MAUNULAN YHTEISKOULU
Helsingin matematiikkalukio

INHIMILLINEN SHAKKIKONE

Zhiyuan Liu Matias Manninen Kalle Wesanko

2024

Työn Kuvaus

Työn ideana oli kehittää shakkikone eli shakkia pelaava tekoäly, joka pyrkii pelaamaan kuin ihminen. Toisin sanoen, tavoitteena on kehittää tekoäly, joka pystyy ennustamaan ihmiskäyttäytymistä missä tahansa shakkiasemassa.

Tutkimukseen sisältyy shakkikoneen koodaaminen, inhimillisyyden määrittäminen tilastollisesti ja testiympäristön rakentaminen. Lisäksi hyödynnettiin hyperkuutio-otantamenetelmää (Latin hypercube sampling). [1]

Tutkimuksen yhteydessä kehitettiin prototyyppikone, ja hyödynnettiin myös valmiiksi kehitettyä Sunfish-tekoälyä. [2]

Menetelmät

Shakkikoneen ohjelmointiin sisältyy muutama osa:

- Logiikkayksikkö: Kokonaisuus, joka kuvaa esim. laillisten siirtojen tunnistamista, siirtojen suorittamista sekä pelin lopun määrittämistä.
- Hakualgoritmi: Shakkikoneen algoritmi, jolla se löytää asemasta riippuvaisen optimaalisen siirron. Tekoälyssämme *Sunfish* käytetään MTD-f hakua, mikä löytää aseman minimax-arvon. Näiden arvojen perusteella valitaan optimaalisin siirto. [3]

```
function Minimax(depth, state, maximizing):
    if depth == 0 or state is TERMINAL:
        return StaticEvaluation(state)
    if maximizing:
        eval = -INFINITY
        for child of state:
            eval = MAX(eval, Minimax(depth - 1, state, FALSE))
    else:
        eval = INFINITY
        for child of state:
            eval = MIN(eval, Minimax(depth - 1, state, TRUE))
```

Kuva 1. *minimax algoritmin pseudokoodi*

- Arviointifunktio: Shakkikoneen tapa arvioida asemien ominaista hyvyyttä. Sitä hyödynnetään hakualgoritmissä. [3]

Hakualgoritmi karsii ja vertailee siirtojen arvoa arviointifunktion perusteella. Tutkimuksessamme kehitetyt prototyypit omistavat primitiiviset arviointifunktiot, jotka ottavat huomioon materiaalimäärän sekä nappuloiden sijainnit. [3]

Shakkikoneen inhimillistäminen toteutettiin arviointifunktion säätelyllä. Valittiin yhteensä 10 parametriä (5 materiaalitekijää ja 5 sijaintiin liittyvää tekijää) ja tuotettiin otoksia parametrien arvoista hyperkuutio-otannalla.

Kuva 2. *Shakkikoneen PyCharm-käyttöliittymä siirtojen 1. e4 e5 2. Nf3 Nc6 3. Bb5 jälkeen*

Tämän jälkeen kehitettiin testiympäristö, joka pystyi mittaamaan parametrien arvoyhdistelmien pätevyyttä inhimillisten siirtojen löytämisessä. Tämä toteutettiin käytännössä antamalla säädetylle koneelle satunnaisia asemia, ja mittaamalla millä prosenttiosuudella ihmispelaajan siirto on myös arviointifunktion arvoyhdistelmän laskema siirto. Tätä arvoa kutsuttiin inhimillisyysosamääräksi.

Parametrien arvoyhdistelmien inhimillisyysosamäärän mittaaminen mahdollistaa inhimillisyyden ja parametrin hajontakuvion määrittämisen.

Yhteensä kehitettiin 3 shakkikonetta, jotka vastaavat 3 tasoa: 1000-1100, 1300-1400 ja 1600-1700 Elo.

Lopuksi tehtiin ristivertailutesti, jossa vertailtiin alkuperäisen ja optimoidun shakkikoneen inhimillisyyksiä. Tämä toteutui samalla testiympäristöllä, mutta pelien otoskoko laajennettiin 3000 peliin parantaakseen tilastollista tarkkuutta.

Tulokset

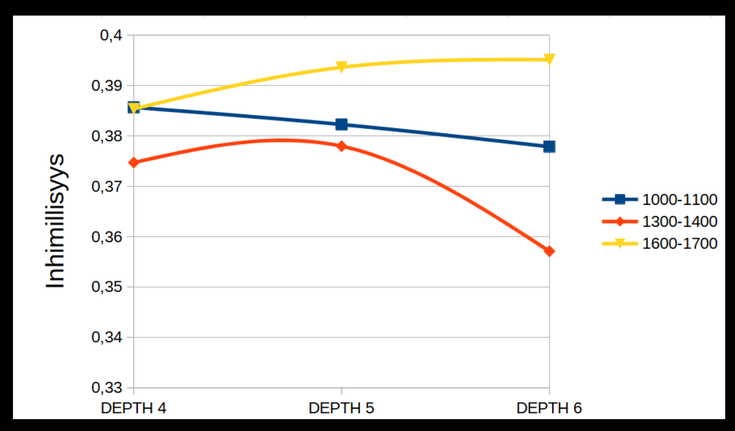
Sunfish-tekoälyyn pohjautuva prototyyppi pystyi pelaamaan noin 2000 Elo:n tasolla, kun syvyysasetelma on 7. Toisaalta, 7-syvyys haut kestävät yleensä alle 20 sekuntia.

Tilastollisten testien perusteella määriteltiin optimaaliset hakusyvyydet 1000-1100 Elo-tasolle (4), 1300-1400 Elo-tasolle (5) ja 1600-1700 Elo-tasolle (6).

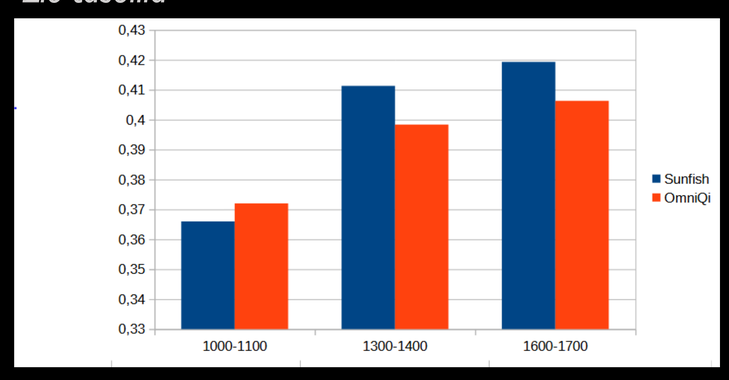
Lisäksi havaittiin tilastollisesti merkitsevä korrelaatio ($r = 0.469$, $p = 0.0372$) daamin arvon ja inhimillisyyden välillä, kun optimoitiin 1600–1700 Elo-tason shakkikonetta. Muissa hajontakuvioissa ei ilmennyt tilastollisesti merkitseviä korrelaatioita.

Ristivertailussa havaittiin, että pelkästään 1000-1100 Elo-tasolla optimoitu shakkikone pelasi inhimillisemmin kuin alkuperäinen.

Toisaalta, 1300-1400 sekä 1600-1700 Elo-tasolla opti-mointimenetelmä ei onnistunut, sillä säädetyt arviointifunktiot osoittautuivat vähemmän inhimillisiksi kuin alkuperäiset.



Kuva 3. *Hakusyvyyden vaikutus inhimillisyyteen eri Elo-tasoilla*



Kuva 4. *Shakkikoneiden inhimillisyysprosentti*

Johtopäätökset

Ristivertailutestin perusteella optimointimenetelmämme ei ole tehokkain tapa inhimillistää shakkikone, ja että sillä on pitkälti päinvastainen vaikutus inhimillisyysprosenttiin.

Hakusyvyyden positiivinen korrelaatio eli Elo-tasojen kanssa selittyy osittain sillä, että syvemmälle hakeva shakkikone pelaa paremmin, jolloin se kykenee emuloimaan parempaa ihmispelaajaa.

Havaitaan, että optimointimenetelmämme ei ole paras tapa inhimillistää shakkikonetta, vaan esimerkiksi neuroverkkoihin pohjautuva lähestymistapa on tehokkaampi.

Todetaan lisäksi, että shakkikoneen koodaaminen Pythonilla tuottanee lukuisia epäedullisuuksia liittyen laskentanopeuteen. Nämä ongelmat eivät olisi läsnä, mikäli käytettäisiin muita kieliä kuten C++.

Lähteet

1. McKay, M. D., et al (2000)

A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42.1 (2000): 55-61.

2. Ahle, T. (2014)

Sunfish. Github tietosäiliö. https://github.com/thomasahle/sunfish

3. Maharaj, S., et al (2020)

Chess AI: competing paradigms for machine intelligence. Entropy, 24(4), (s. 550).