t išsaugomi rezultatų sąraše rez
end for
iš rezultatų sąrašo išrenkami rezultatai su didžiausia r_k + r_v verte
suskaičiuojami išrinktų rezultatų kaladėlių pasikartojimų kiekiai k
prie k pridedamos visų žaidėjų išmestų ir „pavogtų“ kaladėlių kiekiai
iš k išrenkama kaladėlė su didžiausia verte
gražinami rezultatai r_k + r_v vertė ir išrinkta kaladėlė
```

5.15 pav. Brutalios jėgos metodo algoritmo pseudokodas



5.16 pav. Brutalios jėgos metodo žaidėjo ėjimo veiksmų algoritmo blokinė schema



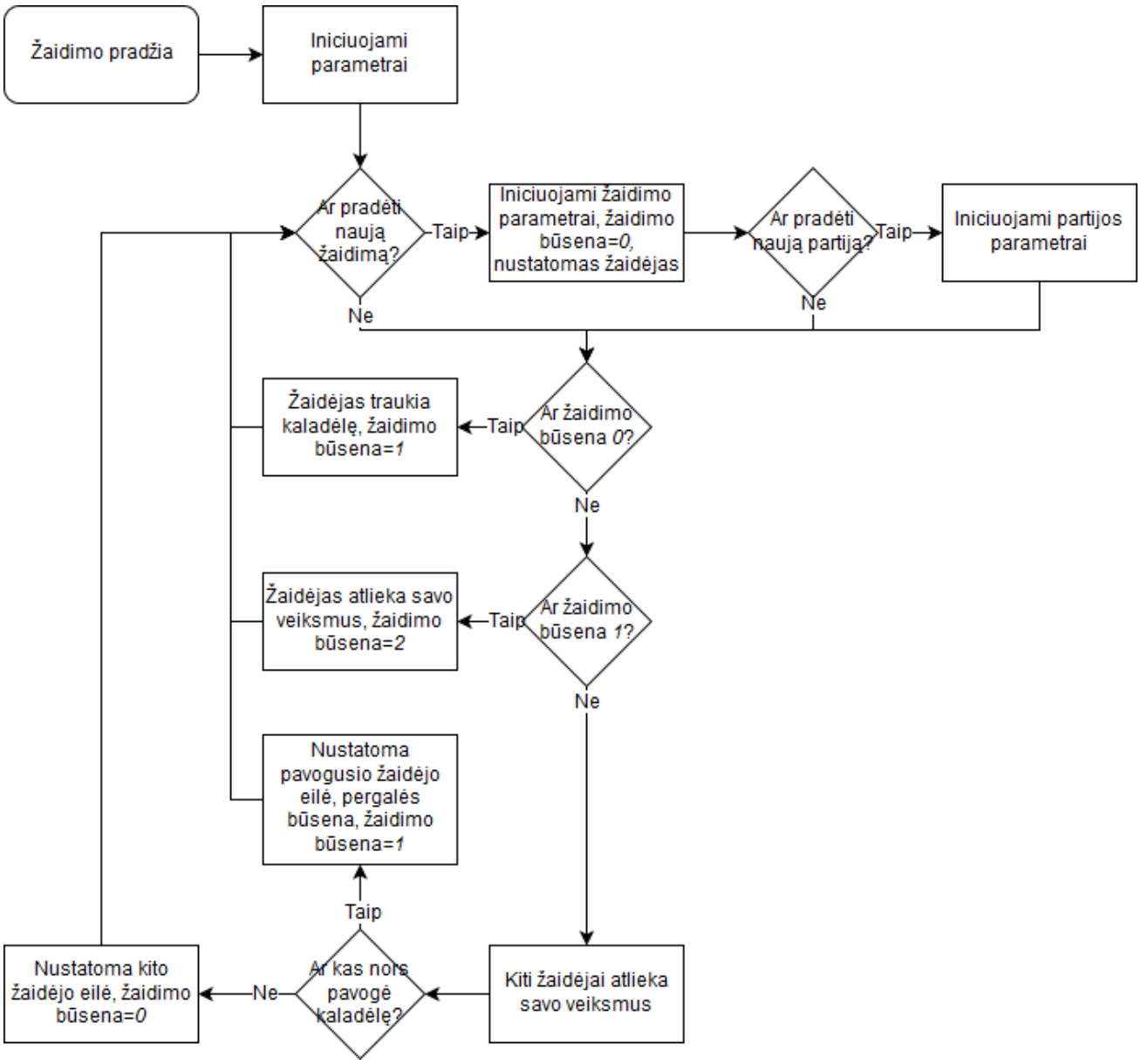
5.17 pav. Brutalios jėgos metodo žaidėjo veiksmų kito žaidėjo ėjimo metu algoritmo blokinė

6. EKSPERIMENTINĖ DALIS

6.1. Pasiūlytų „Mahjong“ žaidimo algoritmų realizacija ir testavimas

Visiems algoritmams įgyvendinti naudojama *Python* programavimo kalbos *Anaconda* duomenų mokslo platforma [12]. Ši platforma pasirinkta todėl, kad ši platforma yra nemokama, šioje platformoje jau yra sudiegta daug pagalbinių paketų, ir naudojant šią platformą naujus paketus yra lengviau įsidiegti negu naudojant paprastą *Python* distribuciją.

„Mahjong“ žaidimui realizuoti naudojama atviro kodo „Mahjong“ žaidimo *pyriichi* biblioteka [13]. Šią biblioteką reikėjo ištaisyti, kadangi joje buvo nemažai klaidų. Pagrindinį žaidimo ciklą galima pamatyti 6.1 pav.



6.1 pav. Pagrindinis žaidimo ciklo blokinė schema

Godus atsitiktinis algoritmas, godus euristinis algoritmas ir godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi realizuotas tiesiogiai *Python* kalba nenaudojant papildomų bibliotekų.

Giliu Q neuroniniu tinklu pagrįstas algoritmas realizuotas naudojant *Tensorflow* atviro kodo biblioteką [14]. Neuroninis tinklas apmokymus pradeda su Ɛ lygiu 1, t.y. rezultatai visą laiką yra sumaišomi. Tai yra atliekama tam, kad žaidimo pradžioje dar nėra pakankamai duomenų iš ko mokytis. Po 5000 apmokymo ciklų Ɛ per dar 5000 ciklų po truputį mažėja, kol pasiekia 0,05 vertę. Tai reiškia,

kad yra 5% tikimybė, kad rezultatai bus sumaišyti. Neuroninis tinklas buvo apmokytas sužaidžiant virš 140000 žaidimų. Taip pat apmokymams buvo panaudoti ir visų kitų algoritmų veiksmų duomenys.

Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstas algoritmas realizuotas naudojant *python-constrains* biblioteką. Šią biblioteką reikėta truputi pakoreguoti, kad minimalių konfliktų sprendėjas godžiai pasirinktų pirmas kintamųjų reikšmes. Taip pat ši biblioteka nesugeba panaudoti „džiokerinių“ kaladėlių kaip buvo planuota.

Problema randasi tame, kad sprendėjas supranta „džiokerinę“ kaladėlę, kaip vieną unikalią kaladėlę. Ši problema buvo išspręsta vietoj vienos, prie „rankos“ pridedant 8 „džiokerines“ kaladėles, žinant, kad „rankos“ tipas, kuriam trūksta 8 kaladėlių iki pergalės tikrai nebus geriausias sprendinys, kadangi egzistuoja septynių porų *yaku*, kuriam iki pergalės minimalus trūkstančių kaladėlių kiekis yra 7 kaladėlės.

Taip pat sprendėjas nesugeba minimizuoti „džiokerinių“ kaladėlių. Funkcija, kuri tikrina „džiokerinių“ kaladėlių kiekį gražina tik reikšmę, kuri nurodo ar sprendime yra mažiau „džiokerinių“ kaladėlių nei nustatytas skaičius ar ne. Todėl „džiokerinių“ kaladėlių minimizavimui panaudota ciklas, kai nustatytas „džiokerių“ skaičius mažinamas ir bandoma išspręsti uždavinį. Pavyzdžiui, pirmiausia nustatoma, kad sprendime turi būti ne daugiau aštuonios „džiokerinės“ kaladėlės. Jeigu gaunamas tokios užduoties sprendimas, sprendime galimų „džiokerinių“ kaladėlių suma sumažinama vienetu. Jeigu vis dar egzistuoja sprendimas, sprendime galimų „džiokerinių“ kaladėlių suma sumažinama dar vienetu ir t.t. Modifikuotą algoritmą galima pamatyti 6.2 pav.

Reikia pastebėti, kad jeigu apribojimas riboja keletą kaladėlių, ir viena iš tų kaladėlių neatitinka apribojimo, visos ribojamos kaladėlės neatitinka apribojimų. Jeigu sprendėjas neišsprendžia uždavinio per 2000 žingsnių, nutariama, kad šis uždavinys neišsprendžiamas. Jeigu sprendėjui būtų leistina daugiau žingsnių algoritmo veikimas užtruktų daugiau laiko, o algoritmo veikimo laikas ir taip užtrunka apie 10 sekundžių.

Visi 5.1 lentelė randami apribojimai realizuoti parašant po funkciją, kuri atitinka tą apribojimą.

iš žaidimo gaunama žaidėjo „ranka“
inicijuojami rezultatų sąrašas *rez*
inicijuojamos uždaviniai pagal *yaku*
for uždavinys uždaviniuose **do**
    jeigu uždavinys uždaras, o „ranka“ atvira tęsiamas kito ciklo darbas
    for „džiokerių“ leistinas kiekis nuo 8 iki 0 **do**
        sprendžiamas uždavinys
        jeigu gaunami rezultatai *r,* rezultatai užsirašomi į r_p
        kitu atveju stabdomas šio ciklo veikimas
    end fo
    jeigu yra rezultatai r_p
        paskaičiuojamas kaladėlių kiekis r_k, uždavinio „rankos“ vertė r_v, nereikalingos kaladėlės r_t
        r_k r_v r_t išsaugomi rezultatų sąraše *rez*
end for
iš rezultatų sąrašo išrenkami rezultatai su didžiausia $r_k + r_v$ verte
suskaičiuojami išrinktų rezultatų kaladėlių pasikartojimų kiekiai *k*
prie *k* pridedamos visų žaidėjų išmestų ir „pavogtų“ kaladėlių kiekiai
iš *k* išrenkama kaladėlė su didžiausia verte
gražinami rezultatai $r_k + r_v$ vertė ir išrinkta kaladėlė

6.2 pav. Modifikuotas SPU algoritmo pseudokodas

Brutalios jėgos algoritmas realizuotas naudojant objektus. „Kibirui“ sukurta klasė *bucket*, kuri turi tokius duomenis:

```
class Bucket:
      def __init__(self, constrains, links = []):
            self.constrains = constrains
            self.tiles = []
            self.tile_count = 0
```

```
        self.suit = None
        self.linked = []
        self.links = links
        self.locked = False
```

Čia *constrains* kintamajame talpinami „kibiro“ apribojimai, *tiles* talpina kaladėlių sąrašą, *tile_count* yra „kibire“ randamų kaladėlių kiekis, *suit* – „kibire“ patalpintų kaladėlių rūšis, *linked* – nurodo su kuriais kitais „kibirais“ šitas kibiras yra susietas, *links* – nurodo, kokiais apribojimais „kibirai“ yra susieti, *locked* – nurodo, kad „kibiras“ yra užrakintas, jo keisti negalima.

Taip pat ši klasė turi keletą metodų:

```
        def fits(self, tile)
        def fill(self, tile)
        def within_chi_range(self, tile)
        def load_state(self, other_bucket)
```

Čia *fits* metodas atsako, ar nurodyta kaladėlė *tile* telpa į „kibirą“ ir atitinka visus apribojimus. *Fill* metodas patalpina kaladėlę *tile* į „kibirą“ ir atnaujina „kibiro“ bei kitų, su šiuo „kibiru“ susietu, „kibirų“ apribojimus. W*ithin_chi_range* yra pagalbinė funkcija, kuri pasako ar kaladėlė *tile* yra eilės galimoje srityje lyginant su jau „kibire“ esančiomis kaladėlėmis, *load_state* nukopijuoja visus duomenis iš kito „kibiro.“

Testavimo metu buvo išsiaiškinta, kad brutalios jėgos metodo įvykdymas užtrunka ilgai. Kai kurių „kibirų“ rinkinių sprendimas užtrukdavo iki 3 min. realaus laiko, nes didesnės laisvės turintys „kibrų“ rinkiniai gali duoti virš milijono rezultatų. Buvo bandyta nenaudoti rekursinio algoritmo (6.3 pav.), kad sutaupyti procesoriaus resursų, bet rezultatai žymiai nepagerėjo, todėl buvo nuspręsta šio algoritmo nenaudoti.

Eksperimentų rezultatai pateikiami *logs* katalogo tekstiniuose failuose.

```
iš žaidimo gaunama žaidėjo „ranka,“ seni kibirai ir „super-kibiras“ informacijai gražinti
nukopijuojame senus „kibirus“ į naujus „kibirus“
į „super-kibirą“ įdedame naujus „kibirus“
for kaladėlė žaidėjo „rankoje“ do
        inicijuojame tuščią naujų „kibirų“ sąrašą
        for „kibirai“ „super-kibire“ do
                for „kibiras“ „kibiruose“ do
                        jeigu kaladėlė telpa „kibire“
                                nukopijuojame „kibirus“ į tusčius laikinus „kibirus“
                                į laikinų „kibirų“ atitinkantį „kibirą“ patalpiname kaladėlę
                                prie naujų „kibirų“ sąrašo pridedame laikinus „kibirus“
                end for
        end for
        prie „super-kibiro“ pridedame naujų „kibirų“ sąrašą
end for
gražiname „super-kibirą“
```

6.3 pav. Modifikuotas brutalios jėgos metodo „kibirų“ pildymo algoritmo pseudokodas

6.2. Eksperimentų planavimas

Eksperimentai atlikti naudojant jau realizuotus metodus „Mahjong“ žaidimui žaisti. Algoritmams žaidžiant išsaugoma informacija apie žaidėjų pergalės, lygybes, galinius partijos taškus. Žaidėjai partijos metu vietomis nesikeičia. Prasidedant naujai partijai, žaidėjų vietos atsitiktinai sumaišomos. Kadangi žaidimą gali žaisti keturi žaidėjai, o algoritmų yra penki (neskaičiuojant brutalios jėgos metodo, kadangi šio algoritmo veikimas labai lėtas), suplanuoti keli eksperimentai su skirtingais žaidėjais.

Pirmo eksperimento žaidėjai:

- Godus euristinis algoritmas
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi
- Giliu Q neuroniniu tinklu pagrįstas algoritmas
- Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstas algoritmas

Kadangi SPU žaidėjas ėjimus apsvarstyti užtrunka daugiau laiko, žaidimų sužaisti nesitikima daug.

Antro eksperimento žaidėjai:

- Godus atsitiktinis algoritmas
- Godus euristinis algoritmas
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi
- Giliu Q neuroniniu tinklu pagrįstas algoritmas

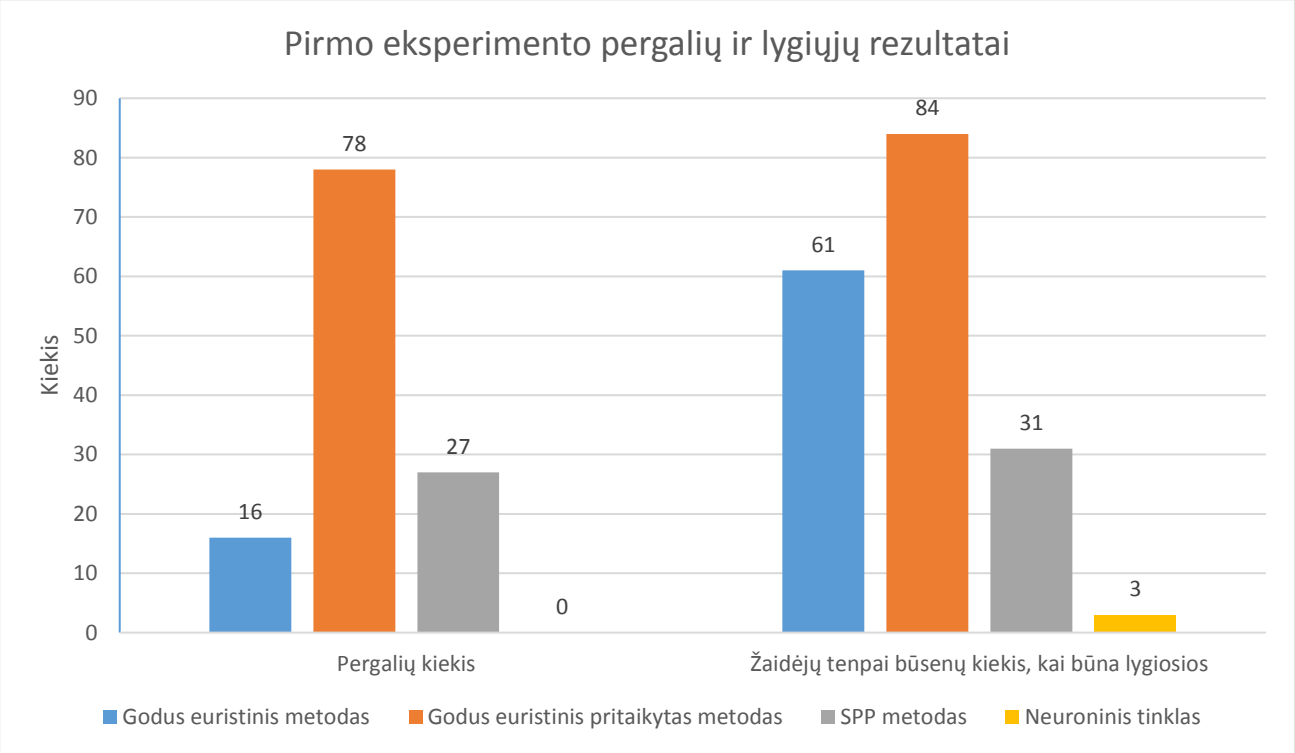
Trečio eksperimento žaidėjai:

- Godus euristinis algoritmas
- Godus euristinis algoritmas
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi
- Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi

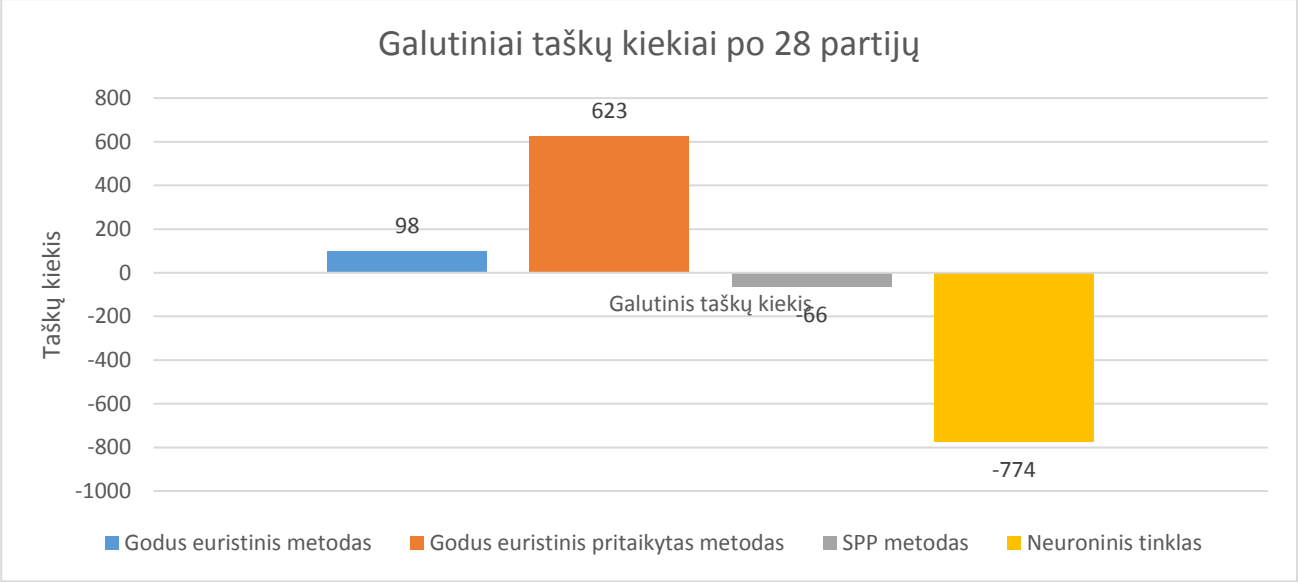
6.3. Atlikti eksperimentai ir jų rezultatai

6.3.1. Pirmas eksperimentas

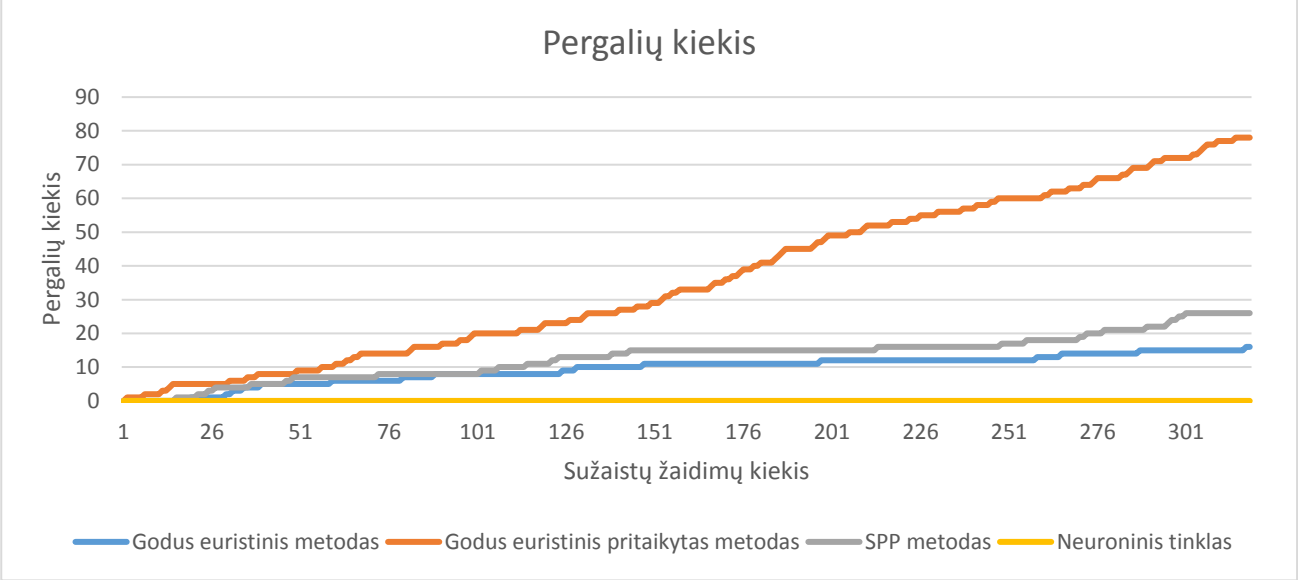
Pirmo eksperimento rezultatai po per 28 partijas sužaistų 320 žaidimų.



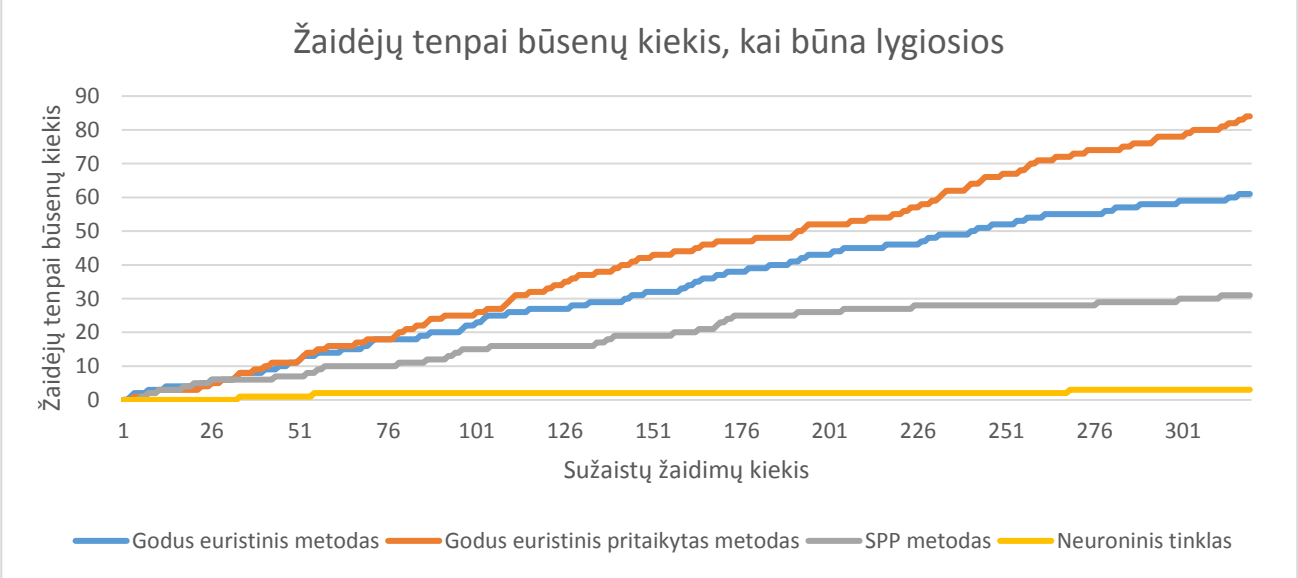
6.4 pav. Pirmo eksperimento žaidėjų rezultatai



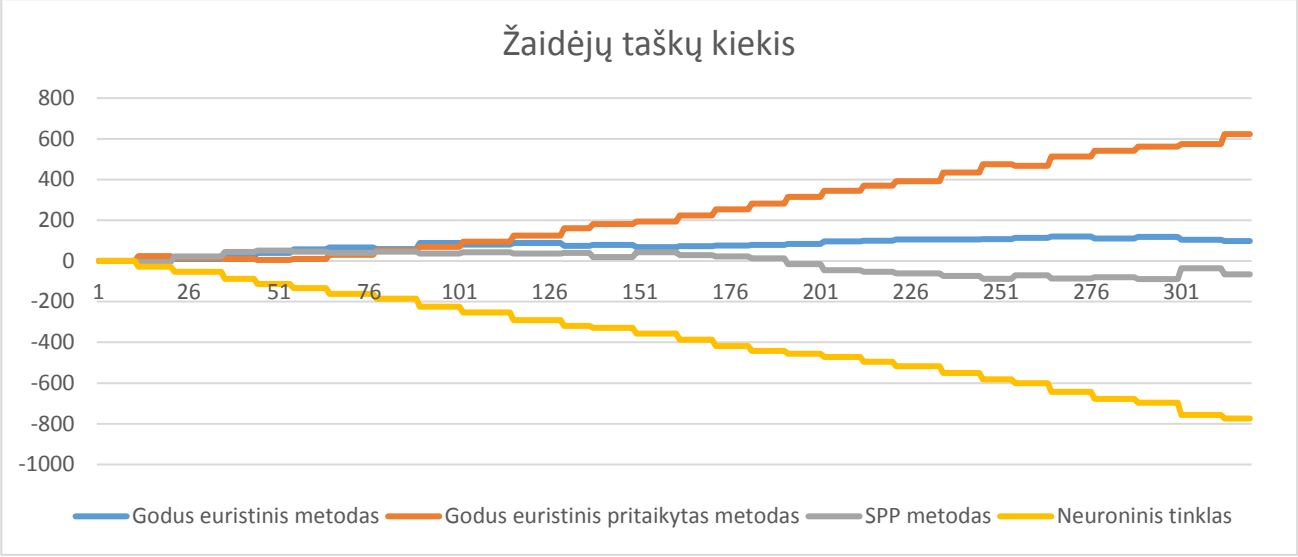
6.5 pav. Pirmo eksperimento žaidėjų galutiniai taškai



6.6 pav. Pirmo eksperimento žaidėjų pergalės



6.7 pav. Pirmo eksperimento žaidėjų *tenpai* būsenų kiekis, kai būna lygiosios



6.8 pav. Pirmo eksperimento žaidėjų taškai

Iš pirmo eksperimento rezultatų (6.4 pav., 6.5 pav., 6.6 pav., 6.7 pav., 6.8 pav.) matoma, kad godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi yra efektyviausias iš eksperimente dalyvavusių metodų. Taip pat įdomu paminėti, kad nors godus euristinis metodas laimėjo mažiau negu suvaržymų patenkinimo uždavinio minimalių konfliktų metodas, godaus euristinio metodo daugiau pasiektų *tenpai* lygiųjų būsenose kiekis, nulėmė, kad šis metodas užėmė antrą vietą pagal galutinių taškų kiekį. Neuroninio tinklo žaidėjas nesugebėjo laimėti nė vieno žaidimo.

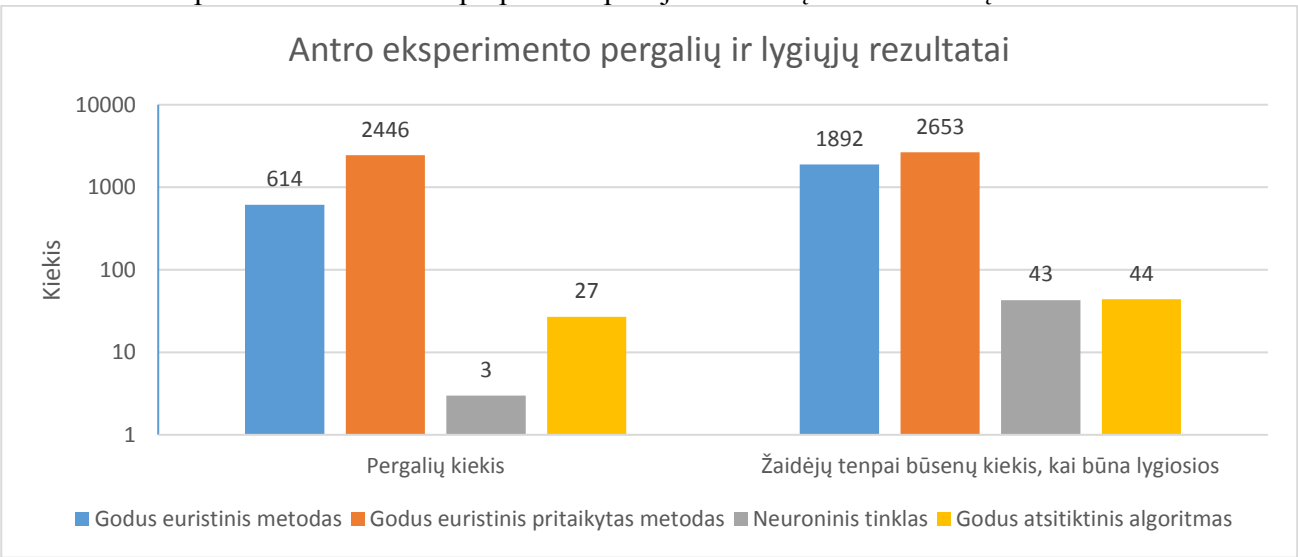
Įdomi statistika, kad suvaržymų patenkinimo uždavinio minimalių konfliktų metodo pergalių santykis su iš viso žaidėjo pasiektais *tenpai* yra artimas godaus euristinio pritaikyto metodo santykiui. Šis rodiklis nurodo, kad kai SPU metodas pasiekia *tenpai*, šis metodas laimi apie 47% tų žaidimų, kai tuo tarpu efektyvesnis godus euristinis pritaikytas metodas laimi apie 48% *tenpai* „rankų.“ Tai nurodo, kad SPU metodas, neteisingai pasirenka, kokios „rankos“ siekti. Reikėtų pakoreguoti SPU metodo „rankų“ vertes, kad šis metodas dažniau pasirinktų greitas, lengviau išvystomas „rankas.“ Tuo tarpu godus euristinis metodo pergalių ir *tenpai* santykis yra tik apie 21%. Tai parodo, kad godus euristinis metodas godžiai „vogdamas“ kaladėles dažnai susigadina savo „ranką“ ir nebegali laimėti. Šiuos duomenis galima pamatyti 6.1 lentelė.

6.1 lentelė. Pirmo eksperimento rezultatai

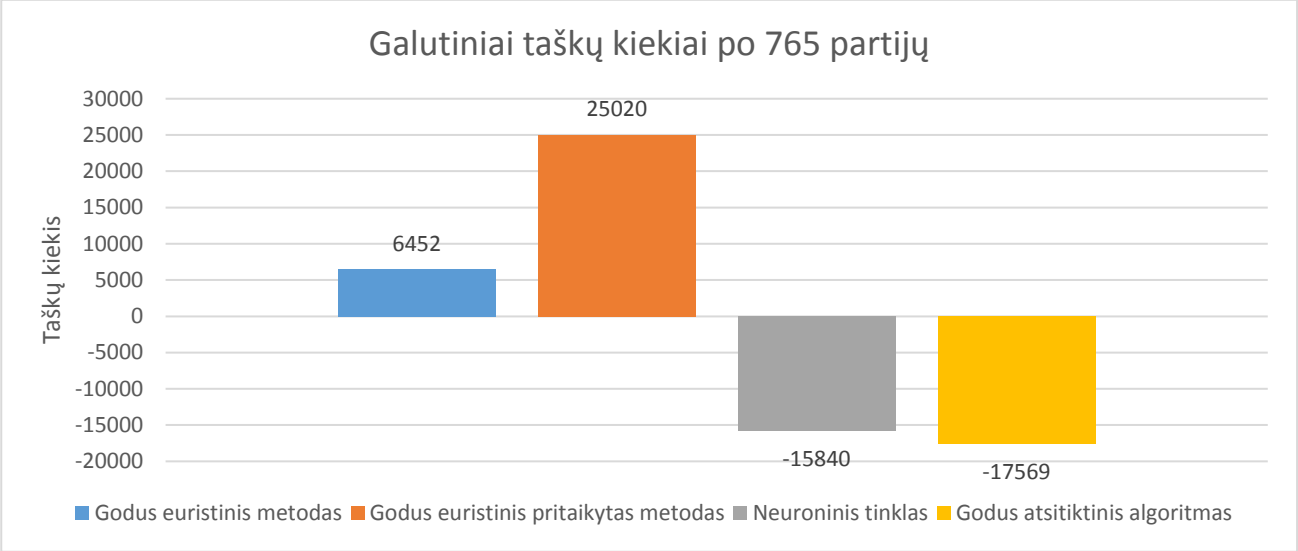
	Godus euristinis metodas	**Godus euristinis pritaikytas metodas**	**SPU metodas**	**Neuroninis tinklas**
Pergalės	13.22%	64.46%	22.31%	0.00%
Pergalės pagal žaidimų kiekį	5.00%	24.38%	8.44%	0.00%
Lygiosios	34.08%	46.93%	17.32%	1.68%
Lygiosios pagal žaidimų kiekį	19.06%	26.25%	9.69%	0.94%
Pergalių, iš viso žaidėjo pasiektų *tenpai* santykis	20.78%	48.15%	46.55%	0.00%
Uždirbtų partijos taškų vidurkis	3.5	22.25	-2.36	-27.64

6.3.2. Antras ekperimentas

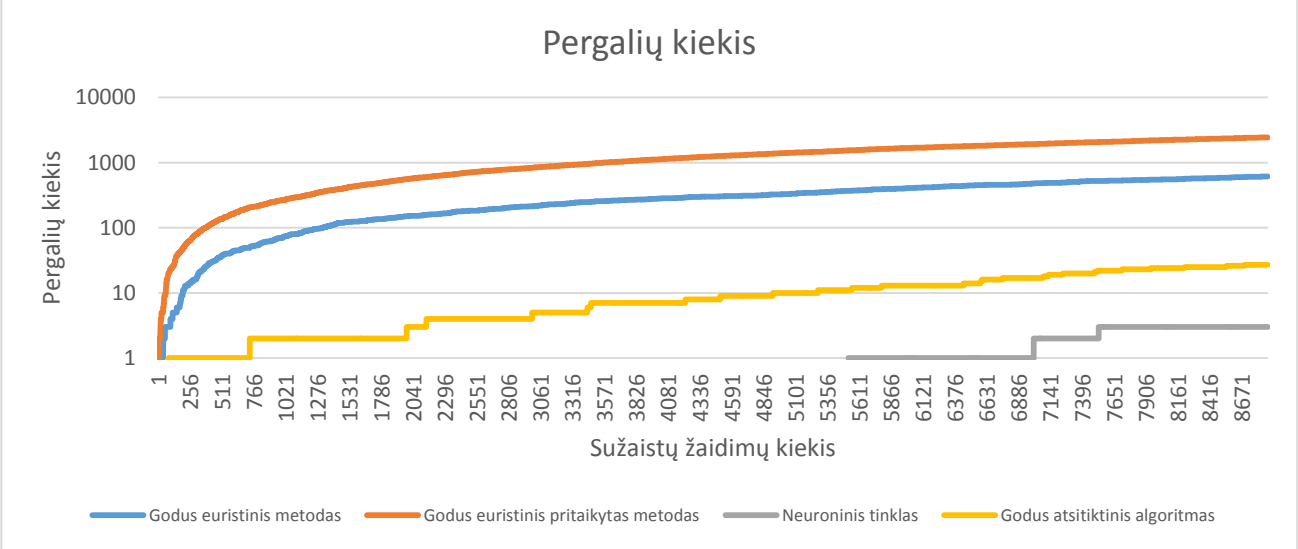
Antro eksperimento rezultatai po per 765 partijas sužaistų 8882 žaidimų.



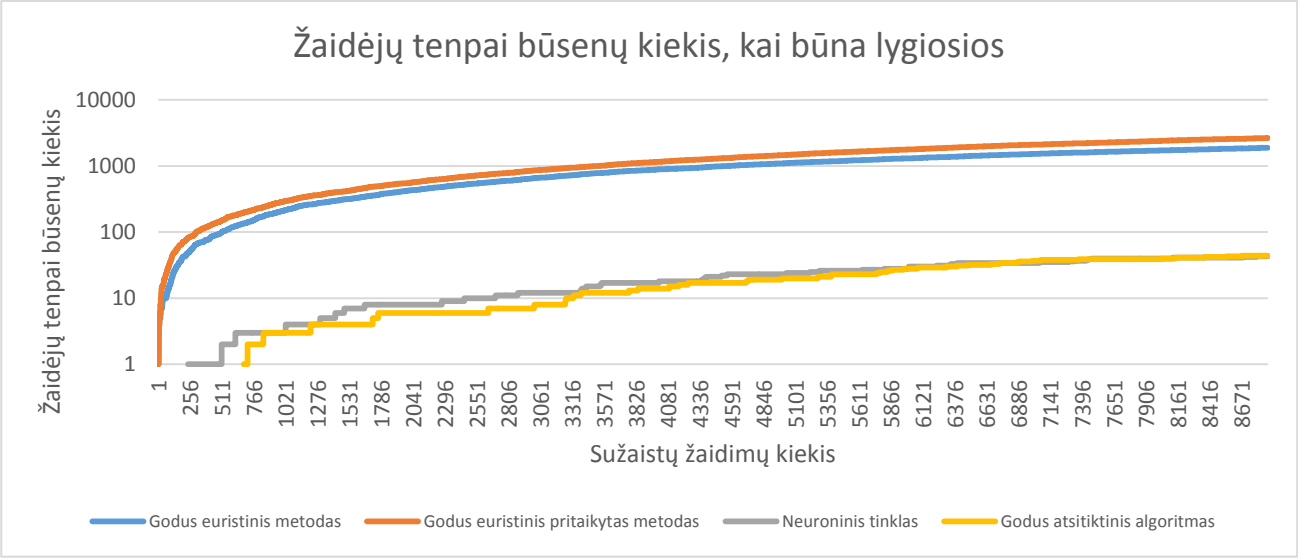
6.9 pav. Antro eksperimento žaidėjų rezultatai



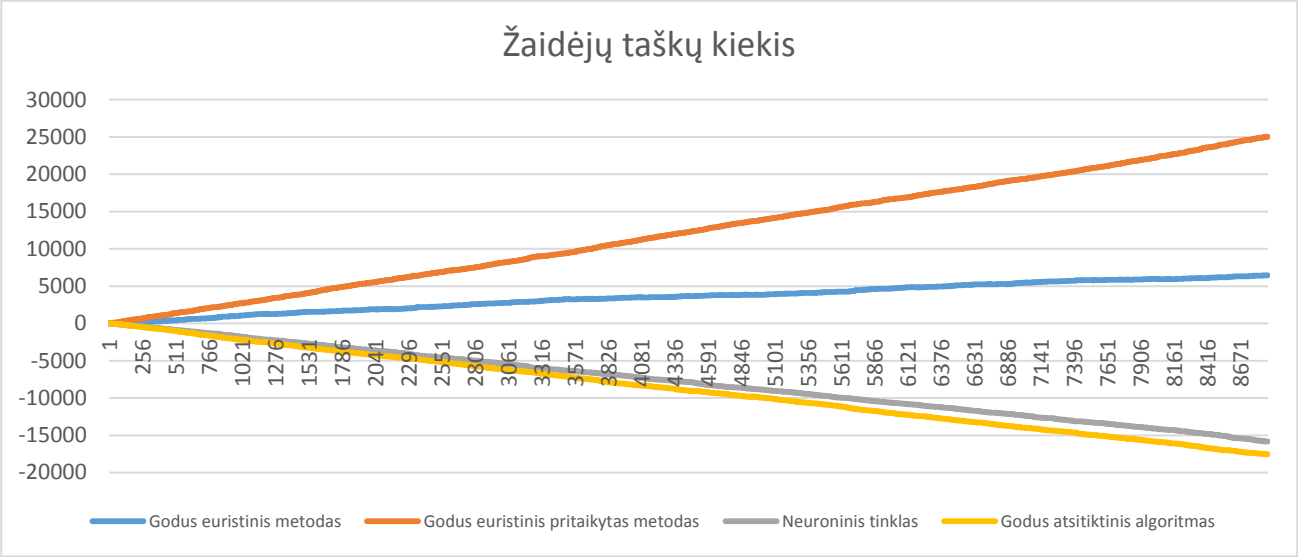
6.10 pav. Antro eksperimento žaidėjų galutiniai taškai



6.11 pav. Antro eksperimento žaidėjų pergalės



6.12 pav. Antro eksperimento žaidėjų *tenpai* būsenų kiekis, kai būna lygiosios



6.13 pav. Antro eksperimento žaidėjų taškai

Iš antro eksperimento rezultatų (6.9 pav., 6.10 pav., 6.11 pav., 6.12 pav., 6.13 pav.) matoma, kad godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi vis tiek yra efektyviausias iš eksperimente dalyvavusių metodų. Taip pat įdomu paminėti, kad neuroninis tinklas savo žaidimu nelabai skiriasi nuo godaus atsitiktinio algoritmo. Nors neuroninis tinklas laimėjo mažiau žaidimų ir pasiekė mažiau *tenpai* būsenų esant lygiosioms, neuroninis tinklas taškų prarado mažiau nei godus atsitiktinis algoritmas. Tai reiškia, kad neuroninis tinklas nepralaimi tiek daug taškų tiesiogiai kitam žaidėjui, t.y. neuroninis tinklas ne taip dažnai išmeta kitų žaidėjų laiminčias kaladėles. Neuroninis tinklas yra šiek tiek efektyvesnis nei godus atsitiktinis algoritmas

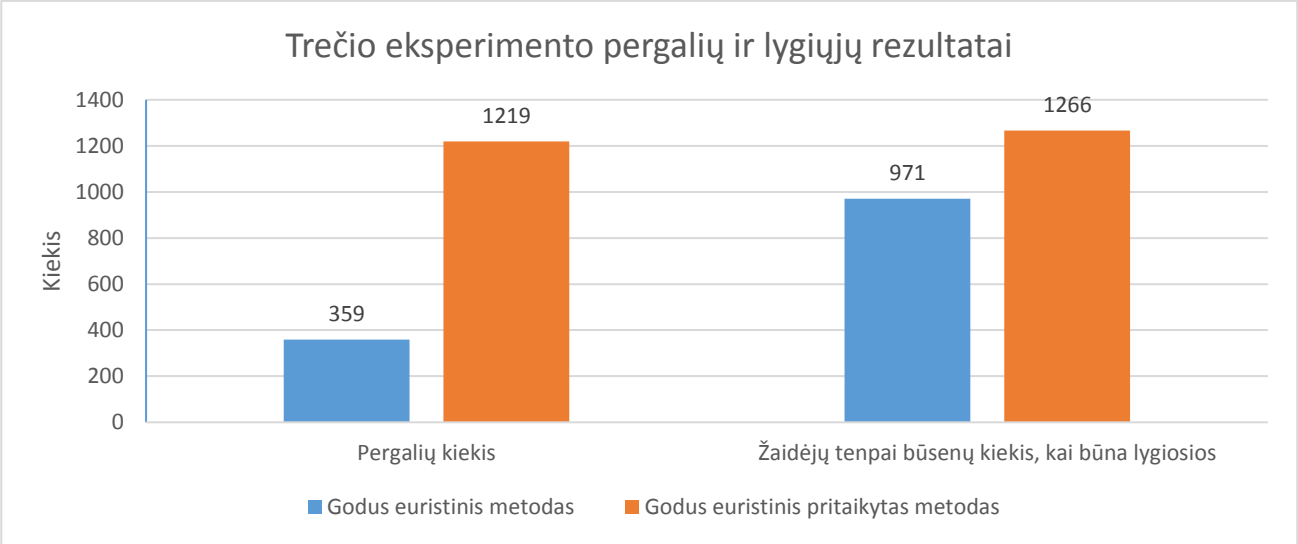
Antras eksperimentas parodo (6.2 lentelė), kad abiejų godžių euristinių metodų pergalių kiekis padidėjo palyginus su pirmu eksperimentu apie 5%, nors bendras pergalių skaičius sumažėjo apie 3%. Tai nurodo, kad pirmo eksperimento tuos 5% žaidimų laimėdavo SPU algoritmas. Godaus euristinio metodo pergalių ir *tenpai* santykis padidėjo apie 4%, kai tuo tarpu godaus euristinio pritaikyto metodo santykis ryškiai nepakito (pokytis mažesnis nei 0,2%). Įdomu pastebėti, kad godaus atsitiktinio algoritmas pergalių ir *tenpai* santykis yra didesnis nei godaus euristinio metodo.

6.2 lentelė. Antro eksperimento rezultatai

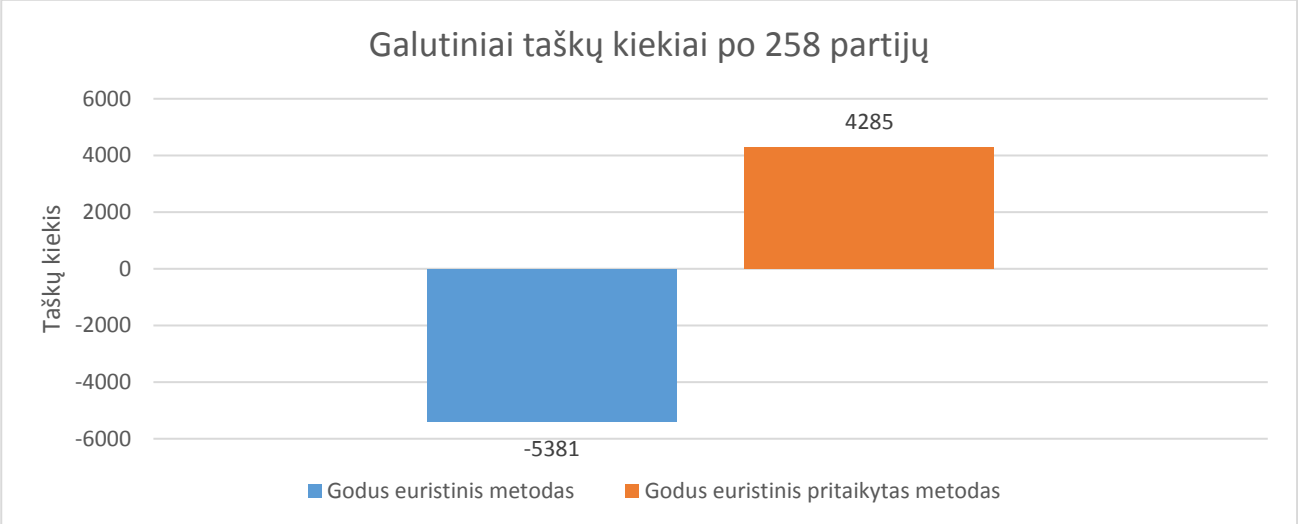
	Godus euristinis metodas	**Godus euristinis pritaikytas metodas**	**Neuroninis tinklas**	**Godus atsitiktinis algoritmas**
Pergalės	19.87%	79.16%	0.10%	0.87%
Pergalės pagal žaidimų kiekį	6.91%	27.54%	0.03%	0.30%
Lygiosios	40.85%	57.28%	0.93%	0.95%
Lygiosios pagal žaidimų kiekį	21.30%	29.87%	0.48%	0.50%
Pergalių, iš viso žaidėjo pasiektų *tenpai* santykis	24.50%	47.97%	6.52%	38.03%
Uždirbtų partijos taškų vidurkis	8.433987	32.70588	-20.7059	-22.966

6.3.3. Trečias eksperimentas

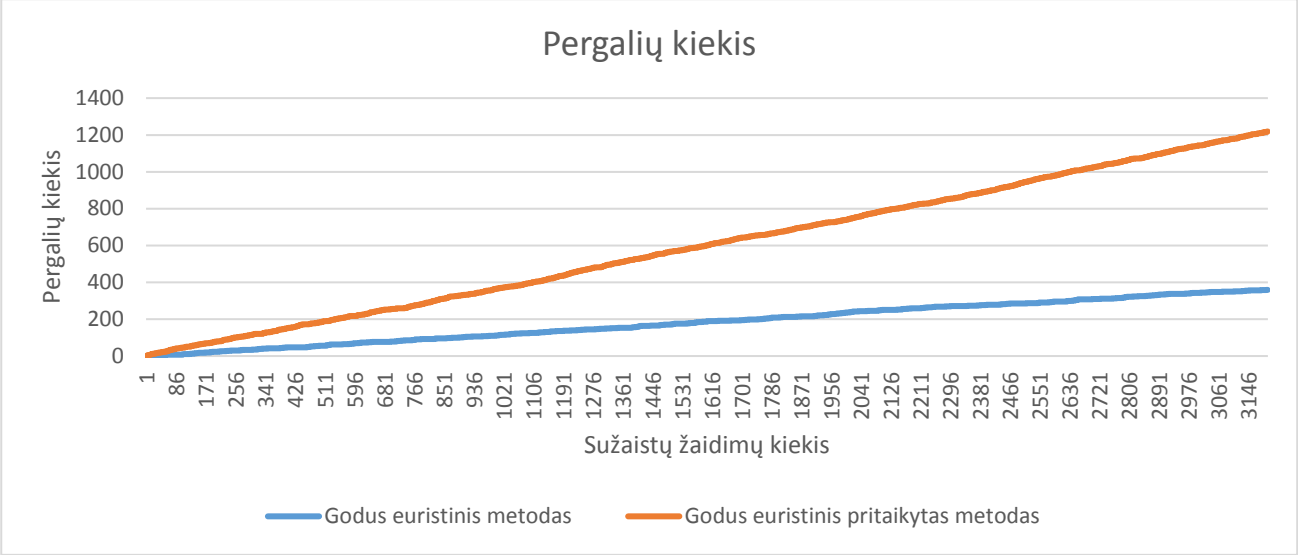
Trečio eksperimento rezultatai po per 258 partijas sužaistų 3202 žaidimų.



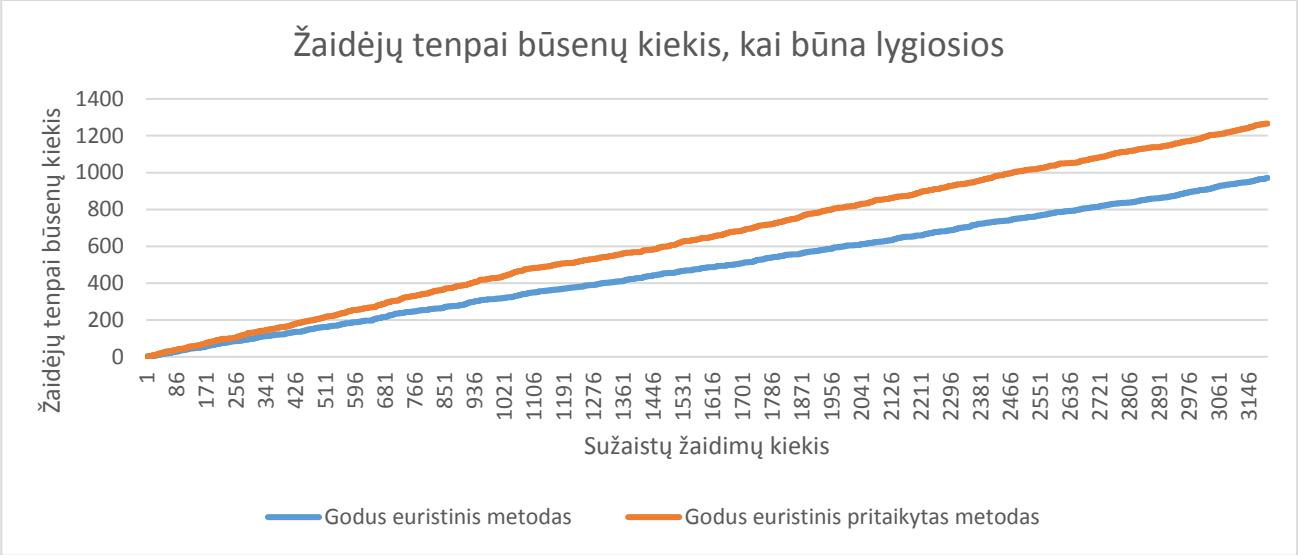
6.14 pav. Trečio eksperimento žaidėjų rezultatai



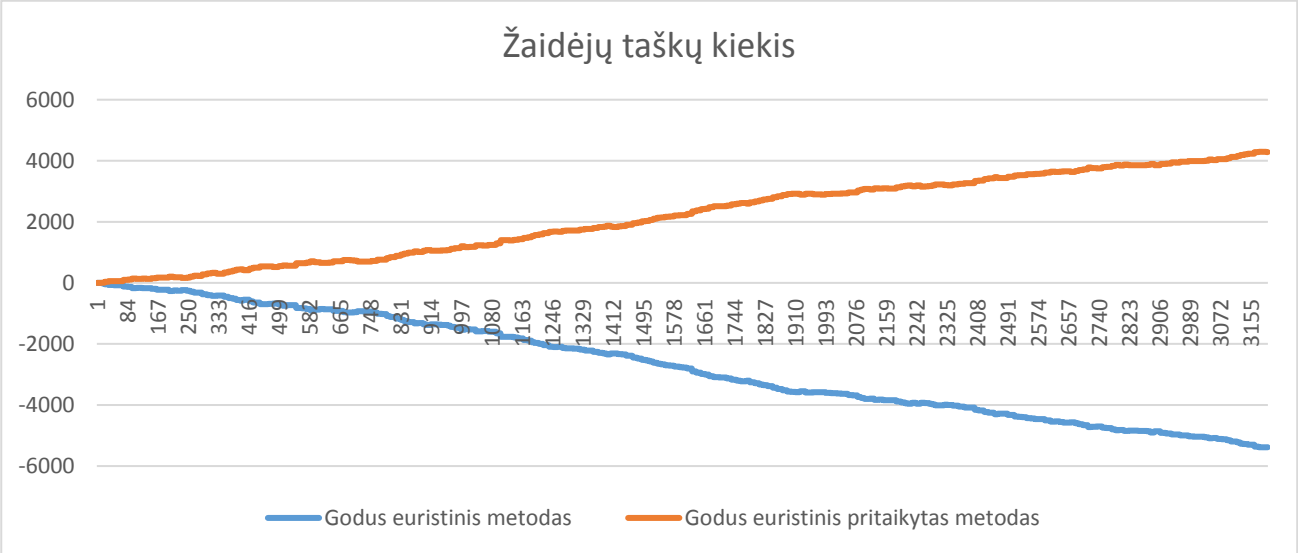
6.15 pav. Trečio eksperimento žaidėjų galutiniai taškai



6.16 pav. Trečio eksperimento žaidėjų pergalės



6.17 pav. Antro eksperimento žaidėjų *tenpai* būsenų kiekis, kai būna lygiosios



6.18 pav. Antro eksperimento žaidėjų taškai

6.3 lentelė. Trečio eksperimento rezultatai

	Godus euristinis metodas	**Godus euristinis pritaikytas metodas**
Pergalės	22.75%	77.25%
Pergalės pagal žaidimų kiekį	11.21%	38.07%
Lygiosios	43.41%	56.59%
Lygiosios pagal žaidimų kiekį	30.32%	39.54%
Pergalių, iš viso žaidėjo pasiektų *tenpai* santykis	26.99%	49.05%
Uždirbtų partijos taškų vidurkis	-20.8566	16.60853

Trečias eksperimentas galutinai užtikrina godaus euristinio pritaikyto metodo pirmą vietą (6.13 pav., 6.14 pav., 6.15 pav., 6.16 pav., 6.17 pav., 6.18 pav.). Reikia pastebėti, kad godaus euristinio metodo rezultatai pagerėjo. Padidėjo godaus euristinio metodo pergalių ir lygiųjų procentai. Pagerėjo pergalių ir iš viso pasiektų *tenpai* santykis. Taip pat reikia pastebėti, kad žymiai (apie 12%) padidėjo bendras pergalių kiekis. Tai galima paaiškinti tuo, kad eksperimente šie metodai žaidė po du. T.y. kadangi laimėnčių žaidėjų žymiai padaugėjo, jie pradėjo vienas kitam trukdyti. Todėl laimimų pergalių skaičius nepadvigubėjo, o tik išaugo 12%.

6.4. Eksperimentų rezultatų apibendrinimas

Atlikus eksperimentus išsiaiškinta, kad godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi yra geriausiai veikiantis algoritmas iš visų bandytų. Godus euristinis algoritmas per plauką užėmė antrą vietą. Pagal suvaržymų patenkinimo metodo pergalių kiekį galima manyti, kad algoritmo „rankos“ *yaku* vertes pakoregavus laukti greitesnių „rankų,“ šis algoritmas veiktų geriau. Neuroninį tinklą naudojantis algoritmas veikia vos geriau nei godus atsitiktinis algoritmas, kurias realiai neatlieka jokių rimtų skaičiavimų. Taigi arba toks neuroninio tinklo modelis netinka „Mahjong“ žaidimui, arba ši neuroninį tinklą reikia apmokyti žymiai ilgiau nei 140000 sužaistų žaidimų. Brutalios jėgos algoritmas deja veikia per lėtai, kad šį algoritmą būtų realu įvertinti.

7. IŠVADOS

Darbe suprojektuoti ir ištirti šeši „Mahjong“ žaidimo algoritmai:

1. Godus atsitiktinis algoritmas pasirinktas kaip atskaitos taškas, kadangi atsitiktinis kaladėlių pasirinkimas yra mažiausiai apgalvotas pasirinkimas. Šio algoritmo rezultatai naudojami kitų algoritmų efektyvumui parodyti. Kaip ir būta tikėtasi, šis algoritmas veikė prasčiausiai, surinko mažiausiai partijos taškų eksperimentuose.
2. Godus euristinis algoritmas pasirinktas kaip paprastas, greitas, nesudėtingas algoritmas. Šis algoritmas yra antras pagal efektyvumą, vos laimint prieš suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstą algoritmą. Šio algoritmo pergalių ir iš viso algoritmo pasiektų *tenpai* santykis palyginus su kitais efektyviais algoritmais yra žemas, kas nurodo, kad algoritmas dažniau nei kiti algoritmai patenka į nelaimimą būseną.
3. Godus euristinis algoritmas pritaikytas greitam „rankos“ vystymuisi yra modifikuotas godaus euristinio algoritmo variantas, kuris pritaikytas greitoms „rankoms“ laimėti. Šis algoritmas yra efektyviausias iš visų išbandytų algoritmų, surenka daugiausiai partijos taškų ir pergalių, bei *tenpai* būsenų.
4. Giliu Q neuroniniu tinklu pagrįstas algoritmas yra antras nuo galo pagal efektyvumą algoritmas. Šis algoritmas vos laimėtų taškų kiekiu aplenkė godų atsitiktinį algoritmą, nors godus atsitiktinis algoritmas ir laimėjo daugiau pergalių, o lygiųjų buvo beveik vienodai. Tai nurodo, kad šis algoritmas šiek tiek geriau žino nei godus atsitiktinis algoritmas, kokių kaladėlių nemesti, kad nepralaimėtum dar daugiau taškų.
5. Suvaržymų patenkinimo uždavinio minimalių konfliktų metodu pagrįstas algoritmas artimai kovoja dėl antros vietos pagal efektyvumą. Nors algoritmas ir laimėjo daugiau pergalių nei godus euristinis algoritmas, mažesnis *tenpai* būsenų kiekis nulėmė šio algoritmo trečią vietą pagal efektyvumą. Šio algoritmo geras pergalių ir iš viso algoritmo pasiektų *tenpai* santykis nurodo, kad jeigu šis algoritmas dažniau pasiektų *tenpai* būseną, šio algoritmo efektyvumas pagerėtų. *Tenpai* būsena lengviausia būtų pasiekti pakoreguojant algoritmo „rankų“ vertes, padidinant algoritmo greitų „rankų“ vystymo dažnį.
6. Brutalios jėgos metodu pagrįstas algoritmas nebuvo įvertintas, kadangi net vienas algoritmo ėjimas užtrunka virš minutės. Šį algoritmą reikėtų modifikuoti taip, kad būtų sumažintas apdorojamų „kibirų“ kiekis.

LITERATŪRA

[1] European Mahjong Association, „Riichi Rules for Japanese Mahjong,“ 2016. [Tinkle]. Available: http://mahjong-europe.org/portal/images/docs/Riichi-rules-2016-EN.pdf. [Kreiptasi 18 04 2017].

[2] „Shanten,“ 2017. [Tinkle]. Available: http://arcturus.su/wiki/Shanten. [Kreiptasi 01 05 2017].

[3] N. Mizukami ir Y. Tsuruoka, „Building a Computer Mahjong Player Based on Monte Carlo Simulation and Opponent Models,“ įtraukta *Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games (CIG 2015)*, 2015.

[4] M. Bowling, N. Burch, M. Johanson ir O. Tammelin, „Heads-up limit hold’em poker is solved,“ *Science*, t. 347, nr. 6218, p. 145–149, 2015.

[5] A. Van Der Kleij, „Monte Carlo tree search and opponent modeling through player clustering in no-limit Texas hold’em poker,“ Master’s thesis, University of Groningen, 2010.

[6] A. Davidson, „Opponent modeling and search in poker: Learning and acting in a hostile and uncertain environment,“ Master’s thesis, University of Alberta, 2002.

[7] M. Buro, J. R. Long, T. Furtak ir N. R. Sturtevant, „Improving state evaluation, inference, and search in trick-based card games,“ įtraukta *Proceedings of 21th International Joint Conference on Artificial*, 2009.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra ir M. Riedmiller, „Playing Atari with Deep Reinforcement Learning,“ 2013.

[9] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[10] G. Tesauro, „Temporal difference learning and td-gammon,“ *Communications of the ACM*, t. 38, nr. 3, p. 58–68, 1995.

[11] S. Russell ir P. Norvig, Artificial Intelligence: A Modern Approach, 2nd edition, Prentice Hall, 2003.

[12] Continuum Analytics, Inc., „Continuum,“ 2017. [Tinkle]. Available: https://www.continuum.io/. [Kreiptasi 01 05 2017].

[13] A. Li, „pyriichi,“ 2017. [Tinkle]. Available: https://github.com/darkfeline/pyriichi. [Kreiptasi 01 05 2017].

[14] Google Brain Team, „TensorFlow,“ 2017. [Tinkle]. Available: https://www.tensorflow.org/. [Kreiptasi 01 05 2017].