Testausdokumentti

1. Mitä ja miten on testattu

Tekoälyn pelimenestystä on testattu 1000 kierroksen peleillä joita on toistettu 5 kertaa ja otettu keskiarvo kaikkien pelissä voitettujen ja hävittyjen erien suhteesta.

1.) Yleinen ohjelman tehokkuus:

- 1 000 000 kierrosta ai vs ai peliä (molemmilla 3 strategiaa käytössä) suoriutui keskimäärin noin 900:ssa millisekunnissa.
- 1000 kierrosta ai vs ai peliä (molemmilla käytössä patternmatching) suoriutui 100ms, 10 000 kierrosta 3000ms.
- Toisaalta ennalta-arvattavaa vastustajaa (Stupid-ai) vastaan 1 000 000 kierrosta patternmatchingiä suoriutui 1300ms ja voitti 100% eristä.

2.) Algoritmien kykyä pelata toisiaan vastaan ilman metastrategioita:

- MarkovFirst voittaa 60% MarkovSecond:iä vastaan pelatuista kierroksista.
- MarkovSecond voittaa 100% StupidAi:ta vastaan pelatuista kierroksista.
- PatternMatching voittaa 100% StupidAi:ta vastaan pelatuista kierroksista.
- PatternMatching voittaa 70% MarkovFirst:iä vastaan pelatuista kierroksista.
- PatternMatching voittaa 65% MarkovSecond:iä vastaan pelatuista kierroksista.
- Muut algoritmit olivat tasavahvuisia keskenään.

3.) Metastrategioiden määrän vaikutus pelimenestykseen toista tekoälyä vastaan:

- Yleisesti ottaen suurempi määrä metastrategioita antaa suuremman todennäköisyyden voittaa pelin
- 1 vs 4-6 metastrategiaa häviää 100% kierroksista.
- 2 vs 5-6 metastrategiaa häviää 80% kierroksista.
- 3 vs 4-6 metastrategiaa häviää hieman yli puolet kierroksista.
- Loput yhdistelmät olivat tasavahvuisia keskenään.

4.) Decayn vaikutus pelimenestykseen toista tekoälyä vastaan:

- Kun kaksi identtistä tekoälyä pelaa keskenään yhdellä metastrategialla, decay:tä käyttävä häviää noin 55% kierroksista.
- Kun kaksi identtistä tekoälyä pelaa keskenään kuudella metastrategialla, decay:tä käyttävä voittaa noin 55% kierroksista
- Johtopäätöksenä voidaan todeta, että decay:n käyttäminen tuo etua kun käytetään monta metastrategiaa.

5.) Eri algoritmiyhdistelmien pelimenestys ihmispelaajaa vastaan

- Automatisoitu testaus vaikeaa, koska tekoälyn ensimmäiset siirrot ovat satunnaisia ja voivat vaikuttavat pelin lopputulokseen yllättävän paljon.
- Tällä hetkellä TestPlayer (pelaa vanhoja pelejä automaattisesti) ~~voittaa lähes aina tekoälyn tietyllä valintayhdistelmällä. Ilmeisesti yksikään strategia ei vielä havaitse toistuvia kuvioita tarpeeksi tehokkaasti~~. Pattern-recognition-algoritmin avulla tekoäly onnistui voittamaan kyseisen pelin merkittävän useasti.
- MarkovFirst ja MarkovSecond yhdistelmää käyttävä tekoäly voittaa noin 60% itseäni vastaan pelatuista kierroksista ja 70% testihenkilöä vastaan pelatuista kierroksista.
- Yleensä kaikkien kuuden metastrategian käyttö on turhaa ja jopa haitallista, koska tekoäly voi "huomata" että pelaaja:lla olisi jokin monimutkainen strategia käytössä, vaikka todellisuudessa tätä ei olisi tapahtunut. Tällöin eri metastrategioiden välinen vaihtelu voi tuottaa epäloogisia valintoja pelaajan näkökulmasta.

6.) Luokkien testausta:

- Testattu että markovin malleihin perustuvat algoritmit antavat aina todennäköisimmän valinnan.
- Peli toimii oikein: tarkistaa pelaajien kädet ja päivittää tulokset.
- StrategyHandler pisteyttää strategiat viime kierroksen tulosten perusteella ja osaa valita parhaiten menestyneen metastrategian ehdotuksen.
- Testattu PatternMatchingiä pelaamalla sitä PatternMakeriä vastaan, missä PatternMaker luo toistuvaa, mutta kuitenkin vaihtuvaa kuviota. Mitä useammin kuvio vaihtui (eli mitä satunnaisempi kuvio oli), niin sitä huonommin patternMatching pärjäsi.

7.) Suorituskyky muita netissä olevia tekoälyjä vastaan.

(Pelasin oman ohjelman avustamana muita tekoälyjä vastaan. Laitoin nettipeliin tekoälyn ehdottamat kädet ja vastauksena annoin tekoälylle netin tekoälyn valinnat.)

- **nytimes.com**: Novice, 50 erää: Voitot: 28, Tasapelit: 6, Häviöt: 15
  - Metastrategioiden pisteytyksestä pystyi näkemään, että nettiversion tekoäly käytti jotain MarkovSecondin tapaista algoritmia, mihin oma tekoäly pystyi hyvin vastaamaan.
- **nytimes.com**: Veteran, 50 erää: Voitot: 17, Tasapelit: 20, Häviöt: 13
  - Peli oli tasainen koko matkan, eikä suuria eroja metastrategioiden pisteytyksessäkään syntynyt. Lopputuloksena tekoäly vaihtoi niin usein metastrategiaa, että peliä voisi kutsua täysin sattumanvaraiseksi. Loppua kohden oma tekoäly alkoi kuitenkin pärjätä, mutta en vielä vetäisi johtopäätöksiä siitä.
- **RPSContest.com**: zai_switch_markov1_ml: Voitot: 21, Tasapelit: 11, Häviöt: 10
  - Kyseinen algoritmi perustuu ilmeisesti myös markovin malleihin, mikä selittäisi oman tekoälyn menestyksen. Vastapuolella ei todennäköisesti ollut metastrategioita käytössä, koska sivuston ideanan on luokitella algoritmien menestystä ihmistä vastaan ja ihmiset harvemmin tekevät markovin malleja

vastustajan siirroista.

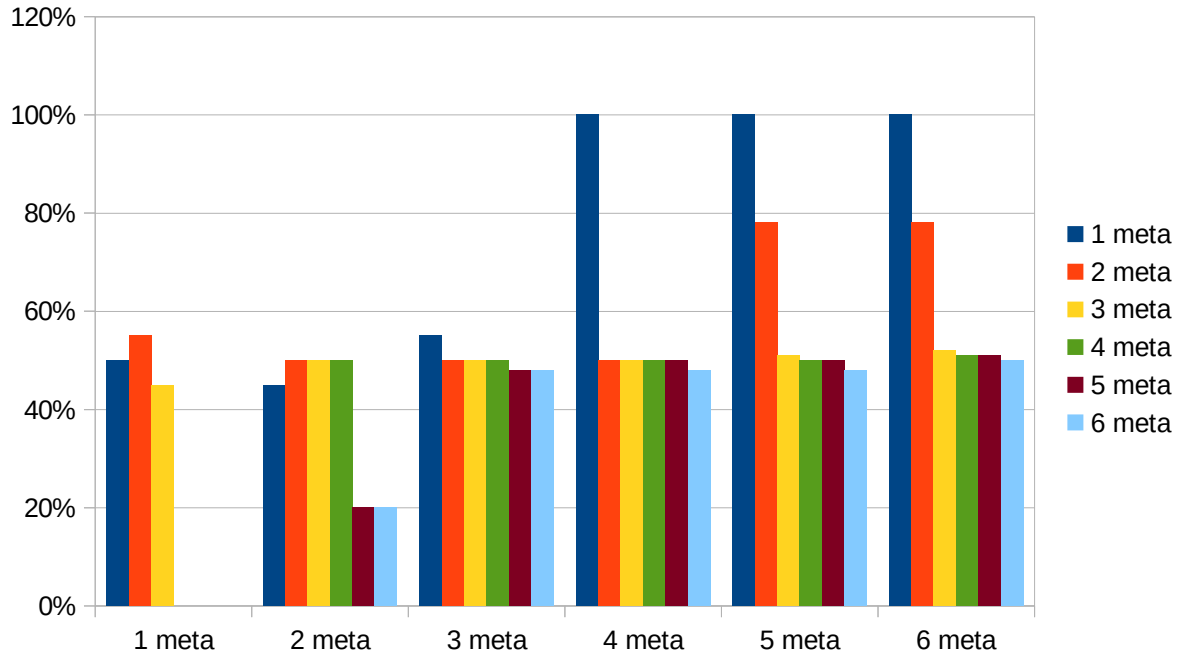
- **RPSContest.com**: fltbot: Voitot: 14, Tasapelit: 18, Häviöt 20
  - Tällä hetkellä sivuston parhaaksi rankattu algoritmi voitti melko selkeästi omani, eikä ihme sillä se perustui myös täysin samoihin metastrategioihin. Olisi mielenkiintoista selvittää mihin itse vastapuolen ennustusalgoritmi perustui ja luoda sille oma vasta-algoritmi.
- **Tekoäly "x":** Voitot: 33, Tasapelit: 10, Häviöt: 6
  - Tekoäly pystyi alun tasaisen pelin jälkeen tunnistamaan vastustajan käyttämän algoritmin (Pattern recognition) ja onnistui lopuksi voittamaan lähes joka siirrolla. Tehokkuus yllätti itsenikin.

3. Miten testit voidaan toistaa

Tekoäly vs Tekoäly-testien toisto vaatii Game-luokassa kahden StrategyHandlerin luomisen, tarvittavien parametrien käytön ja tarvittavien algoritmien lisäyksen StrategyHandleriin. Jos haluat testata itse tekoälyä, niin Game-luokassa täytyy lukea "p1 = new Player();" ja "p2 = new StrategyHandler(2, metastrategioiden_määrä, 0.95, onkoDecayPäällä);" sekä lisätä tarvittavat algoritmit p2:lle käsin.

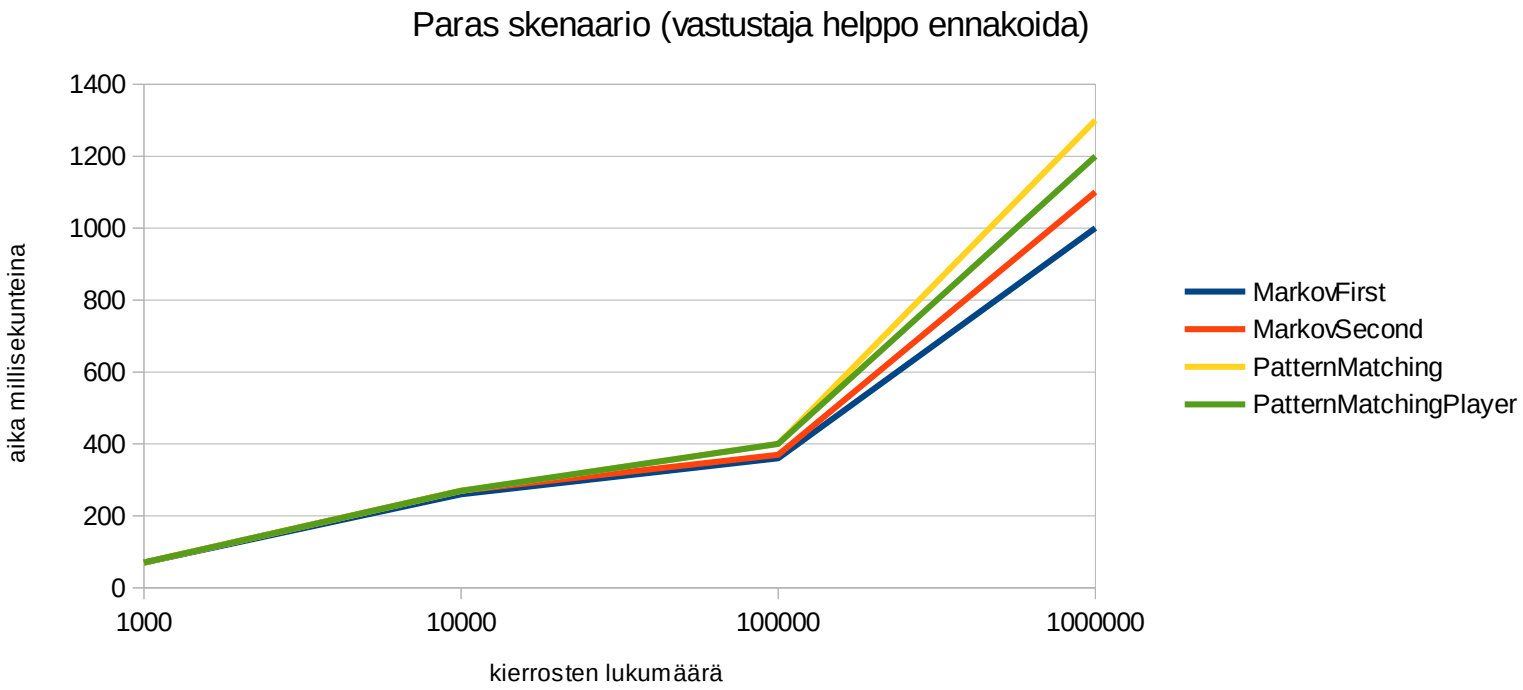
4. Tilastoja

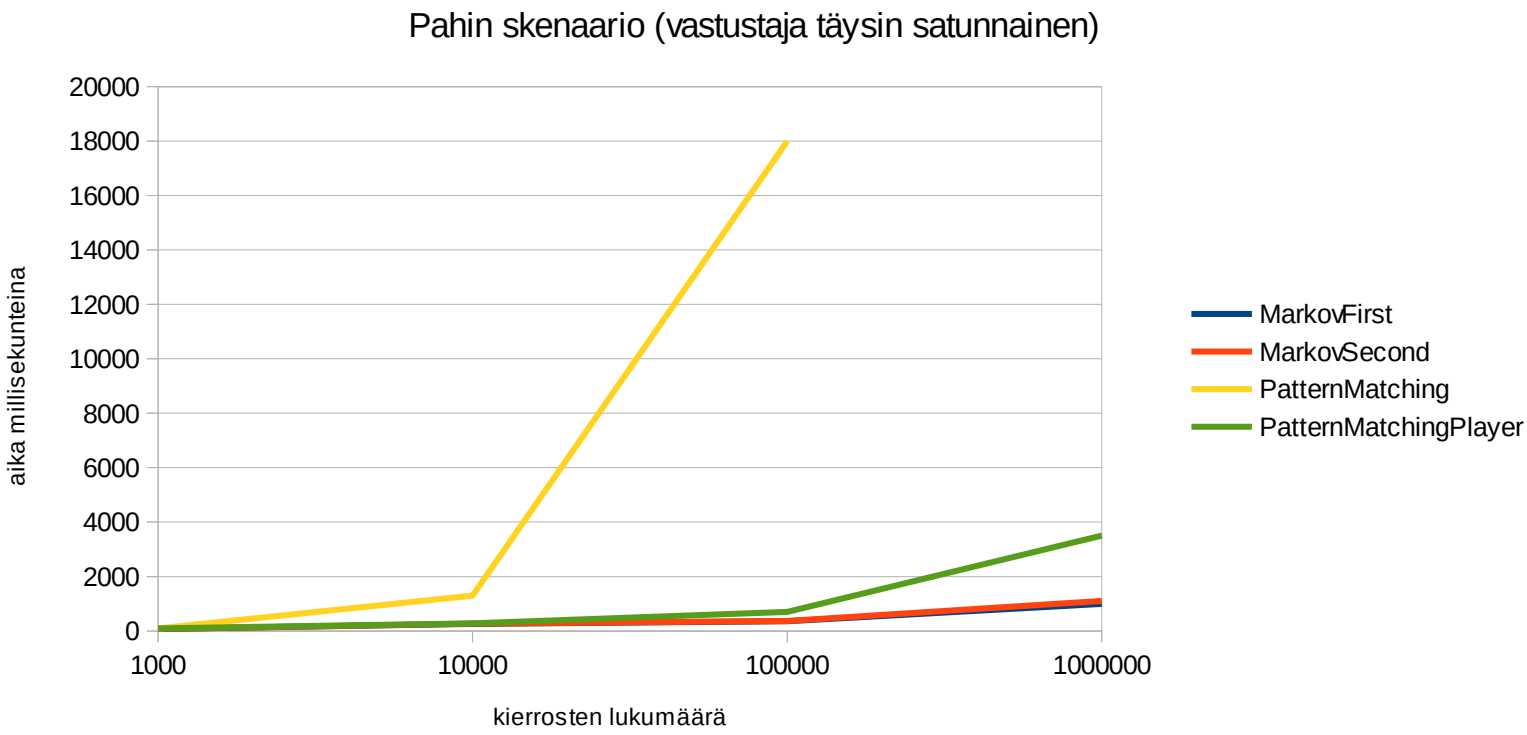
Metastrategioiden määrän vaikutus pelitulokseen toista tekoälyä vastaan.



Taulukossa tulokset eri metastrategioiden vaikutuksesta voittoprosenttiin kun käytössä on yksi algoritmi. Alhaalla on "pelaajan" metastrategioiden määrä, palkeissa on voittoprosentti eri metastrategioiden määrää vastaan. Esim "4 meta"-kohdasta nähdään kuinka 4 vs 1 metastrategia voittaa aina, koska neljäs metastrategia olettaa, että vastustaja käyttää samaa algoritmia pelaajaa vastaan.

Suorituskykykaavioita:





PatternMatching ja PatternMatchingPlayerin erot tulevat siitä, että ensimmäinen etsii toistuvuuksia molempien pelaajien siirroista, ja jälkimmäinen vain vastustajan siirroista. Jos vastustaja on täysin satunnainen niin on paljon todennäköisempää löytää vastaavuus pelkästään vastustajan siirtohistoriaa tarkastellessa, kuin vastustajan ja pelaajan yhteistä pelihistoriaa tarkastellessa.