**Tekoälyille tehtyjä testejä:**

**Johdanto**

Tähän päiväkirjaan on kirjoitettu testituloksia tekoälyille tekemistäni testeistä. Niiden pääasiallisena tarkoituksena on ollut olla apuna minulle kehittäessäni tekoälyistä parempia - vertailemalla uusia tuloksia vanhoihin olen aina nähnyt oliko uusi ratkaisu parempi. Niitä ei ole siistitty ollenkaan, ja paikka paikoin se voi olla melko raskastakin luettavaa.

Päiväkirjasta kuitenkin löytyy selitykset kaikille tehdyille ratkaisuille tekoälyjen hyödyntämistä metodeista, ja tulokset siitä kuinka paljon ne ovat vaikuttaneet. Lopussa on yhteenveto, jossa kerrotaan lyhyesti miten tekoälyn kehitys on kurssin aikana tehty.

Testejä on tehty myös jonkin verran tämän päiväkirjan ulkopuolella, ja joissain kohdissa viitataankin myös testeihin joiden tuloksia ei näytetä. Suurin osa niistä on kuitenkin ollut testejä, joissa uusi ominaisuus on todettu huonoksi tai ne ovat liittyneet pelikorttien tasapainottamiseen. Käsin itse pelaamalla tehdyistä testeistä löytyy tietoa testausdokumentista.

**22.3.2015:**

Testasin ensimmäistä, täysin satunnaisuuteen perustuvaa tekoälyä itseään vastaan. Tavoitteena oli tehdä havaintoja tasapainosta pelaajien 1 ja 2 välillä. Massatestauksessa oli jonkin verran ongelmia, ja StackOverFlowErroria tuli aina jos yritin tehdä liian monta testiä kerralla. Pitänee optimoida pelin restarttausta jotenkin, jotta voidaan suorittaa enemmän testejä kerralla.

Testin tulokset:

Pelaaja 1: 909 voittoa

Pelaaja 2: 1000 voittoa

Pelaajien asetelma tuntuisi olevan siis yllättävän tasainen tällä hetkellä.

**23.3.2015:**

Koodasin uuden tavan testata tekoälyjä vastakkain ja pääsin eroon StackOverFlowErroreista. Nyt tekoälyn testaus voidaan suorittaa launcherista suoraan kutsumalla AITester-luokkaa ja antaa parametrina tehtävien testien määrä sekä käytettävät tekoälyt.

Käyttämälläni tietokoneella testipelejä voidaan pelata nyt noin 1 000 000 / 30 sec. Tehtyäni useita testejä tällä määrällä sekä suuremmilla, totesin että miljoonan testin lopputulos on vakiintunut jo niin lähelle keskiarvoa, että sitä voidaan käyttää tulevaisuuden testauksessa.

Testin tulokset 10 000 000:lla

Pelaaja 1: 4733002 voittoa

Pelaaja 2: 5266998 voittoa

Pelaajan 1 voitto-häviö-ratio: ~0.8986

Pelaajan 2 voitto-häviö-ratio: ~1.1128

Peli siis todellakin tuntuu olevan melko tasainen aloitusvuorosta riippumatta. Uusien tekoälyjen kehittämistä voidaan siis jatkaa tällä asetelmalla, ja kun ollaan päästy tarpeeksi tyydyttävään tekoälyyn, voidaan sen kanssa tehtyjen testien avulla lähteä tasapainottamaan peliä tarkemmin.

**24.3.2015:**

Tein lisää testejä SimpleAI:lla ja kokeilin saisiko peliä vielä tasapainotettua pienillä muutoksilla. Totesin mm. että pelaajan 2 5 enemmän resursseja joka vuoro on erittäin hyvä tasoittava tekijä, sillä poistamalla sen saatiin tulokset:

Pelaaja 1: 722055

Pelaaja 2: 277945

Pelaajan 1 voitto-häviö-ratio: 2.597834

Pelaajan 2 voitto-häviö-ratio: 0.38493606

Vähentämällä molempien pelaajien aloituskortteja yhdellä:

Pelaaja 1: 447335 voittoa

Pelaaja 2: 552665 voittoa

Pelaajan 1 voitto-häviö-ratio: 0.8094144

Pelaajan 2 voitto-häviö-ratio: 1.2354611

Ottamalla pois pelaajan 2 ylimääräisen aloituskortin saatiin seuraavat tulokset:

Pelaaja 1: 604888 voittoa

Pelaaja 2: 395112 voittoa

Pelaajan 1 voitto-häviö-ratio: 1.5309279

Pelaajan 2 voitto-häviö-ratio: 0.6531986

Muuttamalla maxInfluenceksi 15, 25, 30 tai 35 ei ollut mitään vaikutusta pelaajien tasapainoon. 20:llä pelien pituus tuntuisi sopivalta, joten se pidettäköön toistaiseksi.

Lisäämällä molempien pelaajien aloituskortteja yhdellä (pelaaja 1: 8, pelaaja 2: 9):

Pelaaja 1: 500190 voittoa

Pelaaja 2: 499810 voittoa

Pelaajan 1 voitto-häviö-ratio: 1.0007603

Pelaajan 2 voitto-häviö-ratio: 0.9992403

Kauheasti tämän tasaisemmaksi ei peli enää voi mennä, ja suuri aloituskorttien määrä vähentää myös tuurin osuutta pelin lopputuloksesta. Näillä arvoilla jatketaan siis pelin kehittämistä eteenpäin:

Max influence: 20

Max resources: 100

Max turn resources: 80

Pelaajan 1 aloituskortit: 8

Pelaajan 2 aloituskortit: 9

Resurssien kasvu per vuoro: 10

Pelaajan 1 aloitusresurssit: 10

Pelaajan 2 aloitusresurssit: 15

**25.3.2015:**

**Voittovuoron tarkistus: checkLethal**

Tein ominaisuuden, jolla tekoäly tarkistaa aina vuoron alussa voiko se voittaa sillä vuorolla ja se viimeistelee pelin jos mahdollista. Testasin antaa ominaisuuden pelaajalle 1 seuraavilla tuloksilla:

Pelaaja 1: 5081432 voittoa

Pelaaja 2: 4918568 voittoa

Pelaajan 1 voitto-häviö-ratio: 1.033112

Pelaajan 2 voitto-häviö-ratio: 0.9679492

Ja pelaajalle 2 seuraavilla tuloksilla:

Pelaaja 1: 4956593 voittoa

Pelaaja 2: 5043407 voittoa

Pelaajan 1 voitto-häviö-ratio: 0.98278666

Pelaajan 2 voitto-häviö-ratio: 1.0175148

Molemmille pelaajille:

Pelaaja 1: 5028758 voittoa

Pelaaja 2: 4971242 voittoa

Pelaajan 1 voitto-häviö-ratio: 1.0115697

Pelaajan 2 voitto-häviö-ratio: 0.9885626

Näiden testien perusteella voidaan todeta, että checkLethal selvästi parantaa AI:n voittomahdollisuuksia, ja se antaa hieman enemmän etua aloittavalle pelaajalle. Testien perusteella voidaan todeta ominaisuus kannattavaksi, ja se annetaan jatkossa kaikille uusille AI:lle (SimpleAI pysyy ennallaan, koska sen tarkoitus on perustua täysin satunnaisuuteen). Loogisesti ajateltuna "jos pelaaja voi voittaa tällä vuorolla, niin voita" on myöskin perus käytäntö pelissä kuin pelissä, ja tekoäly vaikuttaisi hölmöltä ilman sitä. Se myös lyhentää pelejä.

Tässä vaiheessa checkLethal metodi on melko optimaalinen, mutta siihen voitaisiin vielä lisätä ominaisuus, jossa omia pienen damagen kortteja tuhotaan pöydästä ja tehdään tilaa paremmille mounted-korteille, mutta tapaukset joissa tästä on hyötyä ovat erittäin harvinaisia. Tällä tuskin olisi suurta vaikutusta, se hidastaisi testejä ja se on myös toimintaa jota moni pelaajakaan ei itse tajuaisi tehdä. Sen lisäystä voidaan harkita lopulliselle parhaalle tekoälylle.

Toinen mahdollinen lisäys olisi optimoida mounted-minionien tekemä damage etsimällä paras mahdollinen yhdistelmä kädestä. Se tullaan varmasti lisäämään paremmille tekoälyille.

**26.3.2015:**

Testasin vielä miten tekoäly käyttäytyy jos siltä poistaa toisen playTablen alusta, jolloin välillä voi käydä niin että pöytä on täynnä -> ei pelata kortteja kädestä -> pöytä tyhjenee kun minionit hyökkää, mutta enää ei pelata kortteja. Poistettiin pelaajalta 1:

Pelaajan 1 voitto-häviö-ratio: 0.9672995

Pelaajan 2 voitto-häviö-ratio: 1.033806

Silläkin on siis selvästi vaikutusta. Toisaalta jos playTablen poistaa lopusta, eli mounted minionit ei koskaan hyökkää sillä vuorolla minioneita kun ne pelataan:

Pelaajan 1 voitto-häviö-ratio: 0.6446584

Pelaajan 2 voitto-häviö-ratio: 1.5512091

Sillä on jo suurempi vaikutus.

**31.3.2015:**

**Minioneiden järjestys pöydällä**

(Testit 100 000 000 pelillä).

Alkuperäinen MediumAI vs. MediumAI pelaajan 1 voitto-häviö-ratio: 1.0115697

Koodasin metodin, jolla minionit laitetaan pöytään tietyssä järjestyksessä satunnaisuuden sijaan: järjestys lisää todennäköisyyttä että Guardian suojaa muita minioneita, mutta ei tee sitä vielä optimaalisesti. Se kuitenkin antoi seuraavat tulokset aiempaa MediumAI toteutusta vastaan:

Pelaajan 1 voitto-häviö-ratio: 1.1545196

Eli ero on jo huomattava. Seuraavaksi koodattiin mukaan vielä ominaisuus, joka tarkistaa guardiania laitettaessa onko ennalta määrätyn paikan vieressä oikeasti toinen minion:

Pelaajan 1 voitto-häviö-ratio: 1.1594834

Sitten lisättiin vielä muille minioneille ominaisuus, että ne pelataan guardianin viereen jos mahdollista:

Pelaajan 1 voitto-häviö-ratio: 1.1598407

**1.4.2015:**

Testailin lisää MediumAI:n metodeja vaihtelemalla tehtävien asioiden järjestystä jne. Kaikki toimenpiteet kuitenkin huononsivat voittoprosenttia, joten jätin muutokset tekemättä.

**2.4.2015:**

Testasin vielä tehdyn MediumAI-toteutuksen vanhaa SimpleAI:ta vastaan seuraavilla tuloksilla:

MediumAI:n voitto-häviö-ratio: 1.1899377

**17.4.2015:**

(Tässä vaiheessa tein havainnon, että 1 000 000:lla pelillä tuloksissa alkaa olla jo hieman heittoa toistettuna, joten jatkossa testit tehdään aina 10 000 000:lla pelillä, jolla tuntuisi tulevan jo erittäin tarkka tulos.)

Kehitin seuraavaa AI:ta ja testasin playTurnia, jossa minionit pyrkivät aluksi hyökkäämään kohteita joiden health on sama kuin hyökkääjän attack ja sitten vasta jäljellejääneet hyökkäävät randomilla. Korttien pelaaminen pöytään tapahtui järjestyksessä ABEmpty:

TestAI:n voitto-häviö-ratio MediumAI:ta vastaan: 1.7367374

Kehityksessä ollaan siis selvästi menty oikeaan suuntaan.

Lisäsin ominaisuuden, jolla hyökkääjä yrittää hyökätä kohdetta, joka ei tapa hyökkääjää jos em. ei onnistu:

TestAI:n voitto-häviö-ratio MediumAI:ta vastaan: 1.6594691

Tämä ei siis selvästi ole hyvä juttu, joten se jätetään toteuttamatta. Seuraavaksi kokeilin ominaisuutta, jossa em. jälkeen yritetään hyökätä vielä minioneita, joiden health on vähemmän kuin hyökkääjän attack:

TestAI:n voitto-häviö-ratio MediumAI:ta vastaan: 1.8439664

Poistin random hyökkäyksen tableAttackin ja playABEmptyn välistä:

TestAI:n voitto-häviö-ratio MediumAI:ta vastaan: 1.8464086

Poistin sen myös vuoron lopusta:

TestAI:n voitto-häviö-ratio MediumAI:ta vastaan: 1.844671

Kun lisättiin ominaisuus, että AI pelaa ensisijaisesti mounted minionit, jos ne voivat sillä vuorolla tappaa toisen minionin. Voittprosentti laski 1.820409:een, mutta ominaisuus kuitenkin vaikuttaa hyvältä. Kokeillaan vielä pelata AI:lla jolla se on AI:ta vastaan jolla sitä ei ole. Aloitetaan kokeilemalla miten pelaajien 1 ja 2 voittoprosentit käyttäytyvät tällä toteutuksella (ilman mounted ominaisuutta):

Pelaajan 1 voitto-häviö-ratio: 1.0209941

Pelaajan 2 voitto-häviö-ratio: 0.9794376

Vaihdetaan pelaajalle 2 playMountedToKill-ominaisuus:

Pelaajan 1 voitto-häviö-ratio: 0.8753391

Pelaajan 2 voitto-häviö-ratio: 1.1424145

Tässä testissä nähdäänkin, että ominaisuus paransi huomattavasti tekoälyä, kun ne muuten ovat identtiset. Pidetään siis ominaisuus.

Päivän päätteeksi kokeilin vielä kehityksen tässä vaiheessa olevaa AdvancedAI:ta vanhaa SimpleAI:ta vastaan:

Pelaajan 1 voitto-häviö-ratio: 2.266045 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.44129747 (SimpleAI)

**2.5.2015:**

Testasin, että jos playMountedsToKill kutsuttaisiin ennen neljän vaihtoehdon valitsemista, mutta se vähensi voittoprosenttia yhdellä.

Kokeilin, jos em. käden pelaamisen sijasta kutsutaankin tableAttackToKill sijasta playHandRandomlyä. Voittoprosentti huononi samaa luokkaa kuin ennen tableAttackToKill ja

playMountedsToKill lisäämistä. Suunta mihin on menty tuntuisi siis oikealta, jatkokehitetään siis tältä pohjalta.

(Muutokset tehdään aina pelaaja 2:lle)

Muutin playAEmptyn niin, että ellei mounted unitit saa tyhjennettyä vastustajan pöytää, ei kutsuta enää playWorkersiä alussa jos kädessä on tarpeeksi kortteja. Muuten pelaamisjärjestys pysyy samana:

Pelaajan 1 voitto-häviö-ratio: 0.93310153

Pelaajan 2 voitto-häviö-ratio: 1.0716947

Tämä selvästi paransi voittoprosenttia. Oma osuutensa vaikutukseen on varmasti se, että worker minionit eivät välttämättä ole yhtä hyviä kuin muut, mutta tällä niiden huonoa pelaamista voidaan ainakin välttää.

Poistetaan playMounteds kokonaan playAEmptystä (kokeillaan kannattaako ne säästää tappoihin jos pöydässä on vastustajan minioneita):

Pelaajan 2 voitto-häviö-ratio: 1.0626829

Ei näköjään. Muutetaan sama kuin aiemmin nyt koskemaan myös playABNotEmptyä:

Pelaajan 2 voitto-häviö-ratio: 1.2276016

Workereitä ei siis selvästi kannata pelata koskaan, jos pöydässä on vastustajan minioneita. Kokeillaan playMounteds poistoa vielä tälle:

Pelaajan 2 voitto-häviö-ratio: 1.216769

Selvästi mounted kortteja kannattaa pelata muutenkin kuin poistaakseen vastustajan kortin. Mietittyäni AdvancedAI:n tämänhetkistä toteutusta, tajusin että playABEmpty ja playBEmpty ovat samat ja playAEmpty ja playABNotEmpty ovat samat, enkä keksi mitään millä niiden käytöstä voitaisiin erotella toisistaan. Refaktoroidaan siis toimintaa yhdistelemällä metodeja.

Kokeillaan vielä tätä AIta vanhaa SimpleAI:ta vastaan:

Pelaajan 1 voitto-häviö-ratio: 2.5315747 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.39501104 (SimpleAI)

Ja MediumAI:ta vastaan:

Pelaajan 1 voitto-häviö-ratio: 2.1066108 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.47469613 (MediumAI)

Itseään vastaan (aloitusvuorojen tasapaino):

Pelaajan 1 voitto-häviö-ratio: 0.96263725

Pelaajan 2 voitto-häviö-ratio: 1.0388129

Testasin peliä AI:ta vastaan käsin ja löysin "bugeja": AI yrittää tappaa minioneita Guardianin ohi jolloin se hyökkää vahingossa Guardiania epäoptimaalisesti, ja playMountedsToKill ei hyödynnä minioneille määriteltyä järjestystä mistä olisi hyötyä jos hyökkääjä selviää itse hyökkäyksestä. HUOM. näiden vaikutus on kuitenkin erittäin pieni tässä toteutuksessa, koska Minion hyökkäisi kuitenkin lopuksi randomilla ja Guardian vie parhaimmillaan 3 slottia jotka johtavat hyökkäyksen siihen kuitenkin. Ja Mounted minionit yleensä kuolevat hyökätessään. Korjataan kuitenkin jos ehditään.

**3.5.2015:**

Siirretään playGuardians playBNotEmptyssä ensimmäiseksi (TestAI pelaaja 2):

Pelaajan 2 voitto-häviö-ratio: 1.1148285

Voittoprosentti parani kuten odotinkin. Varsinkin näiden tekoälyjen pelatessa keskenään kestävien Minioneiden pelaaminen pöytään kun vastustajalla on siellä kortteja näyttäisi toimivan, koska suurin osa korteista on heikkoja ja tappavat itsensä hyökätessään Guardianeita. Ne myös toisaalta suojaavat muita omia kortteja.

Vaihdoin BNotEmptyssä playWarriors ja playRangeds paikkoja keskenään, huononi.

guardian->deadly->warrior->ranged->mounted->worker: 1.1137332

guardian->warrior->deadly->ranged->mounted->worker: 1.128926

guardian->warrior->deadly->mounted->ranged->worker: 1.1292783

Ranged minionit menevät selvästi hukkaan, jos ne pelataan tykinruuaksi. Lienee parasta kirjoittaa koodinpätkä, jossa ne pelataan vain guardianin suojaan kun vastustajalla on minioneita pöydässä.

guardian->warrior->deadly->mounted->worker->ranged: 1.0278468

Workerit näyttävät menevän vielä enemmän hukkaan. Jätetään playBNotEmpty tätä edeltäneeseen muotoon.

**5.5.2015:**

Tein mielenkiintoisen havainnon: edellä muutettu ratkaisu suosii aloittavaa pelaajaa huomattavasti enemmän. Pelatessa samoilla AI:lla keskenään, saatiin seuraavat tulokset:

Pelaajan 1 voitto-häviö-ratio: 1.1959344

Pelaajan 2 voitto-häviö-ratio: 0.8361662

Testatasin vielä millaiset tulokset toteutus saa, kun vanha ja uusi järjestys pelaavat eri puolilla:

Pelaajan 1 voitto-häviö-ratio: 1.3056073 (uusi)

Pelaajan 2 voitto-häviö-ratio: 0.7659271 (vanha)

Pelaajan 1 voitto-häviö-ratio: 0.8855818 (vanha)

Pelaajan 2 voitto-häviö-ratio: 1.1292012 (uuusi)

Uusi toteutus on siis selvästi parempi puolesta riippumatta, mutta se hyödyttää enemmän aloittavaa pelaajaa.

Muutin metodia tableAttackToKill niin, että ensin käydään kaikkien hyökkääjien optimaaliset kohteet läpi ja vasta sitten hyökätään overkill-kohteita. Vaikutus oli seuraavanlainen (uusi toteutus pelaajalla 2):

Pelaajan 1 voitto-häviö-ratio: 1.1830033

Pelaajan 2 voitto-häviö-ratio: 0.84530616

Eli muutos paransi marginaalisesti voittoprosenttia pitkällä aikavälillä. Se vaikuttaa muutenkin loogisemmalta pelitavalta AI:lle. Nyt AdvancedAI vs. AdvancedAI antaa seuraavat tulokset:

Pelaajan 1 voitto-häviö-ratio: 1.2093471

Pelaajan 2 voitto-häviö-ratio: 0.82689244

**6.5.2015:**

Testailin muuttaa vielä playBEmpty-metodin järjestyksiä:

alkuperäinen: worker(if cards > 3) -> warrior->ranged->deadly->guardian->mounted->worker

Pelaajan 2 voitto-häviö-ratio: 0.82689244 (alkuperäinen vs. alkuperäinen), muutetaan pelaajaa 2:

worker(if cards > 3)->ranged->warrior->deadly->guardian->mounted->worker: 0.83637595

worker(if cards > 3)->ranged->deadly->warrior->guardian->mounted->worker: 0.8396404

worker(if cards > 3)->ranged->deadly->guardian->warrior->mounted->worker: 0.83817244

worker(if cards > 3)->ranged->deadly->warrior->mounted->guardian->worker: 0.8395883

worker(if cards > 3)->ranged->deadly->warrior->guardian->worker->mounted: 0.8396265

worker(if cards > 3)->deadly->ranged->warrior->guardian->mounted->worker: 0.83395654

worker(if cards > 2)->ranged->deadly->warrior->guardian->mounted->worker: 0.83906454

worker(if cards > 4)->ranged->deadly->warrior->guardian->mounted->worker: 0.8405492

worker(if cards > 5)->ranged->deadly->warrior->guardian->mounted->worker: 0.8414378

worker(if cards > 6)->ranged->deadly->warrior->guardian->mounted->worker: 0.8441763

worker(if cards > 7)->ranged->deadly->warrior->guardian->mounted->worker: 0.8496497

ranged->deadly->warrior->guardian->mounted->worker: 0.8650509

ranged->worker(if cards > 4)->deadly->warrior->guardian->mounted->worker: 0.85591114

ranged->deadly->worker(if cards > 4)->warrior->guardian->mounted->worker: 0.86021626

ranged->deadly->warrior->worker(if cards > 4)->guardian->mounted->worker: 0.85875577

ranged->deadly->worker(if cards > 5)->warrior->guardian->mounted->worker: 0.860418

ranged->deadly->worker(if cards > 6)->warrior->guardian->mounted->worker: 0.86285406

ranged->deadly->worker(if cards > 7)->warrior->guardian->mounted->worker: 0.8654938

Enempää tätä ratkaisua on turha hioa. Muutetaan nyt workers-osuutta niin, että se ei ole riippuvainen kädessä olevien korttien määrästä, vaan niiden hinnasta.

ranged->deadly->worker(if cost>150)->warrior->guardian->mounted->worker: 0.8610409

ranged->deadly->worker(if cost>200)->warrior->guardian->mounted->worker: 0.861379

ranged->deadly->worker(if cost>210)->warrior->guardian->mounted->worker: 0.8617876

ranged->deadly->worker(if cost>220)->warrior->guardian->mounted->worker: 0.86134887

ranged->deadly->worker(if cost>190)->warrior->guardian->mounted->worker: 0.86217314

ranged->deadly->worker(if cost>180)->warrior->guardian->mounted->worker: 0.8618753

ranged->deadly->worker(if cost>185)->warrior->guardian->mounted->worker: 0.86334836

worker(if cost>185)->ranged->deadly->warrior->guardian->mounted->worker: 0.8447713

ranged->worker(if cost>185)->deadly->warrior->guardian->mounted->worker: 0.85756904

ranged->worker(if cost>195)->deadly->warrior->guardian->mounted->worker: 0.8577954

Costin mukaan workerien pelaaminen ei näköjään olekaan kannattavampaa kuin kädessä olevien korttien mukaan. Johtuneeko siitä, että korttien määrä kertoo pelin olevan alkuvaiheessa, kun niitä kannattaa pelata, kun taas hinta ei välttämättä yhtä suurella varmuudella. Kokeillaan vielä niiden

yhdistämistä:

ranged->deadly->worker(if cards>7||cost>185)->warrior->guardian->mounted->worker: 0.86368626

ranged->deadly->worker(if cards>7&&cost>185)->warrior->guardian->mounted->worker: 0.8653849

Eipä tuolle cost-metodille näyttäisi olevan oikeastaan hyötyä. Muutetaan AdvancedAI nyt parhaat tulokset saaneeseen muotoon:

ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker

Uusi pelaajan 2 voitto-ratio AdvancedAI vs. AdvancedAI: 0.8544852

Kokeilen millainen vaikutus on eri minionien pelaamiseen, jos ne pelataan vain kun pöydässä on guardianeita playBNotEmpty-llä:

ranged(if guarded)->guardian->warrior->deadly->mounted->ranged->worker: 0.86093354

ranged(if guarded)->guardian->ranged(if guarded)->warrior->deadly->mounted->ranged->worker: 0.879103

guardian->ranged(if guarded)->warrior->deadly->mounted->ranged->worker: 0.881441

warrior(if guarded)->guardian->ranged(if guarded)->warrior->deadly->mounted->ranged->worker: 0.87309515

guardian->warrior(if guarded)->ranged(if guarded)->warrior->deadly->mounted->ranged->worker: 0.86777705

guardian->deadly(if guarded)->ranged(if guarded)->warrior->deadly->mounted->ranged->worker: 0.86049

guardian->ranged(if guarded)->deadly(if guarded)->warrior->deadly->mounted->ranged->worker: 0.8831523

Muutetaan AdvancedAI tähän. Kokeillaan metodin hyödyntämistä playBEmptyssä:

Uusi pelaajan 2 voitto-ratio AdvancedAI vs. AdvancedAI: 0.8514579

worker(if guarded)->ranged->deadly->worker(if cards > 7)->warrior->guardian->mounted->worker: 0.8508925

ranged(if guarded)->worker(if guarded)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker:

deadly(if guarded)->worker(if guarded)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker:

worker(if guarded&&cards>3)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker:

worker(if guarded&&cost>185)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.85238975

worker(if guarded&&cost>190)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.8526602

worker(if guarded&&cost>195)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.8514853

worker(if guarded&&cost>200)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.85177606

worker(if guarded&&cards>7)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.8518539

worker(if guarded&&cards>6)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.8508284

worker(if guarded&&cost>190)->deadly(if guarded)->ranged->deadly->worker(if cards>7)->warrior->guardian->mounted->worker: 0.8519098

Nyt on mennyt testailu kyllä niin pilkun viilaukseksi, että en näillä enää testaile. Valitaan parhaan tuloksen saanut ja aletaan kehittää uusia metodeita. Kokeillaan vielä itseään, SimpleAI:ta ja MediumAI:ta vastaan (testit 100 000 000:lla):

**AdvancedAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 1.173931 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.8518388 (AdvancedAI)

**AdvancedAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 3.1260295 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.3198946 (SimpleAI)

**SimpleAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.37094435 (SimpleAI)

Pelaajan 2 voitto-häviö-ratio: 2.6958222 (AdvancedAI)

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 2.3935428 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.41779074 (MediumAI)

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.46836856 (MediumAI)

Pelaajan 2 voitto-häviö-ratio: 2.1350708 (AdvancedAI)

Voittoprosentit ovat nousseet huomattavasti viime kokeilusta (pelaajalla 1: 2.5316 SimpleAI vastaan, 2.1066 MediumAI vastaan) optimoimalla järjestystä, missä minioneita pelataan ja parantamalla hyökkäysmetodia hieman. Tämä AI antaa jo pelaajaakin vastaan melko vaikean vastuksen, jos ei tahallaan hyödynnä sijoittamisessa järjestystä, missä AI hyökkää. Sen voisikin korjata vielä satunnaiseksi, niin ennalta määrätyt järjestykset eivät enää ilmene muussa kuin kädessä, mitä pelaaja ei itse näe. Eli randomilla slotit taulukkoon, joka sitte käydään läpi.

Tuli myös todettua, että AI:t eivät osaa hyödyntää ranged-minioneita kovin hyvin tai hankkiutua ensisijaisesti eroon vastustajan rangedeista, mikä johtaa useissa peleissä siihen, että tarpeeksi kun niitä saa pöytään niin toisella ei ole enää mahdollisuuksia. Tämä on yksi asioista, joita AITesterillä ei ole tullut huomattua.

Tämänkaltaiset tekijät johtavat siihen, että kun lähdetään kehittämään optimaalisen skenaarion valitsevaa AI:ta, niin minioneille pitänee todennäköisesti keksiä tietyt painoarvot sen sijaan, että katsottaisiin esim. pöytään jäävää damagea tai omien ja vastustajan minionien määrää.

**13.5.2015:**

Vanha pelaajan 2 voitto-häviö-ratio: 0.8518388 (AdvancedAI vs. AdvancedAI)

Kokeilin vaihtaa playMountedsToKill:in samanlaiseksi kuin tableAttackToKill, eli ensin optimaaliset kohteet ja sitten vasta overkillit. Se huononsi voittoprosenttia marginaalisesti, minkä jälkeen kokeilin muuttaa metodia niin, että overkillejä ei pelata ollenkaan:

Pelaajan 2 voitto-häviö-ratio: 0.98142886

Muutos paransi voittoprosenttia huomattavasti. Tein testit vielä kaikkia AI:ta vastaan ja kaikilla aloituspaikoilla, ja muutos voittoprosentissa oli huomattava kaikilla:

**AdvancedAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 1.12992 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.8850184 (AdvancedAI)

**AdvancedAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 3.5070057 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.28514352 (SimpleAI)

**SimpleAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.32453963 (SimpleAI)

Pelaajan 2 voitto-häviö-ratio: 3.0812879 (AdvancedAI)

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 2.7674987 (AdvancedAI)

Pelaajan 2 voitto-häviö-ratio: 0.36133713 (MediumAI)

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.39863446 (MediumAI)

Pelaajan 2 voitto-häviö-ratio: 2.508564 (AdvancedAI)

**14.5.2015:**

Koska playMountedsToKillOverkill-metodi poistettiin kokonaan AdvancedAI:n toiminnasta, kokeilin vielä sijoittaa sen playBNotEmptyyn juuri ennen playMountedsia. Voittoprosentti parani marginaalisesti joitain vastaan ja huononi marginaalisesti joitain vastaan. Jätetään pois. Kokeilin vielä poistaa playMountedin kokonaan playBNotEmptystä. Se huononsi vain voittoprosenttia.

**15.5.2015:**

Kokeilin poistaa AdvancedAI:n playTurnista playTableRandomlyn lopusta. Se huononsi taas voittoprosenttia.

Muutin hyökkäysmetodeita niin, että hyökkäyskohteita ei enää yritetä järjestyksessä 0-7, vaan satunnaisesti. Nyt pelaaja ei voi enää huijata tekoälyä sijoittamalla minioneita pöytään sellasiiin slotteihin, joita haluaa AI:n hyökkäävän.

**AdvancedAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 1.11975

Pelaajan 2 voitto-häviö-ratio: 0.89305645

**AdvancedAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 3.5086706

Pelaajan 2 voitto-häviö-ratio: 0.28500822

**SimpleAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.32371667

Pelaajan 2 voitto-häviö-ratio: 3.089121

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 2.7889771

Pelaajan 2 voitto-häviö-ratio: 0.3585544

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.38766596

Pelaajan 2 voitto-häviö-ratio: 2.5795405

Testeissä voittoprosentit muuttuivat paremmiksi ja muutos tasoitti hieman myös aloituspaikkojen välistä eroa. Suurin ero on kuitenkin pelaajaa vastaan ja muutos on tärkeä.

Seuraavaksi kokeilin muuttaa hyökkäysmetodia niin, ettei guardianin takana olevia optimaalisia kohteita yritetä ollenkaan hyökätä hyökäten vahingossa guardiania. Tein sen muuttamalla logicHandlerin minionAttackia niin, että hyökkäystä ei yksinkertaisesti tehdä, jos minion on suojattu. Näin se poistaa myös pelaajalta mahdollisuuden vahingossa hyökätä väärää kohdetta. Jos kaksi guardiania on vierekkäin, ne eivät suojaa toisiaan. Lisäsin guarded tarkistuksen myös playMountedsToKilliin ja muutin playTableRandomlyä niin, että se jatkaa kohteen etsimistä jos se on randomannut suojatun kohteen.

**AdvancedAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 1.144208

Pelaajan 2 voitto-häviö-ratio: 0.873967

**AdvancedAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 3.6691473

Pelaajan 2 voitto-häviö-ratio: 0.2725429

**SimpleAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.30797958

Pelaajan 2 voitto-häviö-ratio: 3.2469685

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 2.9951663

Pelaajan 2 voitto-häviö-ratio: 0.33387128

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.3622512

Pelaajan 2 voitto-häviö-ratio: 2.7605155

Tämä koko rupeama korjasi sekä AI:ssa esiintyneen ongelman, lyhensi ja selkeytti koodia UserInterfacessa ja LogicHandlerissa ja paransi myös pelaajan käytettävyyttä. AdvancedAI:n voittoprosentti kasvoi huomattavasti tämän seurauksena. AI käyttäytyy nyt myös pelaajan silmään paljon järkevämmin. 5/5.

Seuraavaksi kokeilin vielä laittaa tableAttackToKillit ja playMountedsToKillit tuplana kaikkiin kohtiin: tämän seurauksena jos joku kortti jättää hyökkäämättä koska guardian suojaa optimaalista kohdetta mutta seuraavaksi hyökkäävä kortti tuhoaa sen, niin hyökkäämättä jättäneillä korteilla on vielä mahdollisuus kokeilla uudestaan. Yli kahta kertaa ei kannata kuitenkaan, koska guardianit eivät voi suojata guardianeita:

**AdvancedAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 1.1505973

Pelaajan 2 voitto-häviö-ratio: 0.8691138

**AdvancedAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 3.7016652

Pelaajan 2 voitto-häviö-ratio: 0.2701487

**SimpleAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.30511603

Pelaajan 2 voitto-häviö-ratio: 3.2774417

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 3.0584464

Pelaajan 2 voitto-häviö-ratio: 0.32696337

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.35652667

Pelaajan 2 voitto-häviö-ratio: 2.8048391

Nyt kaikki pienet mokat on korjattu, ja AI toimii juuri niin kuin sen oli tarkoitus. Se ei todellakaan ole optimaalinen, mutta antaa jo haastavan vastuksen pelaajaa vastaan ja voittaa aiemmat, paljon satunnaisuutta käyttäneet AI-toteutukset suurella voittoprosentilla. AdvancedAI on nyt mielestäni valmis.

Koska MediumAI eroaa mielestäni liian vähän SimpleAI:sta ja liikaa AdvancedAI:sta tällä hetkellä, lisäsin sille vielä tableAttackToKillit samaan tapaan kuin AdvancedAI:lla on:

**MediumAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 1.0069331

Pelaajan 2 voitto-häviö-ratio: 0.9931146

**MediumAI vs. SimpleAI:**

Pelaajan 1 voitto-häviö-ratio: 2.2516434

Pelaajan 2 voitto-häviö-ratio: 0.44412005

**SimpleAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 0.4531099

Pelaajan 2 voitto-häviö-ratio: 2.2069702

**AdvancedAI vs. MediumAI:**

Pelaajan 1 voitto-häviö-ratio: 1.6895906

Pelaajan 2 voitto-häviö-ratio: 0.59185934

**MediumAI vs. AdvancedAI:**

Pelaajan 1 voitto-häviö-ratio: 0.6654836

Pelaajan 2 voitto-häviö-ratio: 1.5026666

Voittoprosentit vaikuttavat mielestäni nyt sopivilta. AdvancedAI:n ja MediumAI:n välinen ero on nyt lähinnä järjestyksessä, missä minioneita yritetään pelata pöytään, ja AdvancedAI osaa pelata mounted minioneita tuhotakseen vastustajan kortteja.

**Yhteenveto (kurssin osalta):**

Kaikki kurssin aikana kehittämäni AI:t perustuvat pitkälti peräkkäin kutsuttaviin ehtolauseisiin, joiden järjestystä on parhaan mukaan pyritty optimoimaan peluuttamalla tekoälyjä keskenään. Algoritmit toiminnan takana eivät ole erityisen hienovaraisia tai nerokkaita eivätkä ne perustu mihinkään aiempaan teoriaan, mutta ne on kehitetty ja optimoitu pala palalta tavoitteena luoda

niistä tehokkaampia. Jokainen tehty ratkaisu on perusteltavissa, ja tästä päiväkirjasta löytyy niitä tukevia testituloksia ja spekulointia.

Lopputuloksena on erittäin tehokas tekoäly, joka näyttää pelaajan näkökulmasta tekevän järkeviä ratkaisuja. Lopullisessa pelin julkaisussa AdvancedAI:n kaltainen tekoäly menisi todennäköisesti vaikeusasteena Medium ja MediumAI olisi Easy. Haastavammaksi tekoälyksi tulen kehittämään vielä ratkaisun, joka luomalla joka vuoro kaikki mahdolliset skenaariot osaa valita optimaalisia ratkaisuja ja mahdollisesti laskelmoida tulevia vuoroja. Harmi etten kurssin aikana ehtinyt sitä toteuttaa, innostuin liikaa viilaamaan pilkkua AdvancedAI:n toteutuksessa ja en ollut varma miten sen toteuttaisin niin, että tulevaisuudessa lisättävät kortit eivät riko sitä. Siinä onkin sitten kesäksi tekemistä :)