TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKA TEADUSKOND
Arvutiteaduse instituut
Informaatika õppekava

Marit Asula

Strateegia ja mängupositsiooni hindamine juhuslikkuse elementi sisaldavas kahe isiku mängus

Bakalaureusetöö (6 EAP)

Juhendaja: prof. emer. Mare Koit

Tartu 2015

Strateegia ja mängupositsiooni hindamine juhuslikkuse elementi sisaldavas kahe isiku mängus

Lühikokkuvõte:

Antud töö eesmärk on Tartu Ülikoolis õpetatava tehisintellekti kursuse kuulajatele lisamaterjali loomine, mis aitaks paremini mõista strateegia ja mängupositsiooni hindamist juhuslikkuse elementi sisaldavas kahe isiku mängus. Töös keskendutakse juhuslikkuse elementi sisaldavate kahe isiku mängude jaoks mõeldud expect-minimax algoritmile ning võimalustele selle otsingut parandada. Lisaks on töö käigus loodud programm, mis aitab triktraki mängu näitel expect-minimaxi algoritmi töö käiku visualiseerida, võimaldades algoritmis valida erinevate otsingut mõjutavate parameetrite vahel.

Võtmesõnad:

triktrakk, expect-minimax, minimax, alfa-beeta kärpimine, hinnangufunktsioon, otsing mänguseisude puul, juhuslikkuse tipud

Evaluation of Strategy and Game Position In Two Player Game That Includes an Element of Chance

Abstract:

The goal of this thesis is to produce a learning material for students taking artificial intelligence course at University of Tartu to help improve understanding of two player strategic games that include an element of chance. The thesis focuses mainly on expect-minimax algorithm, which is a common algorithm used in two player games with element of chance. In addition, a program was developed to illustrate the work of expect-minimax algorithm on the example of backgammon.

Keywords:

backgammon, expect-minimax, minimax, alpha-beta pruning, evaluation function, game tree search, chance nodes

Sisukord

Sissejuhatus

Triktrakk on tüüpiline juhuslikkuse elementi sisaldav kahe isiku strateegiamäng, mille mängupositsiooni hindamise üks võimalusi on expect-minimaxi algoritmi kasutades sooritada otsing juhuslikkuse tippe sisaldaval mänguseisude puul. Antud töös võetakse vaatluse alla expect-minimaxi algoritm ning erinevad võimalused selle otsingu parandamiseks. Algoritmi tööga aitab lähemalt tutvuda töö käigus valminud programm, mis võimaldab valida erinevaid expect-minimaxi algoritmis kasutatavaid parameetreid nagu otsingupuu sügavus ja strateegia ning seeläbi analüüsida algoritmi edukuse sõltuvust nendest faktoritest.

Esimeses peatükis (triktrakk) antakse ülevaade triktraki ajaloost, mängureeglitest ning levinumatest strateegiatest.

Teises peatükis (otsing juhuslikkuse elementi sisaldava kahe isiku mängu seisude puul) kirjeldatakse, kuidas toimub otsing juhuslikkuse tippe sisaldaval mängupuul ning antakse ülevaade võimalustest otsingu parandamiseks.

Kolmandas peatükis (programmi ülevaade) tutvustatakse töö käigus valminud programmi, mis aitab triktraki näitel visualiseerida expect-minimaxi algoritmi tööd. Lisaks antakse ülevaade programmis kasutatud hinnangufunktsioonidest ning nende põhjal kombineeritud strateegiatest.

Neljandas peatükis analüüsitakse expect-minimaxi tulemusi erinevate parameetrite korral.
Viiendas peatükis tehakse ettepanekuid programmi võimalikeks edasiarendusteks.
Lisana on kaasas töö käigus valminud programm ning selle kasutusjuhend.

1 Triktrakk

Triktrakk (ing. k *Backgammon*) on üks vanimaid kahele isikule mõeldud lauamänge, mis kombineerib oskusi ning õnne. Nuppe liigutatakse vastavalt täringu veeretamise tulemusel saadud silmade arvule ning mängija võidab, kui suudab kõik oma nupud enne vastast laualt eemaldada.

Selles peatükis antakse ülevaade triktraki ajaloost, mängureeglitest ning levinumatest strateegiatest.

1.1 Triktraki ajalugu

Triktrakk on koos go ja kabega üks vanimaid mänge maailmas. Selle vanuseks hinnatakse umbes 5000 aastat ning päritolupiirkonnaks arvatakse olevat Mesopotaamia. Mängu populariseerisid roomlased, kelle tolleaegne versioon kandis nime *Duodecum Scripta et Tabulae.* Triktraki nimetus ilmus trükituna esimest korda 1645. aastal. [1] Kuna osades ringkondades tohtis seda mängida vaid kõrgem klass nagu aristokraadid ja kuningliku perekonna liikmed, on seda kutsutud ka Mängude Kuningaks.[2]

Mängul on tulnud mitmeid kordi vastu astuda võimuorganitele ja kirikule, kes on üritanud selle mängimise keelata õnnemängu elemendi tõttu. [1] Nii juhtus ka 1982. aastal USA-s Oregoni osariigis, kus triktraki mängimine taheti ära keelata väitel, et see on õnnemäng ning seega vastuolus Oregoni rangete hasartmänge keelustavate seadustega. Triktraki kaitseks välja astunud Paul Magriel väitis aga, et kõnealuses mängus on kogu informatsioon mänguseisu kohta mõlemal mängijal olemas ning võidab see, kes seda paremini kasutada oskab. Kohus jõudis otsusele, et triktrakk on siiski strateegia-, mitte õnnemäng ning nii pääses see keelustamisest.[3]

1.2 Triktraki mängureeglid

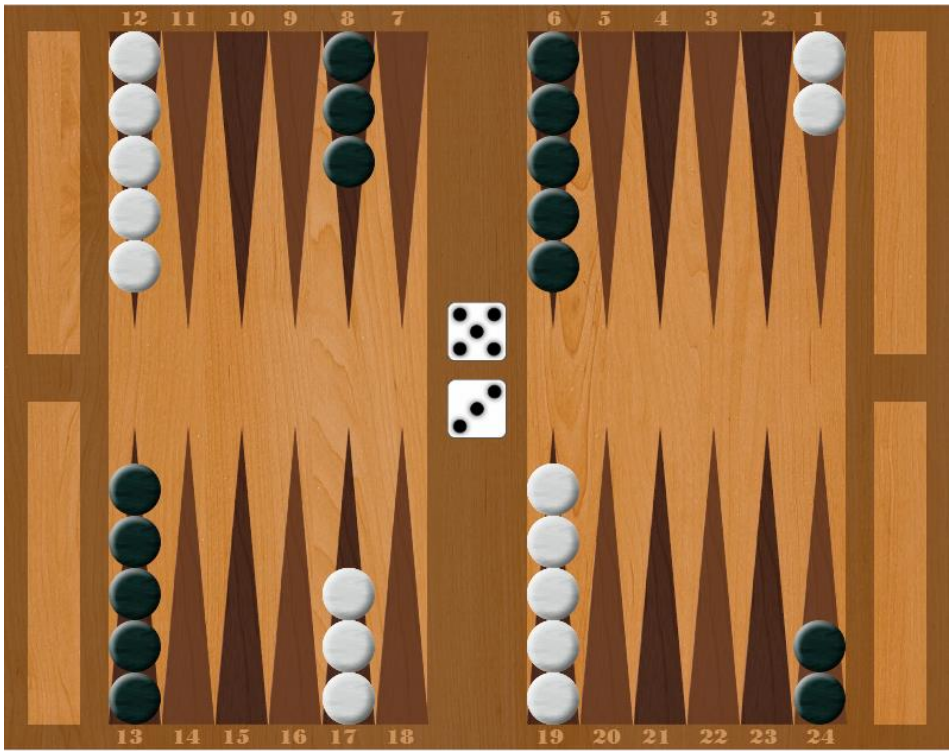
Mängulaud ja nupud

Triktraki mängulaud koosneb 24 kiilust (ing. k *points*), millest 12 asuvad valge ja 12 musta poolel. Kiilud on vaheldumisi heledat ja tumedat värvi. Mängulaua keskel on ala, mida nimetatakse müüriks (ing. k *bar*) (vt. joonis 1). Mõlema mängija valduses on 15 nuppu, millel on laual kindel algpositsioon, mis on näha joonisel 2. Valge mängib kiilude 13-24 ja must kiilude 1-12 poolel. Kiilud 19-24 on valge siselaud ning kiilud 1-6 musta siselaud (vt. joonis 3). Siselauast väljapoole jäävad kiilud 13-18 on valge sektor ja 7-12 musta

sektor. Valge mängib laual vastupäeva, liigutades nuppe madalamatelt kiiludelt kõrgemate poole. Must peegeldab tema liikumist ja liigutab oma nuppe päripäeva kõrgetelt kiiludelt madalate poole (vt. joonis 4).

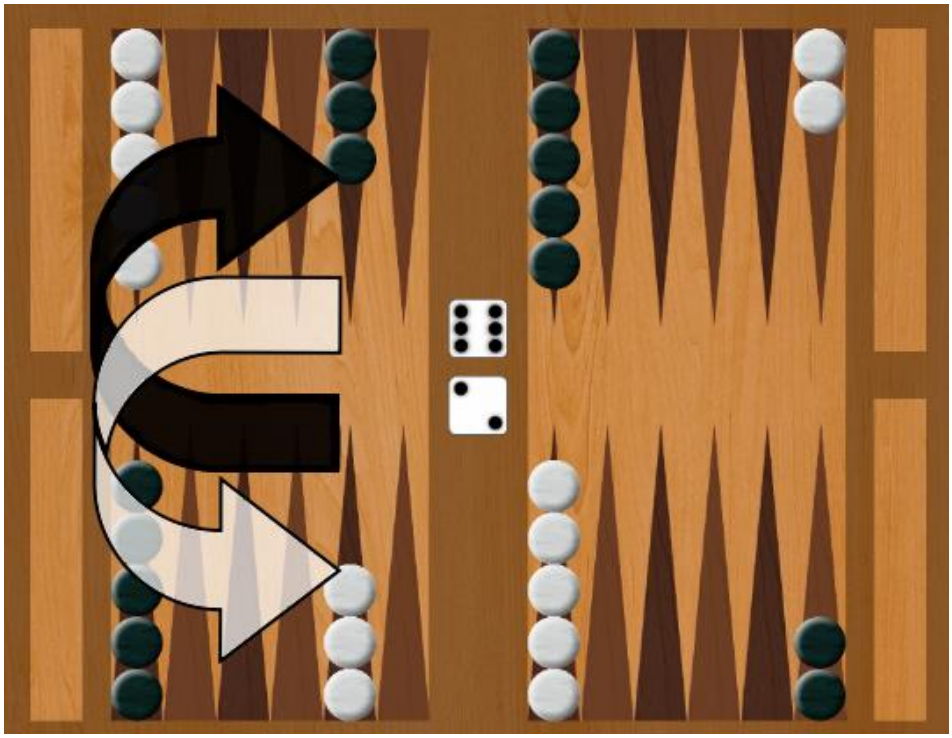
Igal kiilu peal võib olla ükskõik kui palju sama värvi nuppe, kuid vastase nupud ei saa ühtegi kiilu jagada, mis tähendab, et ühel kiilul ei saa korraga olla kunagi erivärvi nuppe. [4]

Joonis 1. Müür ja kiilud.



Joonis 2. Triktraki nuppude algpositsioon mängulaual.

Joonis 3. Valge ja musta siselaud.



Joonis 4. liigub mängulaual päri- ning valge vastupäeva.

Mängu eesmärk

Triktrakk on lihtne võiduajamismäng, kus nuppe liigutatakse mängija siselaua suunas. Kui kõik 15 nuppu on mängija siselauale jõudnud, võib need laualt kõrvaldada. Võitjaks saab esimene mängija, kes kõik oma nupud laualt eemaldab. [4]

Mängu algus

Mängijad käivad kordamööda ning sammude arvu määravad kaks täringut. Mängu alguses veeretab kumbki mängija täringut ja kõrgema silmade arvu veeretanu alustab, võttes aluseks mõlemad veeretatud silmad. Näiteks, kui valge veeretab 6 ja must 2, siis alustab mängu valge (tänu suuremale silmade arvule) ja silmade arvuks on 6 – 2. Kui mängijad veeretavad sama arvu silmi, tuleb uuesti veeretada.[5]

Veeretamine ja käigud

Mängijal on lubatud liikuda kiilule, mis on kas

a) hõivatud ühe või mitme tema enda nupuga,

b) tühi või

c) hõivatud täpselt ühe vastase nupuga.

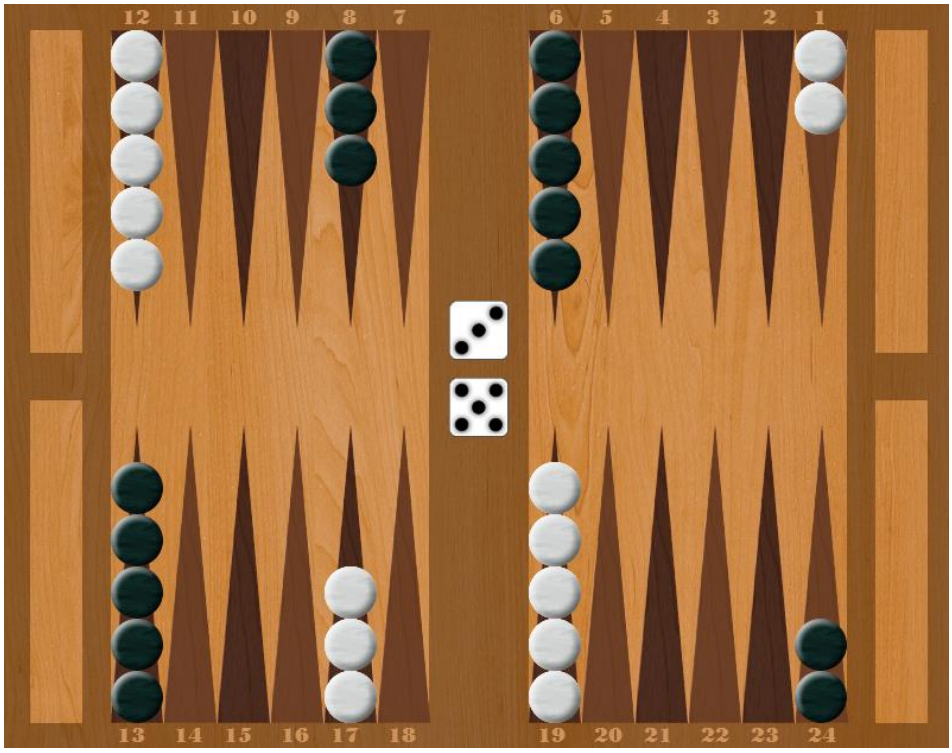
Seega on mängija jaoks suletud kõik kiilud, millel on kaks või enam vastase nuppu.

Kahe täringu silmad näitavad, kui kaugele mängija võib oma nuppu laual liigutada. Silmad võib kokku liita ja liigutada vaid ühte nuppu, kuid samuti on võimalik liigutada korraga kahte nuppu vastavalt veeretatud silmadele. [5]

Joonisel 5 on toodud valge avapositsioon koos täringute silmade arvuga (3-5), mis võimaldab mängukorra ajal liigutada ühte nuppu kaks korda (kokku 8 sammu võrra) või kahte erinevat nuppu vastavalt 3 ja 5 sammu. Vaatleme näidet mõlemast juhust mängulaua algseisu puhul.

1. Valge võib liigutada oma nuppu kõigepealt kiilult 1 kiilule 4 (kasutades täringu silmade arvu 3) ning seejärel kiilult 4 kiilule 9 (kasutades täringu silmade arvu 5). Siinjuures tuleb aga tähele panna, et valgel peab käigu sooritamise ajal olema võimalik ühel täringu silmade arvuga näidatud kiiludest maanduda. Kui ka 4. kiilu kataks vastase nupud, poleks valgel võimalik antud käiku sooritada, olgugi, et 9. kiil on ju vaba. Kuna nii kiilust 1 kolme sammu kaugusel olev 4. kiil kui ka kiilust 1 viie sammu kaugusel olev 6. kiil oleks sellisel juhul suletud, ei ole valgel võimalik vahepeal mängulaual maanduda.

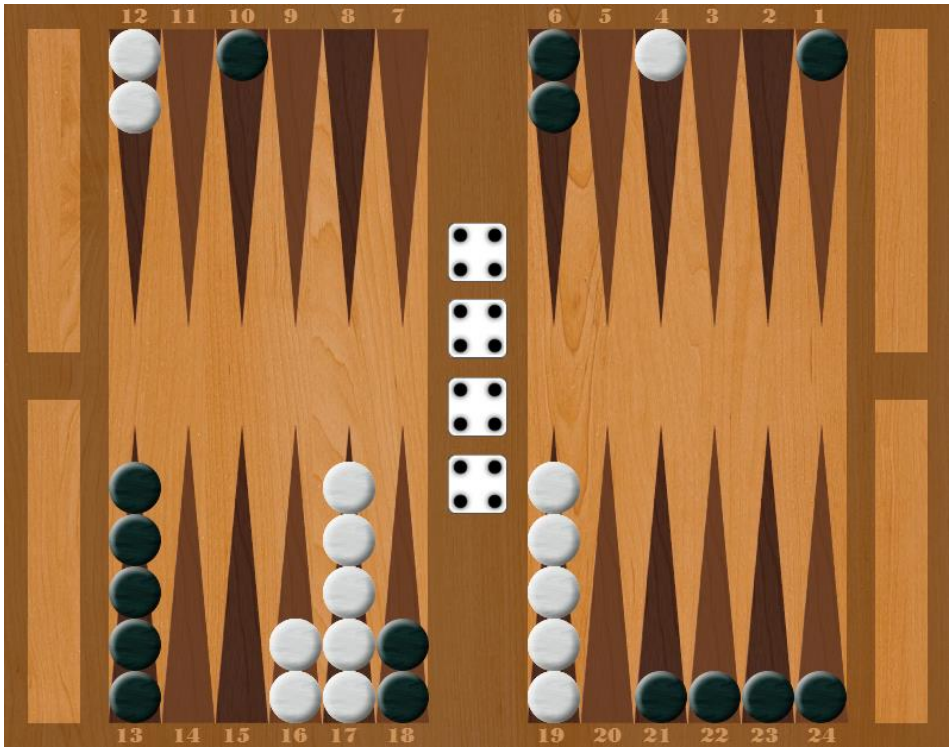
2. Valge võib liigutada ühte nuppu 12-lt 15-le ning teist nuppu 17-lt 22-le.



Joonis 5. Täringute silmade arv määrab ära valge legaalsed käigud.

Kõikvõimalikud täringukombinatsioonile (3-5) vastavad avapositsiooni käigud on järgnevad:
(1-4,4-9), (1-4,12-17), (1-4,17-22), (12-15,17-22), (12-17,17-20), (12-17,19-22), (17-22,19-22)

Kui veeretatakse duubel (sama arv silmi mõlemal täringul), kahekordistub silmade arv, otsekui oleks veeretatud nelja täringut, millest kõik andsid sama arvu silmi. Seetõttu on nüüd võimalik liigutada kuni nelja nuppu.



Joonis 6. Sama silmade arvu veeretamise korral kahekordistub mängija võimalike käikude arv.

Sama silmade arvuga täringu veeretamise tulemusel on mängijal oma korra ajal võimalik teha 4 erinevat käiku, joonisel 6 kujutatud mängulaua seisu arvestades nende hulgas näiteks:

1. liikuda terve käik ainult ühe nupuga: esmalt 4-lt 8-le, seejärel sama nupuga 8-lt 12-le, uuesti sama nupuga 12-lt 16-le ning lõpuks 16-lt 20-le ning seega nupp vastase siselaualt edukalt enda omale liigutada;
2. teha neli käiku nelja erineva nupuga: näiteks liikuda esimese nupuga 4-lt 8-le, teise nupuga 12-lt 16-le, kolmanda nupuga 17-lt 21-le ning lõpuks neljanda nupuga 19-lt 23-le.

Kahe viimase käigu tulemusel rünnatakse sealjuures ka vastase nuppe.

Rünnak ja tagasikäik müürilt

Üksikuid nuppe mängulaual ähvardab pidevalt vastase rünnaku oht. Kui vastane nuppu ründab, tõstetakse see müürile. Müürile sattunud nupp tuleb mängu tagasi tuua vastase siselaua kaudu. Mängija, kelle nupp on müüril, ei saa senikaua mängulaual ainsatki käiku teha. Joonisel 7 on valge nupp müürile löödud. Kuna valge on veeretanud täringute silmade arvu (3 - 4), peab ta müürilt pääsemiseks liikuma kiilule number 3 või number 4.

Kuna kiil number 4 on antud juhul vastase nuppudega suletud, on valgel müürilt maha saamiseks võimalik käia ainult 3. kiilule. Juhul kui valge täringute veeretamise tulemuseks oleks olnud silmade arv (4 – 6), poleks valgel nuppu seekord müürilt maha kanda olnud võimalik, kuna nii 4. kui ja 6. kiil on vastase nuppudega suletud, ning seega oleks valge pidanud ühe käigu vahele jätma. Kui valgel oleks õnnestunud aga ühe täringu silmade arvuks veeretada 5, oleks tal olnud võimalik müürilt maha liikuda ning seejuures omakorda vastase üksik nupp müürile saata.

Joonis 7. Valge nupp on müürile löödud.

Mahakandmine ja mängu lõpp

Triktraki lõppeesmärgiks on kõik oma nupud vastasest kiiremini maha kanda. Mängija saab mahakandmisega alustada, kui kõik tema nupud on jõudnud tema enda siselauale. Mahakandmise käigus liiguvad nupud tavapärasel viisil, ainult, et nuppudel on võimalik liikuda mängija 1. (või 24.) kiilult kujuteldavale 0-ndale (või 25-ndale) kiilule. Niisugusel juhul eemaldatakse nupp mängulaualt.

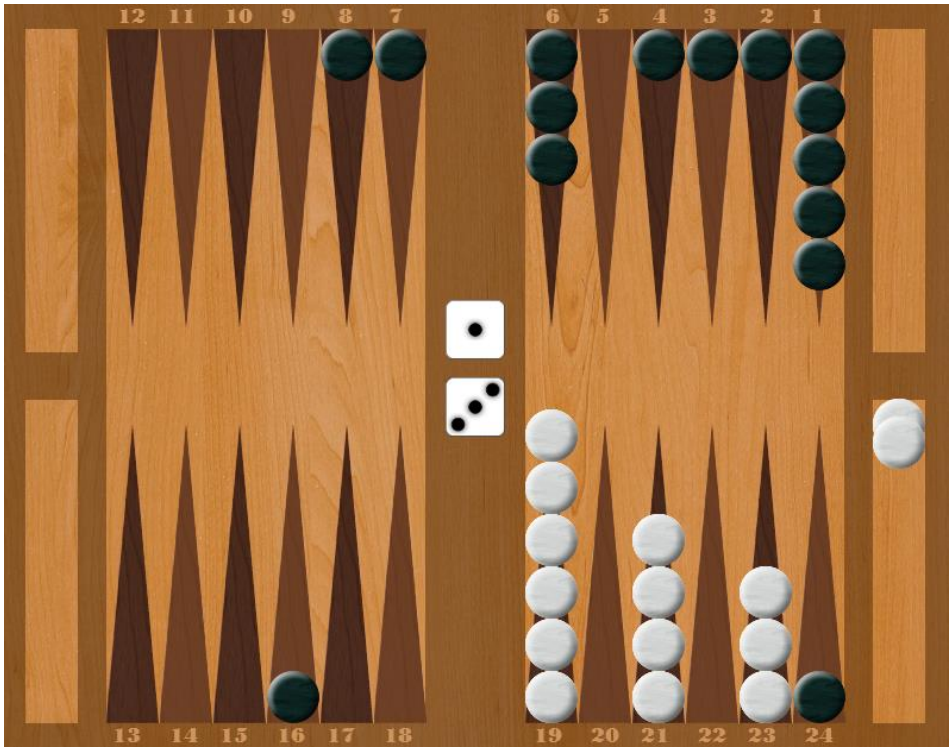
Nuppe on võimalik maha kanda vaid siis, kui kõik mängija nupud on tema siselaual. Vahel juhtub, et näiteks valge hakkab nuppe maha kandma, kui mustal on veel mõned nupud valge siselaual. Niisugusel juhul on võimalik, et valge jätab mõne nupu üksikuks ja see

saadetakse müürile. Kui nupp on müürilt mängu tagasi toodud, peab valge selle uuesti oma siselauale manööverdama.

Mahakandmiseks võib kasutada ka vajalikust suuremat silmade arvu, kui selle arvu silmadega ei ole võimalik teha ühtki muud käiku peale mahakandmise.

Joonis 8. Valge mängija nuppude mahakandmine.

Kuna joonisel 8 kujutatud mängulaua seisul pole valgel ei mahakandmisest viie sammu kaugusel oleval 20. kiilul ega ühelgi sellest kaugemal asetseval kiilul nuppe, võib mängija nupu maha kanda hoopis 22. kiilult, mis on mahakandmislauast kõige kaugemal asuv kiil valge mängija nuppudega. Teise käiguna võib valge kas kiilult 22 nupu kiilule 24 liigutada või kiilult 23 nupu maha kanda.



Joonis 9. Täringu silmade arv ei luba valge mängija esimese käigu jooksul ühtegi nuppu maha kanda.

Joonisel 9 on kujutatud olukord, kus kõik valge mängija nupud on tema siselaual, kuid kus mängijal ei ole oma esimese käigu ajal võimalik ühtegi nuppu laualt maha kanda, kuna ei mahakandmislauast ühe sammu kaugusel asuval 24. kiilul ega ka mahakandmislauast kolme sammu kaugusel asuval 22. kiilul pole ühtegi valge mängija nuppu. Samuti ei ole mahakandmislauast kaugemal asuvatelt kiiludelt nuppe maha kantud. Seega tuleb esimese käigu ajal mängijal oma nuppe lihtsalt ühe või kolme kiilu võrra edasi liigutada. Teise käigu jooksul on mängijal siiski võimalik üks oma nuppudest mängulaualt maha kanda, kui esimesel käigul liikuda kiilult 21 kiilule 24 (võttes aluseks täringu silmade arvu 3) ning seejärel nupp laualt maha kanda (võttes aluseks täringu silmade arvu 1). Boonusena saadetakse selle käigu tulemusel ka üks vastase nupp müürile.

Punktide lugemine

Harilikult teenivad mängijad triktraktis ühe punkti iga võidetud matši pealt. On aga kaks erandit, mil võitjal on õigus rohkem nõuda. Kui mängija jõuab kõik oma nupud maha kanda enne, kui vastane mahakandmist alustab, saab ta topeltpunktid. Veelgi parem tulemus ootab ees juhul, kui mängija kannab maha kõik oma nupud, vastasel aga on veel mõni

nupp tema siselaual. Seda nimetatakse triktrakiks ja selle eest on ette nähtud kolmekordne punktisumma.

Topeltpunkte saadakse üsna sageli ja selle tõenäosus sõltub suuresti mängija strateegia-oskustest. Triktrakk on aga väga haruldane ja seda võib pidada õnnelikuks lisavedamiseks. [5]

Topeldamistäring

Iga triktrakikomplekti juurde kuulub ka täring numbritega 2, 4, 8, 16, 32 ja 64. Topeldamistäring on leidlik võte, mis aitab mängu tuua pingelisust, võimaldades ühel mängijaist teha ettepanek panuseid tõsta, kui talle tundub, et õnn on tema poolel. Mängu alguses pannakse topeldamistäring laua keskele ja mõlemad mängijad võivad seda kasutada. Harilikult pööratakse täringu ülemiseks numbriks 64 ja täring asub mängulaua kõrval. Sel ajal on panuseks üks ühik. Mäng jätkub tavapärasel viisil, kuni üks mängijatest otsustab, et tal on võimalus mäng võita ja võib-olla koguni topeltpunktideni jõuda. Niisugusel juhul pakub mängija täringut oma vastasele.

Oletame, et valge on eelisseisus ja soovib seda sammu astuda. Valge võtab laua keskelt täringu, pöörab selle nii, et ülemiseks numbriks jääb 2 ja asetab selle musta lauapoolele. Kui mustale tundub, et valge võit on tõenäoline, tunnistab ta kaotust ja jääb ilma mängu pandud ühikust. Edasisi käike pole vaja ning algab uus mäng. Teisalt võib must aga soovida mängu jätkata. Sellisel juhul võtab must valge pakutud duubli vastu ja mäng jätkub. Nüüd aga on panuseks kaks ühikut ja musta kaotuseks võib ühe asemel olla kaks ühikut (või koguni neli, kui valge peaks jõudma topeltpunktideni).

Duublit vastu võttes saab must aga enda valdusessse topeldamistäringu. Kui mäng peaks hilisemas staadiumis pöörduma musta kasuks, võib ta omakorda pakkuda täringut valgele, nõudes panuste tõstmist kahekordselt neljakordsele. Valgel on sellisel juhul võimalus loobuda ja loovutada kaks ühikut või jätkata mängu panusega neli ühikut. Selline protsess võib kesta kaua ja panused suurenevad kahelt neljale, seejäral kaheksale, siis kuueteistkümnele ja nõnda edasi. Kahe ja neljani jõutakse üsnagi tihti, kaheksa on juba haruldane ja kuusteist lausa tavatu. [5]

1.3 Triktrakis kasutatavad strateegiad

Võtame vaatluse alla tuntuimad triktrakis kasutatavad strateegiad. [6]

Võidujooks

Võidujooks (ing. k *the running game*) on kõige lihtsam triktraki strateegia, mis seisneb võimalikult kärmelt mängija nuppude liigutamises oma siselauale. Juhul, kui mõlemad mängijad on otsustanud just selle strateegia kasuks, võidab mängija, kellel täringute veeretamisel rohkem õnne on.

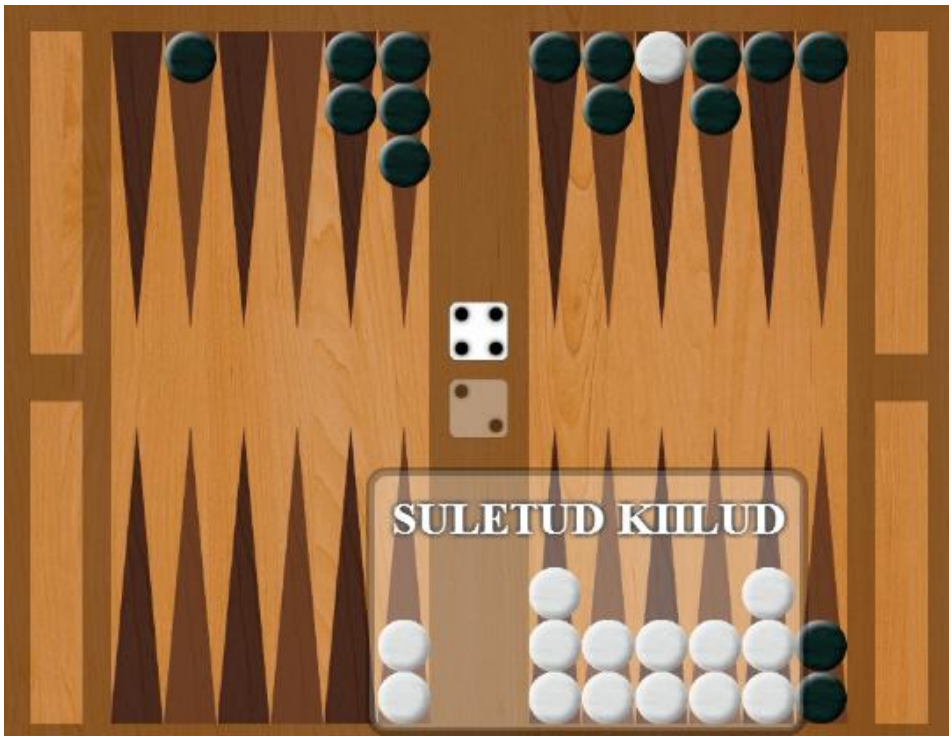
Rünnak

Rünnak (ing. k *the blitz*) seisneb igal avaneval võimalusel vastase üksikuks jäänud nupu müürile saatmises. Lisaks vastase nupu tema siselauast võimalikult kaugele saatmisele, annab see ka võimaluse vastase nuppu vangis hoida, juhul kui vastane ei veereta täringutega silmade arvu, mis lubaks tal nupu mängu tagasi tuua. See omakorda tähendab aga, et vastane peab vähemalt ühe käigu, üldiselt aga isegi rohkem, vahele jätma.
Siiski tasub meeles pidada, et oma siselaual vastase nuppe rünnates ei kaota vastane just palju edumaad ning juhul, kui ründe sooritamise tulemusel jääb mõni mängija enda nuppudest üksikuks, võib see vastase ründele kergeks saagiks osutuda ning kaotatud edumaa saab vastase omast tunduvalt suuremaks.

Kiilude sulgemine

Kiilude sulgemine (ing. k *priming*) on strateegia, mis kätkeb endas mängulaual järjestikuste kiilude sulgemist enda nuppudega (vt. joonis 10).
Nelja järjestikuse kiilu täitmisel oma nuppudega luuakse müür, millest vastasel pole võimalik mööda pääseda täringuga number viit või kuut veeretamata. Lisaks sellele, et suletud kiilude ahelik vastast aeglustab, on ka vastavad kiilud mängija enda nuppudele turvaliseks peatuspaigaks.
Kiilude sulgemine on kõige efektiivsem koos rünnakuga: rünnak hoolitseb selle eest, et vastase nupud saadetaks müürile ja kiilude sulgemine aitab omakorda takistada müürile

saadetud nuppudel mängu tagasi astuda. Tulemusena võib vastane olla sunnitud mitu käiku vahele jätma.[6]



Joonis 10. Kuna valge on sulgenud kuus järjestikust kiilu, ei pääse valge siselaual olevad mustad nupud liikuma enne kui valge on mõne kiiludest vabastanud.

Kinnipidamine

Kinnipidamine (ing. k *the holding game*) on strateegia, mille rakendamisel jäetakse vastase siselauale üks oma nupp, nii öelda ankur (vt. joonis 11). Kuigi tihtilugu on lõksu jäämise vältimiseks mõistlik oma nupud vastase siselaualt võimalikult kärmelt ära liigutada, annab ankru sinna jätmine kaks eelist:

1. kaitse seisukohast tagab see turvalise maandumispunkti, juhul kui üks mängija enda nuppudest peaks sattuma müürile. See ei lase vastasel kiilude sulgemise tehnikaga mängija nupp pikaks ajaks lõksu panna;
2. ründe seisukohast on ankur vastasele arvestatav oht, kuna oponendi nupu tema siselaualt müürile saatmine tähendab vastase jaoks mängulaua algusesse tagasi

sattumist. Kui vastane peaks aga ankrut ründama, ei ole kaotus kuigi suur, kuna ankur tegutses nagunii samas piirkonnas ning ei liigu seega kiilude arvult sugugi nii palju tagasi kui vastase nupp ründe tagajärel liikuma peab.

Joonis 11. Valge mängija on jätnud ankru musta siselauale.

Tagamäng

Tagamäng (ing. k the *back game*) viib kinnipidamise sammu võrra kaugemale ja tugineb vastase siselaual mitme oma nuppudega kaetud kiilu hoidmisele. Kuna mängu alguses on vastase siselaual ainult kaks mängija enda nuppu, on tagamängu võimalik teostada ainult siis, kui vastane lööb mängija nupud korduvalt müürile. Tagamäng pole niivõrd sihilik strateegia, kui üritus kehva olukorda enda kasuks pöörata.

Võrreldes kinnipidamisega, on tagamängu rakendades väiksem ründeoht mängija oma nuppudele, kuna üksiku nupu asemel tegutseb vastavas piirkonnas mitu. Lisaks aitab tagamäng tekitada vastasele ebamugavust hallates osa tema väärtuslikust siselauast.

2 Otsing juhuslikkuse elementi sisaldava kahe isiku mängu seisude puul

Igapäevaelus võib sageli ette tulla olukordi, mida ette näha on küllaltki raske. Oletame, et kriitilises seisundis olev inimene on vaja haiglasse toimetada ning valida on kahe erineva sõidutee vahel: esimene, mis on pikem, aga ilma ühegi takistuseta, teine küll lühem, aga sisaldab oma teel mitut valgusfoori. Kuidas langetada sellisel juhul otsus? Parimal juhul on teise teekonna valimisel kõikides valgusfoorides neist möödudes roheline tuli, mille tulemusel tasub lühema tee valimine ära. Mis aga juhtub, kui autojuhti tabab ebaõnn ning igas fooris on punane tuli? Juhul kui pikem tee oli lühemast oluliselt suurema kilometraažiga, tasub isegi arvukate punaste foorituledega lühema tee valimine ära. Kui ajavõit sõltub aga puhtalt valgusfoorides põlevast tulest, on otsuse langetamisel kasulik hinnata tõenäosust, et foorini jõudmisel põleb selles just roheline tuli. [7]

Paljud mängud imiteerivad seesugust igapäevaelus esinevat ettearvamatust lisades juhuslikkuse elemendi, näiteks täringu veeretamise. Üks seesugustest mängudest on ka triktrakk, kuna mängija legaalsete käikude teada saamiseks veeretatakse tema mängukorra alguses täringuid. Kuna mängijate lubatud käigud selguvad alles pärast täringu veeretamist, ei ole võimalik konstrueerida standardset mänguseisude puud.

2.1 Minimaxi algoritm

Vaatleme esmalt otsuste langetamist mängus, kus mängija on alati teadlik kõikidest võimalikest oma käigule järgnevatest vastase käikudest. Seesuguse mängu võib formaalselt defineerida kui teatud liiki otsinguprobleemi järgnevate komponentidega.

- **Algseis**, mis hõlmab mängulaua algseisu ja teeb ühtlasi kindlaks, milline mängija alustab.
- **Järglaste leidmise funktsioon** (ing. k *successor function*), mis võtab sisendiks lauaseisu ning tagastab listi paaridest kujul (*käik*, *lauaseis*), mille esimene komponent näitab üht võimalikku käiku ning teine komponent sellest käigust tulenevat lauaseisu.
- **Terminaalsuse test** (ing. k *terminal test*), mis teeb kindlaks, millal mäng läbi saab. Lauaseise, millega mäng lõpeb, kutsutakse lõppseisudeks.

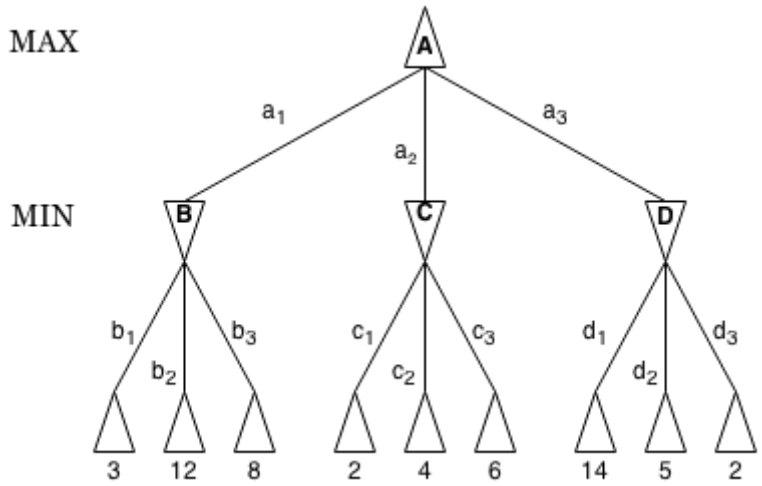
- **Kasulikkusfunktsioon** (ing. k *utility function*), mis annab numbrilise väärtuse lõppseisudele. Males on tulemuseks võit, kaotus või viik ning vastavateks väärtusteks +1, -1 või 0. Võimalike tulemuste piirid võivad aga ka laiemad olla: triktrakis näiteks +192-st kuni -192-ni (kui üks osapooltest saavutab 3 punkti andva triktraki ning topeldamistäringul on jõutud 64-ni).

Algseis ja mõlema osapoole legaalsed käigud määravad mängu jaoks üheselt ära mängu-puu. [8]

Minimax-protseduurid püüavad minimeerida käigust tuleneda võivat maksimaalset kahju. Algoritmi idee on genereerida jooksvast positsioonist lähtudes võimalike järglaste hulk kuni puu lehtedeni (terminaalsete tippudeni), nendele positsioonidele rakendada kasulikkusfunktsiooni ja, liikudes puus tase-haaval kõrgemale tagasi lähtepositsiooni, arvutada selle hinnang. Siin me eeldame, et kasulikkusfunktsioon omistab suured väärtused headele positsioonidele, nii et meie eesmärk on maksimeerida järgmise positsiooni hinnangut. [9]

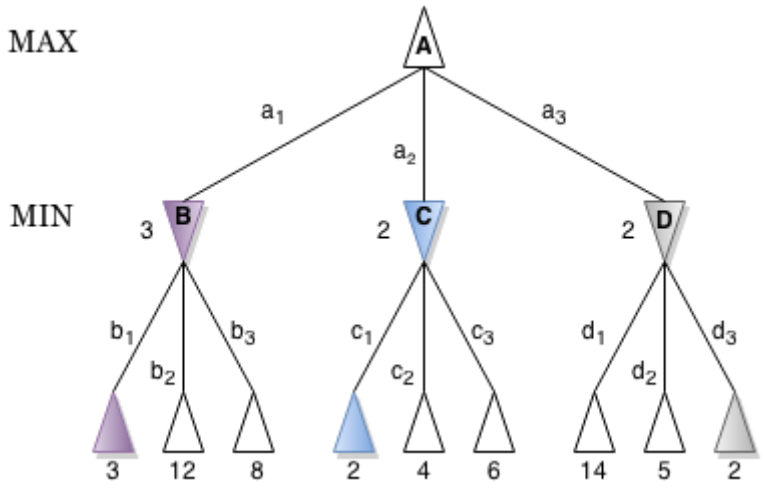
Igas tipus, kus meie oleme käigul, püüame teha parima käigu, s.t liikuda tippu, mille väärtus on võimalikult suur. Seevastu kui käigul on vastane, siis tema püüab valida sellise käigu, mis on parim tema jaoks, kuid halvim meie jaoks, s.t. võimalikult väikese hinnangufunktsiooni väärtuse. Seetõttu nimetataksegi mängijaid minimeerijaks ja maksimeerijaks ning meetodit minimax-meetodiks.[9]

Vaatleme, kuidas toimub tippude väärtustamine puus kasutades minimax protseduure.



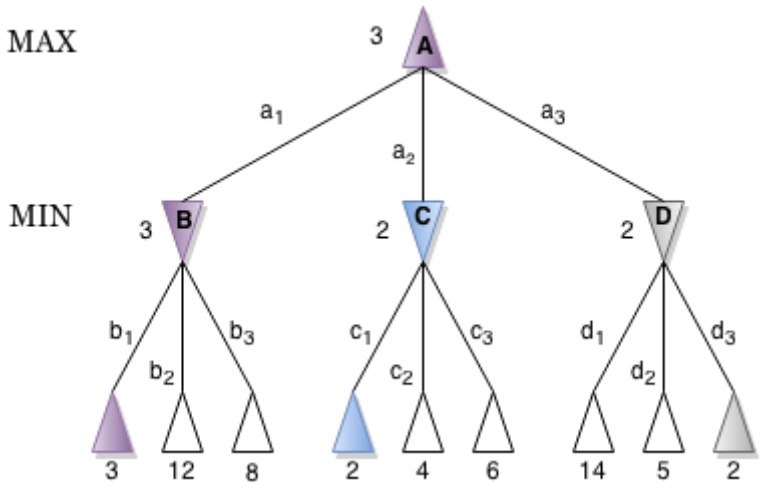
Joonis 12. (a) Minimax mängupuu algseis.

Joonisel 12 väärtustatakse esmalt minimeerija taseme tipud. Iga minimeerija tipu väärtuseks valitakse vähim tema järglaste väärtustest.



Joonis 13. (b) Minimeerija taseme tippude väärtusteks on valitud vähim nende järglaste väärtustest.

Joonisel 13 kujutatud minimeerija taseme tippude detailne väärtustamisprotsess on järgnev: $v_b = min(3,12,8) = 3$; $v_c = min(2,4,6) = 2$; $v_d = min(14,5,2) = 2$, kus v_x, $x \in \{B,C,D\}$, tähistab vastavalt tippude *B*,*C* ja *D* väärtust.



Joonis 14. Maksimeerija tipu A väärtuseks valitakse maksimaalne tema järglaste väärtustest.

Joonisel 14 kujutatud maksimeerija taseme tipu detailine väärtustamisprotsess on järgnev: $v_a = max(v_b,v_c,v_d) = max(3,2,2) = 3$.

Minimax algoritm [9]

Hindamaks tippu *n* mängupuul, tee järgmist:

1. Olgu $L=\{n\}$ – läbimata tippude nimestik.

2. Olgu *x* esimene tipp nimestikust *L*. Kui $x = n$ ja talle on omistatud väärtus, siis tagasta see väärtus.

3. Kui *x* väärtus on v_x, siis olgu *p* tipu *x* vahetu eellane ning v_p tema jooksev väärtus.

- Kui *p* on minimeerija tipp, siis võta $v_p=min(v_p,v_x)$.
- Kui *p* on maksimeerija tipp, siis võta $v_p=max(v_p,v_x)$.

Eemalda *x* nimestikust *L* ja mine *2*.

4. Kui tipule *x* pole omistatud väärtust ja ta on terminaalne tipp, siis omista talle väärtus

- 1 (kui see on maksimeerija võit)
- -1 (kui see on minimeerija võit)
- 0 (kui see on viik)

Jäta *x* nimestikku *L* (et käsitleda tema vahetut eellast) ja mine *2*.

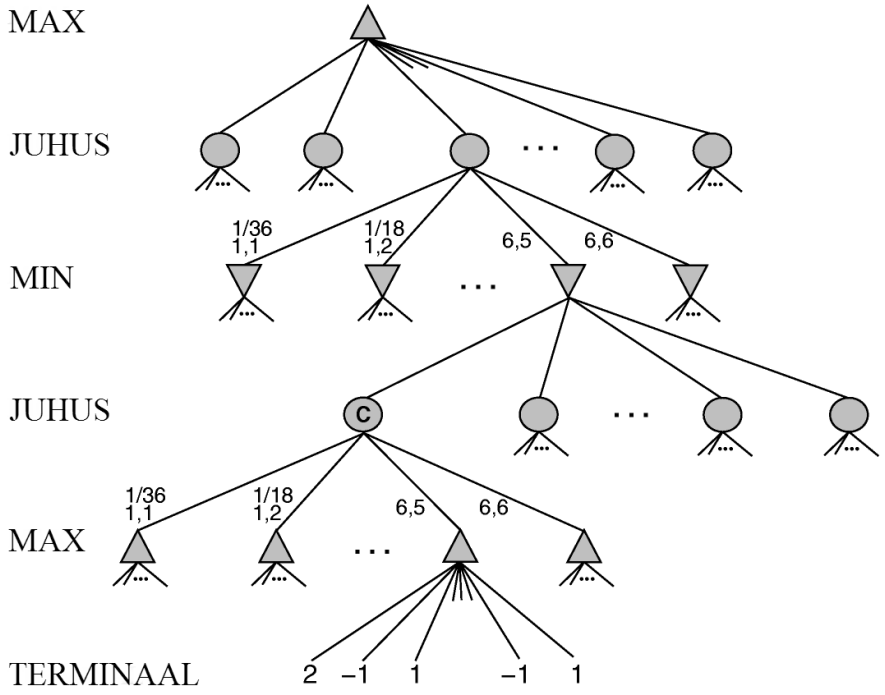
5. Kui tipule *x* pole omistatud väärtust ja ta pole lõpptipp, siis olgu:

- $v_x = -\infty$ (kui *x* on maksimeerija tipp)
- $v_x = +\infty$ (kui *x* on minimeerija tipp).

Lisa *x* vahetud järglased nimestikku *L* algusesse ja mine 2.

2.2 Expect-minimaxi algoritm

Kuidas peaks välja nägema mängupuu aga juhuslikkuse elementi sisaldavate kahe isiku mängude puhul? Triktrakis saab mängija enda käigule järgnevaid vastase käike hinnata vaid teatud tõenäosusega, kuna vastase võimalikud käigud selguvad alles pärast täringu veeretamist. Seega tuleb mänguseisude puu konstrueerimisel arvesse võtta ka täringute veeretamisest tulenevat juhust.

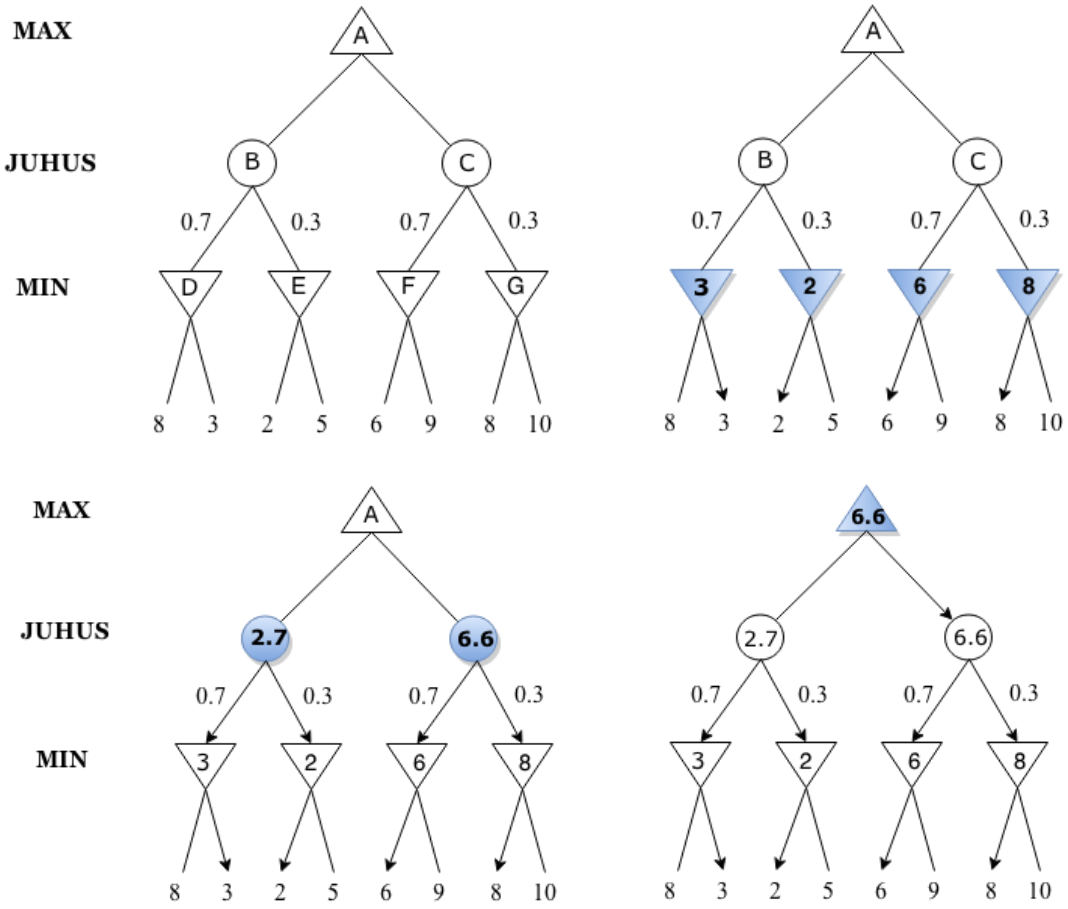


Joonis 15. Triktraki mängupuu

Joonisel 15 on juhuslikkuse tipud tähistatud ringidena. Igast juhuslikkuse tipust väljuvad servad tähistavad üht võimalikku täringusilmade kombinatsiooni ning tõenäosust, et selline kombinatsioon aset leiab. Kahe täringu veeretamise võimalikke tulemusi on kokku $6 \cdot 6 = 36$ ning iga selline kombinatsioon leiab aset võrdse tõenäosusega. Kuna aga näiteks silmade arv 6-5 on sisuliselt sama, mis 5-6, on järjestust arvesse võtmata erinevaid veeretustulemusi 21. Kuue duubli (1-1 kuni 6-6) esinemise tõenäosus on 1/36, ülejäänud 15 erineva täringukombinatsiooni esinemistõenäosus on 1/18. [8]

Järgmise sammuna peame selgeks tegema, kuidas langetada õigeid otsuseid. Loomulikult soovime me endiselt sooritada käigu, mis viib parima tulemuseni. Kuna triktrakis ei tea me legaalseid käike enne täringu veeretamist, saame mingi käigu sooritamise perspektiivikust hinnata arvutades **keskväärtuse** üle kõikvõimalike täringukombinatsioonide. Nii jõuamegi minimaxi algoritmi modifikatsioonini **expect-minimax** juhuslikkuse elementi sisaldavate mängude jaoks. Terminaalsete ning minimeerija ja maksimeerija tippude väärtustamine toimib täpselt samamoodi kui minimaxi algoritmis. Juhuslikkuse tippe hinnatakse aga kõikide järglaste väärtuste keskväärtuse leidmise teel. Juhul, kui kõik tipud on võrdse tõenäosusega, on juhuslikkuse tipu väärtuseks lihtsalt järglastippude aritmeetiline keskmine.

Vaatleme, kuidas toimub tippude väärtustamine expect-minimax algoritmi kasutades lihtsustatud mängupuul, kus mängija võimalikud käigud määrab ära sellise mündi vise, mille tulemuseks 70% juhtudest on kull ning 30% juhtudest kiri.



Joonis 16. Expect-minimaxi mängupuu väärtustamine.

.

Minimeerija taseme tippude väärtustamine toimub analoogselt tavalisele minimaxile, kus tipu väärtuseks valitakse minimaalne tema järglaste väärtustest. Seega toimub joonisel 16 tippude *D*, *E*, *F* ja *G* väärtustamine järgnevalt:

$v_d = min(8,3) = 3$; $v_e = min(2,5) = 2$; $v_f = min(6,9) = 6$; $v_g = min(8,10) = 8$, kus $v_{x,}$ $x=\{D,E,F,G\}$, tähistab vastavalt tippude *D*, *E*, *F* ja *G* väärtust.

Samal joonisel esitatud juhuslikkuse tippude *B* ja *C* väärtustamisel tuleb leida aga nende tippude järglaste keskväärtus:

$v_b = \sum v_x \cdot P(X) = 0.7 \cdot D + 0.3 \cdot E = 0.7 \cdot 3 + 0.3 \cdot 2 = 2.1 + 0.6 = 2.7$ (kus $x \in \{D,E\}$)

$v_c = \sum v_x \cdot P(X) = 0.7 \cdot F + 0.3 \cdot G = 0.7 \cdot 6 + 0.3 \cdot 8 = 4.2 + 2.4 = 6.6$ (kus $x \in \{F,G\}$)

Maksimeerija tipus A valitakse väärtuseks taaskord lihtsalt maksimaalne tema järglaste

väärtustest.

$v_a = max(v_b, v_c) = max(2.7, 6.6) = 6.6$

Modifitseerides minimaxi algoritmi [9], et see võtaks arvesse ka juhuslikkuse tippe, saame formaalselt kirja panna expect-minimaxi algoritmi [8].

Expect-minimaxi algoritm

Hindamaks tippu n mängupuul, tee järgmist:

1. Olgu $L=\{n\}$ – läbimata tippude nimestik.

2. Olgu x esimene tipp nimestikust L. Kui $x = n$ ja talle on omistatud väärtus, siis tagasta see väärtus.

3. Kui x väärtus on v_x, siis olgu p tipu x vahetu eellane ning v_p tema jooksev väärtus.

- Kui p on minimeerija tipp, siis võta $v_p = min(v_p, v_x)$.
- Kui p on maksimeerija tipp, siis võta $v_p = max(v_p, v_x)$.
- **Kui p on juhuslikkuse tipp, siis võta $v_p = v_p + P(x) \cdot v_x$,**
  (kus $P(x)$ tähistab tippu x jõudmise tõenäosust).

Eemalda x nimestikust L ja mine *2*.

4. Kui tipule x pole omistatud väärtust ja ta on terminaalne tipp, siis omista talle väärtus

- 1 (kui see on maksimeerija võit)[1]
- -1 (kui see on minimeerija võit)

Jäta x nimestikku L (et käsitleda tema vahetut eellast) ja mine *2*.

5. Kui tipule x pole omistatud väärtust ja ta pole lõpptipp, siis olgu:

- $v_x = -\infty$ (kui x on maksimeerija tipp)
- $v_x = +\infty$ (kui x on minimeerija tipp)
- **$v_x = 0$ (kui x on juhuslikkuse tipp)**

Lisa x vahetud järglased nimestikku L algusesse ja mine 2.

---

[1] Terminaalsetele tippudele omistatavad väärtused võivad sõltuvalt mängust olla ka midagi muud. Topeldamistäringuga triktrakis näiteks [-192,192].

2.3 Expect-minimaxi ajaline keerukus

Minimax teostab mängupuul täieliku süvitsiotsingu. Kui puu maksimaalne sügavus on *m* ja igas tipus on *b* legaalset käiku, on minimaxi ajaline keerukus $O(b^m)$. Mäluvajadus on *O(bm)* algoritmi jaoks, mis genereerib kõik järglased korraga või *O(m),* mis genereerib järglased ükshaaval. [9]

Kui me teaks ette kõiki täringuveeretusi (või muude juhuslikkuse elementide tulemusi), mis mängu jooksul aset leiavad, oleks **expect-minimaxi ajaline keerukus** täpselt sama, mis minimaxi algoritmi puhul: $O(b^m)$. Kuna expect-minimax peab aga lisaks läbi vaatama ka kõikvõimalikud täringukombinatsioonid, kulub aega $O(b^m n^m)$, kus *n* on erinevate täringuveeretuste arv. Triktrakis on $n = 21$ ja hargnemistegur *b* tavaliselt umbes 20, aga vahepeal, kui täringuga veeretatakse duubel, võib see küündida koguni 4000-ni.[8] See tähendab, et isegi väikese puusügavuse korral võtab kõikide tippude läbi vaatamine liiga kaua aega, et otsingu sellisel kujul rakendamine end praktikas ära tasuks.

Siiski on võimalik kasutada erinevaid meetodeid otsingu kiiruse parandamiseks. Vaatleme lähemalt kahte kõige sagedamini kasutatavat: puu kärpimist alfa-beeta algoritmi abil ning hinnangufunktsiooni rakendamist.

2.4 Alfa-beeta kärpimine

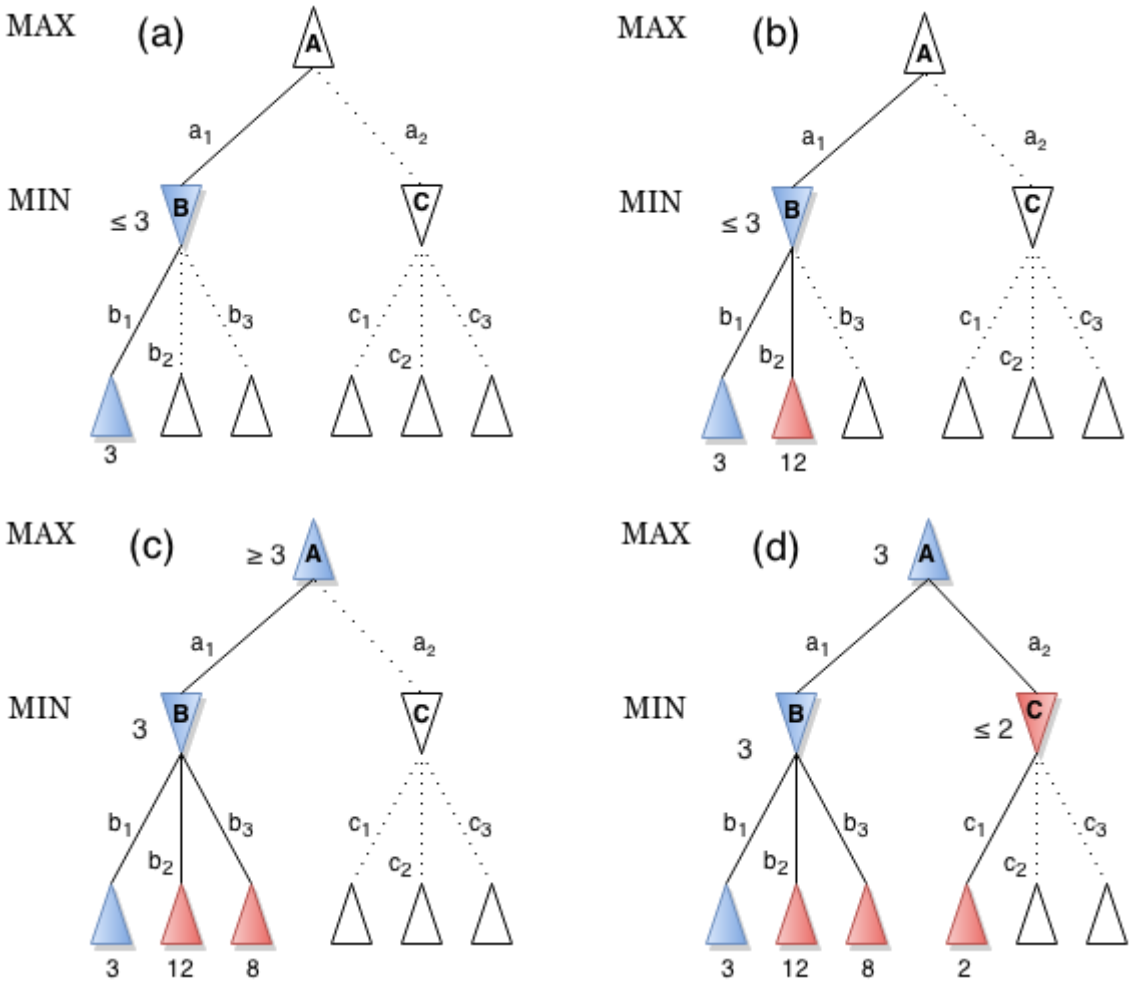
Üheks minimaxi otsingu suurimaks probleemiks on eksponentsiaalne ajavajadus. Kuigi me ei saa eksponenti täielikult eemaldada, on meil võimalik seda poole võrra vähendada. Tehnika, mille me vaatluse alla võtame, kannab nime **alfa-beeta kärpimine.** Standardsele minimax mängupuule rakendades tagastab see sama käigu, mille tagastaks minimaxi algoritm, aga kärbib välja harud, mis lõplikku otsust kuidagi ei mõjuta. [8]

Alfa-beeta kärpimist on võimalik rakendada mistahes sügavusega puudele ning sageli on lehtede asemel võimalik kärpida terveid alampuid. Algoritmi üldine põhimõte on järgmine: Võtame vaatluse alla mängupuu tipu *n*. Kui mängijal on võimalik teha parem valik mistahes tipule *n* eelnevas tipus, siis tipuni *n* tegelikus mängus ei jõutagi. Seega, kui oleme tipu *n* kohta piisavalt infot kogunud (läbi vaadates osad tema järglastest), et sellele otsusele jõuda, võime *n*'i mängupuust kärpida. Alfa-beeta kärpimine on oma nime saanud kahelt parameetrilt, mis seavad piirid puuharudes esinevatele väärtustele: [8]

α = parima valiku väärtus, mille seni maksimeerija teel leidnud oleme;

β = parima valiku väärtus, mille seni minimeerija teel leidnud oleme.

Vaatleme, kuidas töötab alfa-beeta kärpimine minimaxi mängupuul.



Joonis 17. Alfa-beeta kärpimine.

Pärast esimese lehttipu väärtustamist joonisel 17 kujutatud mängupuus (a) saame teada, et tipu *B* väärtus on maksimaalselt 3, kuna tegutseme minimeerija tasemel. Järgmise lehttipu väärtustamisel *B* väärtus ei muutu, kuna 12 pole väiksem kui 3. Väärtustanud ka viimase lehttipu harus b_3, saame teada, et *B* väärtus on täpselt 3, kuna ka viimase tipu väärtus polnud väiksem kui 3. Nüüd aga saame teada, et tipu *A* väärtus ei saa olla väiksem kui 3, kuna selleks valitakse maksimaalne tema järglaste väärtustest. Võtame nüüd vaatluse alla tipu *C* järglased. Tuleb välja, et neist esimese väärtus on *2* ning seega peab olema minimeerija taseme tipu *C* väärtus väiksem või võrdne kahega. Kuna maksimeerija tipus *A* valib aga oma vahetute järglaste väärtustest maksimaalse, ei ole meil mõtet otsingut jätkata, kuna näeme, et *C* väärtus on sõltumata tema ülejäänud järglaste väärtustest väiksem kui *B* oma, seega valib *A* kindlasti *B* väärtuse.

Alfa-beeta otsing uuendab α ja β väärtusi ning kärbib tippudest tulenevaid harusid niipea kui vaatluse all oleva tipu väärtus on halvem kui jooksev alfa või beeta väärtus.[8] Maksimeerivatel tasemetel võime käigu elimineerida, kui selgub, et tema väärtus on väiksem kui jooksev lävi, minimeerivatel tasemetel võime aga katkestada otsingu niipea, kui avastame väärtuse, mis on suurem kui jooksev lävi. [9]

Alfa-beeta kärpimist kasutav minimaxi algoritm [9]

Hindamaks tippu n mängupuul, tee järgmist:

1. Olgu $L = \{n\}$

2. Olgu x esimene tipp nimestikust L. Kui $x = n$ ja talle on omistatud väärtus, siis tagasta see väärtus.

3. Kui tipule x pole omistatud väärtust, siis mine 5. Kui tipule x on omistatud väärtus v_x, siis olgu p tipu x vahetu eellane. Kõigepealt tuleb määrata, kas tipu p ja tema järglased saab puust kärpida: kui p on minimeerija tipp, siis on α maksimaalne p kõigi vahetute naabrite ja p eellasteks olevate minimeerija tippude kõigi vahetute naabrite jooksvatest väärtustest. (Kui selliseid väärtusi ei ole, siis võta $\alpha = -\infty$.) Kui $v_x \leq \alpha$, siis eemalda tipp p ja kõik tema järglased nimestikust L (*α-kärbe*).

Analoogselt toimi juhul, kui p on maksimeerija tipp: olgu β minimaalne p kõigi vahetute naabrite ja p eellasteks olevate maksimeerija tippude kõigi vahetute naabrite jooksvatest väärtustest. (Kui selliseid väärtusi ei ole, siis võta $\beta = +\infty$.) Kui $v_x \geq \beta$, siis eemalda tipp p ja kõik tema järglased nimestikust L (*β -kärbe*).

4. Kui tippu p koos tema järglastega ei saa kärpida, siis olgu v_p tipu p jooksev väärtus. Kui p on minimeerija tipp, siis võta $v_p = min(v_p, v_x)$. Kui p on maksimeerija tipp, siis võta $v_p = max(v_p, v_x)$. Eemalda x nimestikust L ja mine 2.

5. Kui tipule x pole omistatud väärtust ja ta on kas terminaalne tipp või sa oled otsustanud puud tipust x allapoole mitte läbida, siis arvuta tipu x väärtus, kasutades hinnangufunktsiooni, ja mine 2.

6. Vastasel korral võta $v_x = -\infty$ (kui x on maksimeerija tipp) või $x = \infty$ (kui x on minimeerija tipp). Lisa x kõik vahetud järglased nimestiku L algusesse ja mine 2.

Alfa-beeta kärpimise efektiivsus on suuresti sõltuv käikude läbi vaatamise järjekorrast. Kui joonisel 18 oleks tipu *B* alamad olnud väärtustega 5,4,2 ning need oleks ka vastavas järjekorras läbi vaadatud, poleks kärpimisel olnud mingit mõtet, kuna tipu *A* väärtuse teada saamiseks oleks tulnud läbi vaadata ikkagi kõik puu tipud.

2.2.1 Alfa-beeta kärpimise ajavajadus

Harvendamine võib taandada otsinguruumi suurust. Halvimal juhul on võimalik, et kärpimine ebaõnnestub – kui järjestame iga tipu vahetud järglased nii, et esimesena kontrollitakse halvimat. Parimal juhul kontrollitakse parimaid käike esimesena. Vaatame minimeerija vastust, mis pole temale parim. Et seda saaks kärpida, tuleb meil kontrollida nii palju otsinguruumi, et saaksime väita, et minimeerija jaoks on see käik viga – seega kontrollime selle käigu „eitust", mis näitab, kuidas maksimeerija saab seda ära kasutada. Et nii toimida, kontrollime ühtainsat maksimeerija vastust – parimat. Siis kontrollime minimeerija kõiki valikuid, kuid ainult maksimeerija parimat vastust igaühele neist, jne. Hargnemistegur minimeerija jaoks on b (puu eeldatav hargnemistegur), maksimeerija jaoks aga 1. Kui vaadata protsessi minimeerija seisukohast, siis on analüüs analoogne, üksnes mängijate rollid on vahetatud. Tippude koguarv, mida tuleb kontrollida sügavusel d, on ligikaudu $b^{d/2} + b^{d/2} = 2b^{d/2}$ (mitte aga b^d). [9] Seega oleme saanud ajalist keerukust $O(b^d)$ parandada väärtuseni $\boldsymbol{O(b^{d/2})}$. See tähendab, et hargnemisteguriks on b asemel $\sqrt{b}$, malemängus saame selleks 35 asemel 6. Teisisõnu, alfa-beeta kärpimist kasutav otsing võib vaadata kaks korda kaugemale kui minimax sama aja jooksul. Kui käike ei järjestata paremuse järgi, vaid need vaadatakse läbi juhuslikus järjekorras, on läbivaadatavate käikude arv ligikaudu $O(b^{3d/4})$. [8] Seega on harvendamise puhul oluline, kuidas tipu vahetuid alluvaid järjestada.

2.2.2 Kärpimine juhuslikkuse tippudega puul

Ka juhuslikkust sisaldavate mängude mängupuul on võimalik rakendada midagi alfa-beeta kärpimise taolist.

Minimeerija ja maksimeerija tippude analüüs jääb samaks nagu eelnevalt käsitletud alfa-beeta kärpimise juures. Kuidas aga kärpida juhuslikkuse tippe? Võtame vaatluse alla tipu *C* joonisel 15 ning selle, kuidas sõltub tema väärtustamine järglaste väärtustest. Kas on

võimalik leida ülemine lävi *C*-le enne, kui oleme läbi vaadanud kõik tema järglased? Esmapilgul võib see tunduda võimatu, kuna *C* väärtus on tema järglaste väärtuste keskväärtus. Enne kui me oleme läbi vaadanud kõikvõimalike täringuveeretuste tulemused, ei saa me keskväärtuse tulemust teada, kuna selle leidmiseks on vaja arvesse võtta kõikide järglaste väärtuseid, need võivad olla aga mistahes suurused. Kui me piiritleme aga kasulikkusfunktsiooni väärtused, siis võime leida läve ka keskväärtuse jaoks.

Gamma kärpimine

Juhuslikkuse tippudele mängupuus on võimalik rakendada näiteks gamma kärpimise algoritmi. Algoritmi idee on järgnev: juhuslikkuse tippude väärtustamata järglased hinnatakse mängija jaoks parimaks võimalikuks käiguks. Kui sellise valiku keskväärtus on halvem kui mõnes teises puuharus leitud senine parim väärtus, siis kärbitakse alampuu, mille harus hetkel tippude väärtustamine pooleli oli.[10]

Pythoni jaoks kohandatud gamma kärpimise pseudokood on järgnev [10]:

```
def gamma_prune():

    # v_min = -1
    # v_max = 1
    if (isMax(parent(chance_node))): # juhuslikkuse tipu vanem
                                     # on maksimeerija tipp
        v = -1000
    if (isMin(parent(chance_node))): # juhuslikkuse tipu vanem
                                     # on minimeerija tipp
        v = 1000

    for (chance_node in parent(chance_nodes)):
        s = 0 # keskväärtus
        p = 0 # tõenäosus
        for (child in chance_node):

            s = s+ v_child*P(child) # läbi vaadatud naabertippude
                                    # keskväärtus
            p = p + probability(child) # läbi vaadatud naabertippude
                                       # tõenäosuste summa

            if (isMax(parent(chance_node)):

                # eeldatakse, et kõik seni läbi vaatamata
                # tipud on parima võimaliku väärtusega ning
                # nende keskväärtus liidetakse seni välja
                # arvutatud keskväärtusele s
                alpha = s + (1-p)*v_max

                # kui alfa on väiksem kui seni leitud

                # parim v, tähendab, et mingis teises harus on
```

```
                # leitud juba parem maksimaalne väärtus
                # ning isegi kui kõikidel läbi vaatamata
                # tipu chance_node järglastel on maksimaalne
                # väärtus, ei saavutada tipus piisavalt suurt
                # väärtustet maksimeerija tase selle valiks
                if alpha <= v:
                    break  # kärbi puu

            if (isMin (parent(chance_node))):

                # eeldatakse, et kõik seni läbi vaatamata
                # tipud on parima võimaliku väärtusega ning
                # nende keskväärtus liidetakse seni välja
                # arvutatud keskväärtusele s
                beeta = s + (1-p)*v_min

                # kui beeta on suurem kui seni leitud parim
                #(vähim) v, tähendab, et mingis teises harus on
                #leitud juba parem minimeerija väärtus ning isegi
                #kui kõikidel läbi vaatamata tipu chance_node
                #järglastel on minimaalne väärtus, ei saavutada
                #tipus piisavalt väikest väärtust et minimeerija
                #tase selle valiks
                if beeta >= v:
                    break   # kärbi puu
        if (isMax(parent(chance_node)):
            v = max(alpha,v)
        if (isMin(parent(chance_node)):
            v = min(beeta,v)
    return v
```

2.5 Hinnangufunktsioon

Isegi kui me kärbime otsingupuud, peab alfa-beeta algoritm vähemalt osaliselt sooritama otsingu puu lõpptippudeni välja. Seesugune sügavus ei ole aga enamikel juhtudes praktiline, kuna käigud tuleb sooritada mõistliku aja vältel – tavaliselt kõige enam paari minuti jooksul. Claude Shannon käis 1950. aastal välja idee, et programm võiks otsingu lõpetada varem ning staatilist hinnangufunktsiooni kasutades väärtustada mitteterminaal-sed tipud nagu kasulikkusfunktsioon väärtustab puu tegelikke lõpptippe.[8]

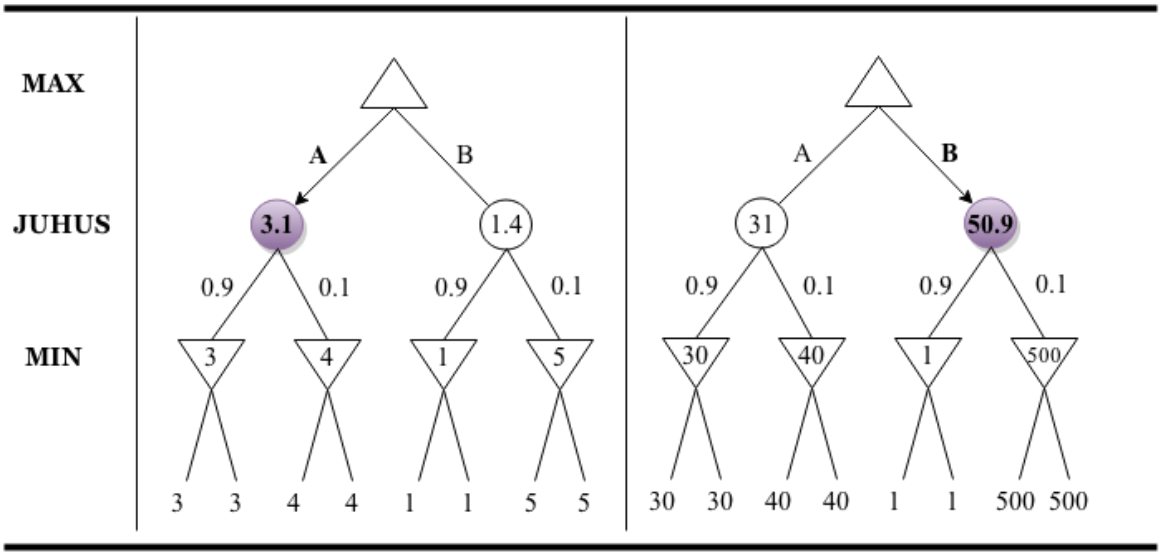
Hinnangufunktsioon tagastab hinnangu käigu eeldatavast tulemuslikkusest mingist kindlast positsioonist. Mänguseisule hinnangu andmise idee ei olnud uus, kui Shannon selle esitas Nii male kui ka paljude teiste mängude mängijad on sajandite vältel leidnud viise, kuidas hinnata mängupositsiooni viljakust, kuna inimesed on otsingu sügavuse suhtes veel limiteeritumad kui arvutiprogrammid. Mänguprogrammi tulemus-likkus on sõltuv kasutatava hinnangufunktsiooni kvaliteedist. Ebatäpne hinnangu-funktsioon võib

viia mängija seisu, mis osutub lõpuks kaotuseks. Kuidas täpsemalt disainida häid hinnangufunktsioone? Silmas tuleks pidada kolme aspekti.

1. Hinnangufunktsioon peab järjestama terminaalsed seisud samamoodi kui tõeline kasulikkusfunktsioon, kuna vastasel juhul võib mänguagent valida käigu, mis pole optimaalne, isegi kui tal on võimalik näha mängu lõppseisuni.
2. Arvutamine ei tohiks võtta liiga kaua aega, kuna muidu kaotab hinnangufunktsioon kogu oma mõtte.
3. Mitteterminaalsete mänguseisude jaoks peaks hinnangufunktsioon olema tugevas korrelatsioonis päris võiduvõimalustega.

Enamus hinnangufunktsioone töötavad erinevate mänguseisude erinevaid karakteristikuid arvutades – malemängus näiteks mõlema osapoole valduses olevate etturite arvu kokku lugedes. Erinevate karakteristikute arvulised väärtused loetakse kokku ning kombineeritakse seejärel, et leida hinnangufunktsiooni koguväärtust. Näiteks kasutatakse malemängus sageli hinnanguna igale malendile antavat arvulist väärtust: iga ettur on väärt 1, ratsu ja oda 3, vanker 5 ning lipp 9. Lisaks võetakse tihtilugu arvesse ka selliseid faktoreid nagu „hea etturite struktuur“ või „kuninga ohutus“. Seejärel liidetakse need väärtused kokku, et leida hinnang tervele mängupositsioonile. [8] Triktrakis sagedamini kasutatavad hinnangud on nuppude arv müüril ning järjestikuste suletud kiilude arv.

Võiks ju arvata, et trikiktraki-taoliste mängude hinnangufunktsioon peaks olema samasugune nagu näiteks malel – parematele positsioonidele tuleb lihtsalt kõrgem hinnang anda. Ilmneb aga, et juhuslikkuse tippude olemasolu tõttu tuleb hinnangufunktsiooni väärtustega ettevaatlikult ümber käia. Joonisel 18 on näha, mis juhtuda võib: hinnangufunktsiooniga, mis määrab lehtedele väärtused (1,3,4,5), valib maksimeerija tipp haru A. Väärtustades lehed aga arvudega (1,30,40,500), on eelistatud haru hoopis B, olgugi, et lehttippude järjestus on sama. [8]



Joonis 18. Lehttippude järjestust säilitav väärtuste muutmine muudab valitavat käiku

Modifitseerides hinnangufunktsiooni kasutavat minimax algoritmi [9], saame kirja panna hinnangufunktsiooni kasutava expect-minimax algoritmi.

Excpect-minimax algoritm, mis kasutab hinnangufunktsiooni

Hindamaks tippu n mängupuul, tee järgmist:

1. Olgu $L=\{n\}$ – läbimata tippude nimestik.

2. Olgu x esimene tipp nimestikust L. Kui $x = n$ ja talle on omistatud väärtus, siis tagasta see väärtus.

3. Kui x väärtus on v_x, siis olgu p tipu x vahetu eellane ning v_p tema jooksev väärtus.

- Kui p on minimeerija tipp, siis võta $v_p = min(v_p,v_x)$**.**
- Kui p on maksimeerija tipp, siis võta $v_p=max(v_p,v_x)$**.**
- Kui p on juhuslikkuse tipp, siis võta $v_p= v_p + P(x) \cdot v_x$,
  (kus $P(x)$ tähistab tippu x jõudmise tõenäosust).

Eemalda x nimestikust L ja mine *2*.

4. Kui tipule x pole omistatud väärtust ja ta on terminaalne tipp või oled otsustanud puud temast allapoole mitte läbida, siis arvuta tema väärtus, kasutades hinnangufunktsiooni ja,

mine 2.

5. Vastasel korral võta

- $v_x = -\infty$ (kui *x* on maksimeerija tipp)
- $v_x = +\infty$ (kui *x* on minimeerija tipp)
- $v_x = 0$ (kui *x* on juhuslikkuse tipp)

Lisa *x* vahetud järglased nimestikku *L* algusesse ja mine 2.

Pythoni jaoks kohandatud hinnangufunktsiooni kasutava expect-minimaxi algoritmi pseudokood on järgmine [11]:

```
def expectminimax(node,depth):
    if (isTerminal(node) or depth == 0):
        return heuristic_value(node)
    if (isMin(node)):
        # Return the value of minimum-valued child node
        vx = + 10000
        for child in (children(node)):
            vx = min(vx,expectminimax(child,depth-1))
    else if (isMax(node)):
        # Return the value of maximum-valued child node
        vx = - 10000
        for child in (children(node)):
            vx = max(vx,expectminimax(child,depth-1))
    else if (isRandom(node)):
        # Return the weighted average of all child nodes' values
        vx = 0
        for child in (children(node)):
            vx = vx + P(child)* expectminimax(child,depth-1)

    return vx
```

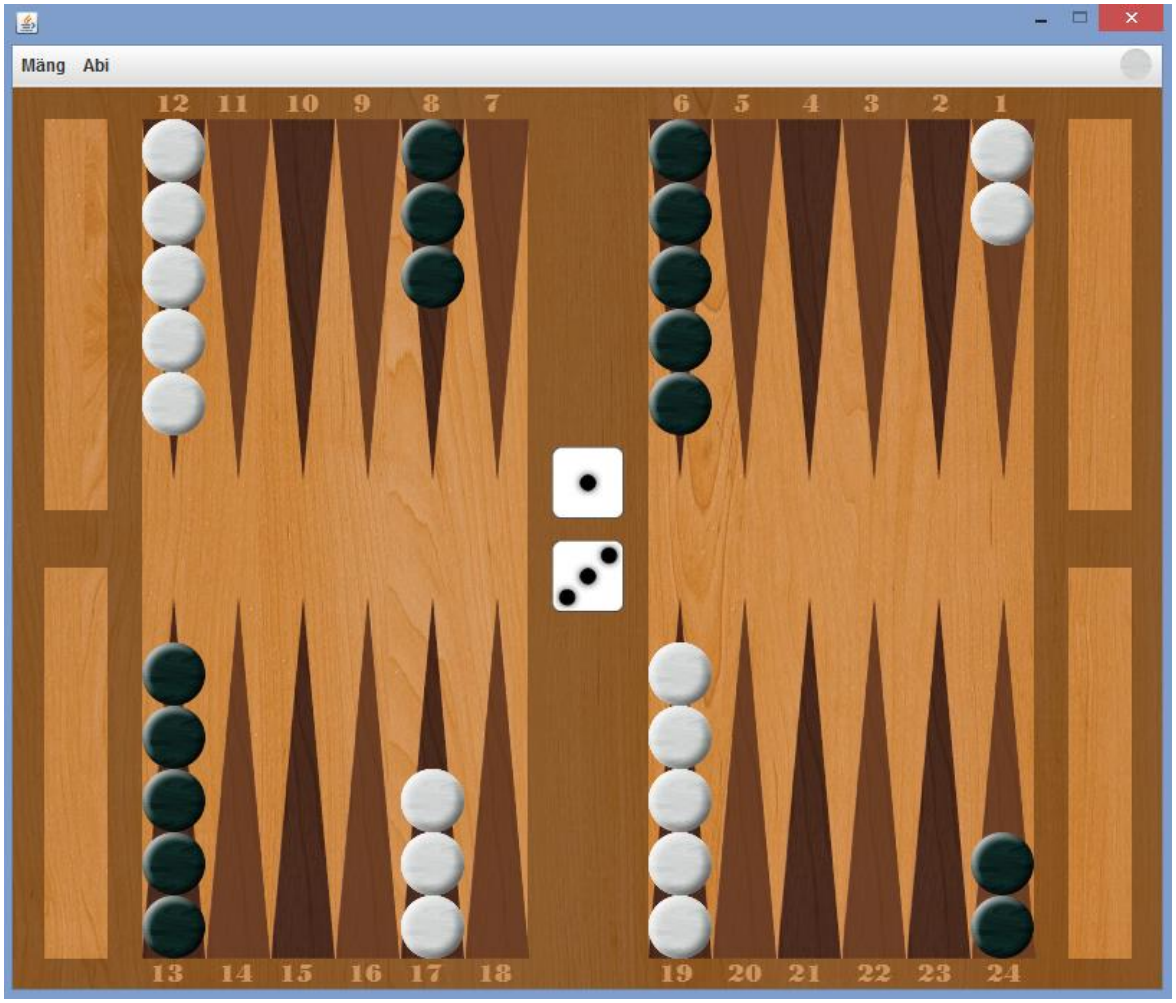
3 Programmi ülevaade

Selles peatükis antakse ülevaade bakalaureusetöö käigus koostatud programmi võimalustest ja implementatsiooni käigus tehtud valikutest.

3.1 Programmi tutvustus

Eesmärk

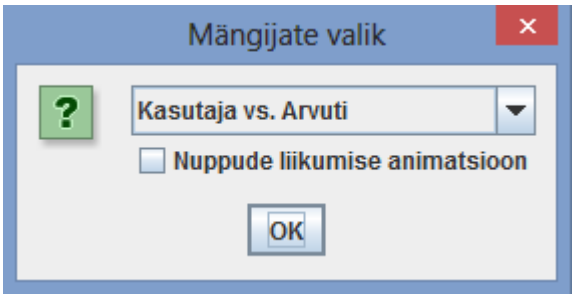
Programmi eesmärgiks on aidata paremini mõista expect-minimaxi algoritmi strateegia ja mängupositsiooni hindamist triktraki mängu näitel. Programm võimaldab valida erinevaid algoritmi töös rakenduvaid parameetreid, nagu puu sügavus, strateegia, ja kärpimise teostamine ning seejärel jälgida, kuidas need mõjutavad mängu tulemust. Programmi avaaken on näha joonisel 19.



Joonis 19. Programmi avaaken algseisuga.

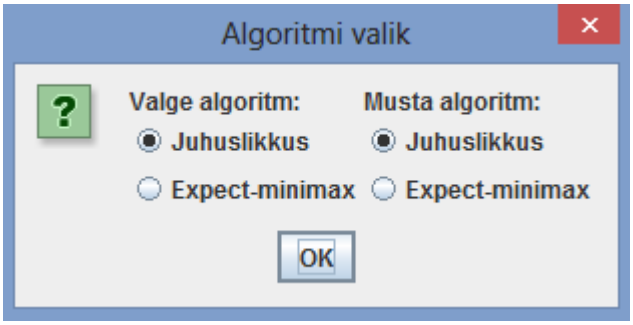
Võimalused

Programmi käivitades on kasutajal võimalik valida kolme erineva mängijate kombinatsiooni vahel (vt. joonis 20). Esimene neist – kaks kasutajat, võimaldab mängijal triktraki mängu ning programmiga tutvust teha ja oma kontrolli all olevas keskkonnas erinevaid käike katsetada; teise võimalusena saab kasutaja arvuti vastu kätt proovida ning kolmanda valikuna võib kasutaja vaadelda arvuti algoritmide omavahelist mänguprotsessi.



Joonis 20. Mängijate valiku aken.

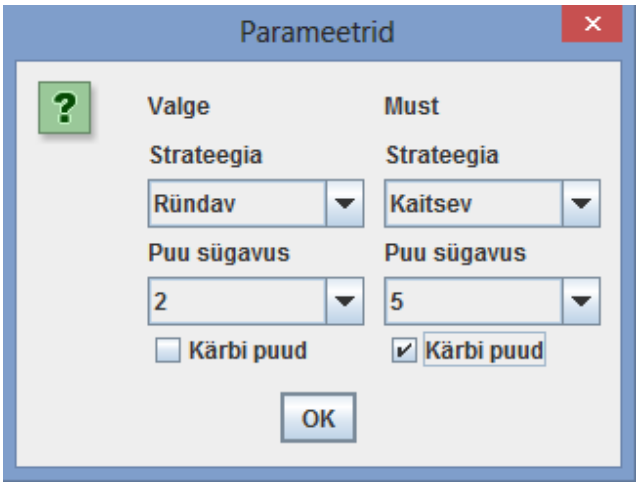
Juhul, kui vähemalt üks kasutaja valitud mängijatest on arvuti, saab kasutaja valida algoritmi, mida arvuti mängimiseks kasutama hakkab (vt. joonis 21). Valiku saab langetada kahe algoritmi vahel: expect-minimax ning juhuslikkuse algoritm, mis iga oma mängukorra ajal täiesti juhusliku käigu sooritab. Juhuslikkuse algoritmi eesmärgiks on illustreerida, kuidas sõltub mängu tulemus laual oleva informatsiooni kasutamisest ehk kui suur roll on mängus edu saavutamisel täringu veeretamisest tuleneval juhusel ning kui suur osa strateegial ja mängupositsiooni hinnangul.



Joonis 21. Algoritmi valiku aken.

Expect-minimax algoritmi valimisel on kasutajal lisaks võimalik valida puu sügavus (1-8), strateegia, ning kas kärpida otsingupuud või mitte (vt. joonis 22). Optimaalne puu sügavus jääb vahemikku [1,4], kuid suuremate sügavuste valimine aitab paremini illustreerida

otsingu ajakulu puu sügavuse kasvades. Arvuti algoritmi valimise puhul saab lisaks määrata, kas näidata nuppude liikumise animatsiooni või mitte. Nuppude liikumise animatsioon on soovitatav sisse lülitada, kui kasutajal on soov saada täpne ülevaade mängus sooritatavatest käikudest, kuna näiteks juhuslikkuse algoritmide omavaheline mäng saab vastasel juhul mõne sekundi jooksul läbi ning detailset ülevaadet mängu käigus toimuvast on üsna raske saada. Nuppude liikumisse kuvamise välja lülitamine sobib aga hästi juhtudeks, kui kasutaja soovib teada ainult mängu tulemust ning selle saavutamise detailid teda ei huvita. Näiteks erinevate parameetrite kohta statistikat kogudes on hea, kui programm töö kiiresti lõpetab.



Joonis 22. Expect-minimaxi parameetrite valiku aken.

Mängu lõppedes kuvatakse teade, kumb mängijatest võitis ning tema teenitud punktide arv (vt. joonis 23). Võitja teenib ühe punkti, kui vastane on jõudnud nuppude maha kandmist alustada; kaks punkti, kui vastane pole veel nuppude maha kandmist alustada jõudnud ning kolm punkti, kui vastane pole nuppude maha kandimist alustada jõudnud ning lisaks sellele on vähemalt üks vastase nuppudest veel võitja siselaual.

Joonised 23. Mängu lõpu teavutusaknad.

3.2 Implementeerimise käigus tehtud valikud

Mängus on implementeerimata jäetud topeldamistäring ning erinevalt klassikalistest reeglitest, kus alustaja selgub pärast täringu veeretamist, alustab lihtsuse huvides alati valge mängija. Kui expect-minimax algoritmi kasutava mängija võimalike käikude arv mängija käimiskorra ajal on üks, siis expect-minimaxi otsingut välja ei kutsuta, kuna sellisel juhul ei ole puul otsingu sooritamisel mingit mõtet.

Tähelepanu tuleks pöörata ka sellele, et kui täringuga veeretatakse duubel, läheb expect-minimaxil üldiselt tunduvalt kauem aega, kuna võimalike käikude arv on siis üldjuhul suurem ning seega suureneb ka puu hargnemistegur tunduvalt.

3.2.1 Programmis kasutatavad hinnangud

Programmis on võimalik valida nelja erineva strateegia vahel. Strateegia tüüpideks on ründav, kaitsev, jooksev ning eelnimetatud strateegiate kombinatsioon. Iga strateegia kasutab erinevat hinnangufunktsiooni, mis on kombineeritud mängulaua seisu erinevatest karakteristikutest. Hinnangufunktsioonide konstrueerimisel on autor lähtunud põhiliselt 1. peatükis vaatluse alla võetud triktraki strateegiatest, aga inspiratsiooni on saadud ka A. Sjöqvusti ja A. Stenlundi tööst "Constructing an Evaluation Function for Playing Backgammon" [11].

3.2.1.1 Mängulaua karakteristikute arvutamise funktsioonid

Karakteristikute funktsioonid on mängijate suhtes sümmeetrilised, st. korraga vaadeldakse nii mängija kui ka vastase kindlaks määratud karakteristikut. Lisaks on kõikide funktsioonide tulemused seatud rangelt piiridesse [-1,1], et vältida joonisel 18 kujutatud haavatavust.

Müüril asetsevate nuppude arv

Funktsioon h_n seab positiivse väärtuse müüril asetsevatele vastase nuppudele ning negatiivse väärtuse müüril asetsevatele mängija nuppudele. h_n arvutatakse järgnevalt:

$$h_n(x) = \frac{n_{min}(x)}{N} - \frac{n_{max}(x)}{N},$$

$h_n(x) \in [-1, 1]$, kus

$n_{min}(x)$ tähistab tipus *x* tehtava käigu järel müüril asetsevate vastase nuppude arvu;

$n_{max}(x)$ tähistab tipus *x* tehtava käigu järel müüril asetsevate mängija nuppude arvu ning

N tähistab maksimaalset mängija nuppude arvu müüril ($N = 15$).

Nuppude kaugus mahakandmislauast

Funktsioon h_d seab positiivse väärtuse vastase mahakandmislauast kaugel asetsevatele vastase nuppudele ning negatiivse väärtuse mängija mahakandmislauast kaugel asetsevatele mängija nuppudele. h_d arvutatakse järgnevalt:

$$h_d(x) = \frac{d_{min}(x)}{D} - \frac{d_{max}(x)}{D},$$

$h_d(x) \in [-1, 1]$, kus

$d_{max}(x)$ tähistab tipus *x* tehtava käigu järel mängija mahakandmislauast kõige kaugemal asetseva mängija nupu kaugust;

$d_{min}(x)$ tähistab tipus x tehtava käigu järel vastase mahakandmislauast kõige kaugemal asetseva vastase nupu kaugust ning

D tähistab maksimaalset kaugust mahakandmislauast ($D = 23$).

Suletud kiilud siselaual

Funktsioon h_{sk} seab positiivse väärtuse mängija siselaua kiilude ning vahetult siselauale järgneva kolme kiilu sulgemisele mängija nuppudega ning negatiivse väärtuse vastase siselaua piirkonnas vastase nuppudega suletud kiiludele. h_{sk} arvutatakse järgnevalt:

$$h_{sk}(x) = \frac{sk_{max}(x)}{SK} - \frac{sk_{min}(x)}{SK},$$

$h_{sk}(x) \in [-1, 1]$, kus

$sk_{max}(x)$ tähistab tipus *x* tehtava käigu järel mängija siselaua ning sellele vahetult järgneva kolme kiilu piirkonnas mängija nuppudega suletud kiilude arvu;
$sk_{min}(x)$ tähistab tipus *x* tehtava käigu järel vastase siselaua ning sellele vahetult järgneva kolme kiilu piirkonnas vastase nuppudega suletud kiilude arvu ning

SK tähistab maksimaalset suletud kiilude arvu ($SK = 7$, kuna minimaalne kiilu sulgemiseks kuluv nuppude arv on 2, ning $\left\lfloor \frac{15}{2} \right\rfloor = 7$).

Üksikute nuppude arv laual

Funktsioon h_s seab positiivse väärtuse üksikutele vastase nuppudele laual ning negatiivse väärtuse üksikutele mängija nuppudele laual. h_s väärtus arvutatakse järgnevalt:

$$h_s(x) = \frac{s_{\min}(x)}{S} - \frac{s_{max}(x)}{S},$$

$h_s(x) \in [-1, 1]$, kus

$s_{max}(x)$ tähistab tipus *x* sooritatava käigu järel mängija üksikute nuppude arvu laual;

$s_{min}(x)$ tähistab tipus *x* sooritatava käigu järel vastase üksikute nuppude arvu laual ning

S tähistab maksimaalselt (ühe mängija) üksikute nuppude arvu laual ($S = 15$).

Võimalikud probleemid üksikute nuppude leidmise hinnanguga

Kuna loendatakse ainult mängulaual asetsevaid üksikuid nuppe ning ei arvestata võimalusega, et osad mängija nuppudest võivad olla müürile saadetud, võib juhtuda, et küllaltki halvale seisule tagastatakse hea hinnang. Illustratsiooniks võib tuua olukorra, kus kõik mängija nupud on müürile saadetud: kuna funktsioon ühtegi üksikut nuppu laualt ei leia, tagastatakse seisule mittenegatiivne hinnang.

Maha kandmine

Funktsioon h_b seab positiivse väärtuse laualt mahakantud mängija nuppudele. h_b arvutatakse järgnevalt:

$$h_b(x) = \frac{b_{max}(x)}{B},$$

$h_b\,(x) \in [0,1]$, kus

$b_{max}(x)$ tähistab tipus *x* sooritatava käigu järel laualt maha kantud mängija nuppude arvu.

B tähistab maksimaalset (ühe mängija) maha kantud nuppude arvu ($B = 15$).

Erinevalt peaagu kõikidest teistest karakteristikute arvutamise funktsioonidest, ei võta h_b arvesse vastase olukorda, kuna enamasti ei ole selles seisus mängijatel enam üksteist nagunii võimalik mõjutada.

3.2.1.2 Programmis kasutatavad strateegiad ja nende hinnangufunktsioonid

Ründav

Ründav strateegia seab rõhu eelkõige vastase nuppude müürile saatmises (h_{nm}), aga võtab arvesse ka nuppude kaugust mahakandmislauast (h_d). Ründava strateegia hinnang tipus *x* arvutatakse järgnevalt:

$$h_{ründav}(x) = w_1 \cdot h_{nm}(x) + w_2 \cdot h_d(x),$$

$h_{ründav}(x) \in [-1, 1]$, kus

$w_1 = 0.8;\ w_2 = 0.2.$

Kaitsev

Kaitsev strateegia seab rõhu eelkõige oma nuppude kaitsmisele vastase eest ning selle hinnangufunktsioon kombineerib kiilude sulgemise siselaual (h_{sk}) üksikute nuppude mi nimeerimise (h_s) ning jooksuga (h_d). Kaitsva strateegia hinnang tipus *x* arvutatakse järgnevalt:

$$h_{kaitsev}(x) = w_1 \cdot h_{sk}(x) + w_2 \cdot h_s(x) + w_3 \cdot h_d(x),$$

$h_{kaitsev}(x) \in [-1, 1]$,

$w_1 = w_2 = 0.4;\ \ w_3 = 0.2.$

Kombineeritud

Kombineeritud strateegia ühendab kaitsva ning ründava strateegia: võimaluse korral üritatakse vastase nupud müürile saata (h_{nm}) ning siselaual kiilude sulgemisega (h_{sk}) neid seal lõksus hoida. Arvesse võetakse ka nuppude kaugust mahakandmislauast. Kombineeritud strateegia hinnang tipus *x* arvutatakse järgnevalt:

$$h_{komb}(x) = w_1 \cdot h_{sk}(x) + w_2 \cdot h_{nm}(x) + w_3 \cdot h_d(x),$$

$h_{komb}(x) \in [-1, 1]$,

$w_1 = w_2 = 0.4;\ \ w_3 = 0.2.$

Jooks

Jooksva strateegia ainus eesmärk on nuppudega võimalikult kiiresti oma siselauale jõuda ning strateegia hinnangu leidmisel võetakse arvesse üksnes nuppude kaugust mahakandmislauast (h_d). Jooksva strateegia hinnang arvutatakse seega järgnevalt:

$$h_{jooks}(x) = w_1 \cdot h_d(x)$$

$h_{komb}(x) \in [-1, 1]$,

$w_1 = 1.0.$

Enne hinnangufunktsioonide rakendamist testitakse esmalt, kas tipp on lõpptipp ning kui nii, siis rakendatakse tipule *x* kasulikkusfunktsiooni k, ($k(x) = 1$, kui tipus *x* on maksimeerija võit ning $k(x) = -1$, kui tipus *x* on minimeerija võit).

Lisaks kontrollitakse veel eelnevalt, kas mängija võib nuppe maha kanda ning kui nii, siis rakendatakse strateegiaga defineeritud hinnangufunktsiooni asemel tipule *x* funktsiooni h_b, mis seab lauaseisule seda parema hinnangu, mida rohkem on laualt maha kantud mängija nuppe. Peale selle tehakse ka kindlaks, kas mängijate nupud saavad veel üksteist mõjutada. Kui nupud üksteisele enam mõju avaldada ei saa, rakendatakse kindlaks määratud hinnangufunktsiooni asemel jooksva strateegia analoogi, mille ainuke erinevus

on see, et arvesse võetakse ainult mängija nuppude kaugust oma mahakandmislauast (kuna mängijate nupud üksteist enam mõjutada ei saa, pole vastase käikudele enam mõtet tähelepanu pöörata).

Tipu *x* väärtuse v_x leidmine toimub järgneva algoritmi järgi:

Kui *x* on terminaalne tipp, siis võta

- $v_x = 1,$ kui tipus *x* on maksimeerija võit;
- $v_x = -1,$ kui tipus *x* on minimeerija võit.

Kui mängija võib tipus *x* nuppe laualt maha kanda, võta

- $v_x = h_b\,(x).$

Kui mängijate nupud ei saa tipus *x* enam üksteist mõjutada, võta

- $v_x = h_r(x).$

Muul juhul võta

- $v_x = h_{strateegia}(x).$

4 Tulemuste analüüs

See peatükk keskendub expect-minimaxi algoritmi erinevate parameetrite analüüsimisele.

4.1 Strateegiate analüüs

Statistika kogumine

Iga erinev mängijate paar mängis omavahel 50 mängu puu sügavusel 3.

	Juhuslikkus	Jooks	Ründav	Kaitsev	Kombineeritud
Juhuslikkus		22%	30%	24%	32%
Jooks	78%		36%	38%	34%
Ründav	70 %	64%		38%	48%
Kaitsev	76%	62%	62%		34%
Kombineeritud	68%	66%	58%	66%	

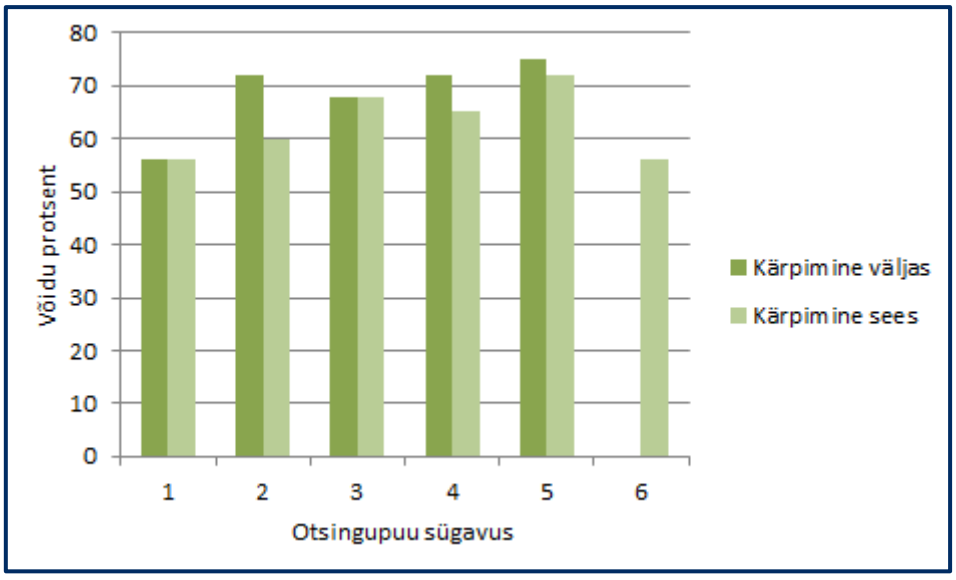
Tabel 1. Iga tabeli i-nda rea ja j-nda veeru ristumispaigas oleva lahtri väärtus näitab, mitu protsenti mängudest võitis i-ndas reas olev mängija j-ndas veerus oleva vastase vastu mängides ($i, j = \{1,2,3,4,5\}$).

Juhuslikke käike tegev algoritm kaotas kõigile teistele strateegiatele, mis kinnitab, et expect-minimaxi algoritm ning selle tippudele rakendatavad hinnangufunktsioonid töötavad. Jooksu strateegiat kasutav mängija kaotas kõikidele teistele mängijatele peale juhuslikkuse, mis kinnitab ka peatükis 1 esitatud väidet, et jooksva strateegiaga võidab see, kellel täringutega rohkem õnne on. Kõige paremaks strateegiaks osutus oodatult kombineeritud strateegia, mis võitis kõikide vastaste vastu mängides.

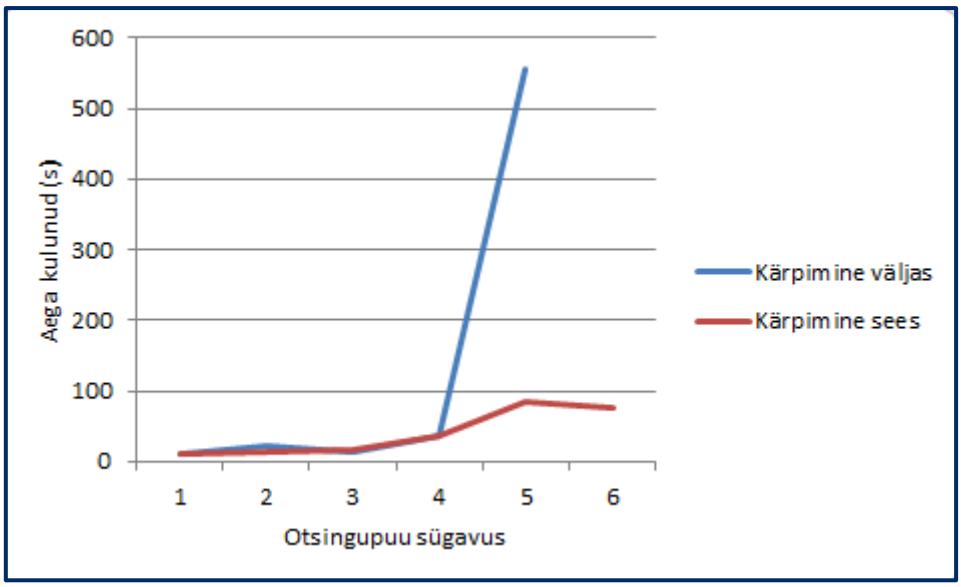
4.2 Puu sügavuse analüüs

Statistika kogumine

Igal puu sügavusel sooritati 25 mängu, kus esimene mängija kasutas jooksu strateegiat ning sooritas otsingut alati konstantsel sügavusel 2 ning teine vastane kasutas kaitsvat strateegiat, kuid sooritas otsingu iga mänguseeria erineval sügavusel. Samade parameetritega seeria viidi läbi ka mängupuu kärpimist kasutades. Tulemustest võis välja lugeda, et mängupuu sügavus võitude arvu kuigi palju ei mõjuta, samuti ei olnud võitude suhtes olulist erinevust sellel, kas mängija kasutas puu kärpimist või mitte. See-eest võis aga juba 5. tasemel märgata kärpimist mitte kasutava algoritmi tohutult suurenenud ajakulu ning 6. tasemel ei õnnestunudki kärpimist mitte kasutava algoritmi käigu tulemust mõistliku aja piires kätte saada, samas kui kärpimist kasutav algoritm mängis nii 5. kui ka 6. tasemel küllaltki edukalt ning käikude sooritamiseks ei kulunud vastava taseme kohta väga kaua aega. Joonistelt 23 ja 24 võib välja lugeda, et optimaalne puu otsingusügavus on umnbes 3, kuna sellel sügavusel sooritatakse käigud küllaltki kiiresti heale tulemusele.



Joonis 23. Otsingupuu sügavus ja keskmine võiduprotsent sellel sügavusel.



Joonis 24. Otsingupuu sügavus ja keskmine üheks mänguks kulunud aeg (sekundites)

5 Ettepanekuid edasiarendusteks

Triktraki programmi hetkeversioonis pole puutippe enne kärpimist ühelgi viisil järjestatud ning seega ei pruugi kärpimine otsingu kiirust iga kord parandada. Ühe võimaliku edasiarendusena võiks programmis implementeerida funktsiooni, mis järjestab puutipud alati nii, et lahendus leitaks $O(b^{d/2})$ aja vältel.

Programm võiks võimaldada ka statistika kogumist ning sealjuures võimalust mõõta igaks mänguks kulunud aega, mis aitaks omakorda paremini analüüsida puu kärpimise ning sügavuse suurendamise mõju.

Lisaks võiks programmis implementeerida ka topeldamistäringu ning sellest seoses arendada hinnangufunktsioone, mis oskaks hinnata, millal võiks teha ettepaneku panuseid tõsta ning millal vastase panuseid vastu võtta. Peale selle võiks programmis implementeerida ka teisi juhuslikkuse elementi sisaldava kahe isiku mängu algoritme. Maailmatasemel triktrakki mängivad tehisintellektid kasutavad mängupositsiooni hindamiseks näiteks tehisnärvivõrke.

Kokkuvõte

Triktrakk on ligi 5000 aastat vana juhuslikkuse elementi sisaldav kahe isiku mäng, kus mängu tulemuses mängib küll oma osa õnn, kuid üldjuhul võidab siiski mängija, kes mängupositsiooni paremini hinnata suudab. Triktraki-taoliste juhuslikkuse elementi sisaldavate kahe isiku mängude jaoks ei saa genereerida tavalist mänguseisude puud, kus vahelduvad vaid minimeerija ja maksimeerija tase, kuna mängija lubatud käigud selguvad alles täringu veeretamise tulemusel.

Antud töö eesmärgiks on selgitada, kuidas toimub mängupositsiooni hinnang juhuslikkuse elementi sisaldava mängu seisude puul ning analüüsida expect-minimaxi algoritmi, mis on üks lihtsamaid ning levinumaid algoritme, mida juhuslikkust sisaldavate kahe isiku mängudes kasutatakse.

Töös on ka antud ülevaade triktraki mängureeglitest ja levinumatest strateegiatest. Töö käigus on valminud triktraki mängu programm, mis võimaldab kasutajal arvuti algoritmi vastu mängida või vaadelda arvuti algoritme omavahel mängimas. Töös implementeeritud algoritmide hulka kuulub juhuslikke käike sooritav juhuslikkuse algoritm ning expect-minimaxi algoritm, mille valimisel saab lisaks valida erinevate parameetrite vahel nagu puu sügavus, kasutatav strateegia ning kas otsingupuud kärpida või mitte. Lisaks on töös antud ülevaade erinevate strateegiate jaoks konstrueeritud hinnangufunktsioonidest ning karakteristikutest, millele need rõhu asetavad.

Töös on ka analüüsitud ja võrreldud programmis kasutatavate expect-minimaxi algoritmi erinevate strateegiate efektiivsust.

Töö sobib kasutamiseks lisamaterjalina Tartu Ülikoolis loetavas kursuses Tehisintellekt I.

Kasutatud kirjandus

[1] C. Bray, *„Backgammon History“,* 2007 http://www.bkgm.com/articles/Bray/BackgammonHistory/ (viimane külastus: 11.05.2015)

[2] M. Strato, *„A Brief History of Backgammon“* http://www.gammonlife.com/history/ (viimane külastus: 11.05.2015)

[3] *„The Trial of Oregon Promoter Ted Barr“* http://www.bkgm.com/articles/BackgammonTimes/LuckVsSkill-TrialOfTedBarr/ (viimane külastus: 11.05.2015)

[4] B. Burns, „ Mängude raamat“, Tänapäev, 2008

[5] B. Burns, *„Mängude entsüklopeedia“,* ERSEN, 2008

[6] S. Brown, *„How to Win at Backgammon“* http://boardgames.about.com/od/backgammon/a/Basic-Backgammon-Strategy.htm (viimane külastus: 11.05.2015)

[7] H. Gillis, „*Backgammon for Life's Challenges*“, 2011 http://usbgf.org/backgammon-for-lifes-challenges/ (viimane külastus: 11.05.2015)

[8] S. Russell, P. Norvig, „*Artificial Intelligence: A Modern Approach“,* Pearson Education, 2nd ed., 2003.

[9] M. Koit, T. Roosmaa, „*Tehisintellekt“*, TÜ Kirjastus, 2011

[10] E. Melkó, B. Nagy, *„Optimal strategy in games with chance nodes“,* Acta Cybernetica nr. 18, 2007 http://www.inf.uszeged.hu/actacybernetica/edb/vol18n2/pdf/Melko_2007_ActaCybernetica.pdf (viimane külastus: 11.05.2015)

[11] A. Sjöqvust, A. Stenlund, *"Constructing an Evaluation Function for Playing Backgammon"*, 2011 http://www.csc.kth.se/utbildning/kth/kurser/DD143X/dkand11/Group3Johan/final/Anders.Sjoqvist.Andre.Stenlund.report.pdf (viimane külastus: 14.05.2015)

[12] Wikipedia – Expectiminimax tree http://en.wikipedia.org/wiki/Expectiminimax_tree (viimane külastus: 14.05.2015)

Lisad

Lisa 1 – Triktraki mänguprogramm

Töö käigus loodud programm, mille eesmärk on triktraki alusel illustreerida erinevaid hinnangufunktsioone kasutava expect-minimaxi algoritmi tööd. Käivitamiseks on vajalik Java olemasolu (versioon 1.7 või hilisem).

Lisa 2 – Programmi kasutusjuhend

Triktraki mänguprogrammi funktsioone ning nende kasutamist kirjeldav juhend.

Lisa 3 – Programmi lähtekood

Triktraki mänguprogrammi lähtekood.

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Marit Asula** (sünnikuupäev: 21.12.1991)
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Strateegia ja mängupositsiooni hindamine juhuslikkuse elementi sisaldavas kahe isiku mängus**,
(*lõputöö pealkiri*)

   mille juhendaja on Mare Koit,
   (*juhendaja nimi*)

   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2015**