Projektisuunnitelma Y2

1.

Nimi: Samuel Hyle

Opiskelijanumero: 900854

Koulutusohjelma: TiK

Vuosikurssi: 2021

Päiväys: 23.2.2024

2. Yleiskuvaus ja vaikeustaso:

Kahden pelaajan peli (tai yksi pelaaja + ”tekoäly” , joka noudattaa pelihistoriaan reagoivaa strategiaa, mikä voi sisältää satunnaistamista). Pelissä on kaksi yritystä, jotka tuottavat samaa hyödykettä, joilla molemmilla on sama kustannusrakenne ja tunnettu kysyntäkäyrä. Pelaajien tavoite on maksimoida oma kumulatiivinen voitto päättämällä hyödykkeelleen sopiva tuottomäärä. Molemmat pelaajat tietävät, että peliä pelataan vähintään N-periodia. N-periodin jälkeen jokaisen periodin lopussa on jokin todennäköisyys pelin päättymiselle (pelaajat eivät siis tiedä tarkkaa periodien määrää). Yksittäisen periodin tilanne on Cournot- duopoli, molemmat pelaajat päättävät siis tuotetusta määrästä samanaikaisesti, ja hyödykkeen hinta määräytyy kokonaistuotannon ja kysyntäkäyrän perusteella. Ohjelma varmistaa, että kysyntäkäyrän optimi ei ole kummassakaan ääripäässä.

Vaikeustaso jota lähdetään tavoittelemaan projektin kanssa on vaativa. Täytetään siis keskivaikean vaatimukset, eli ohjelma, jossa on interaktiivinen graafinen käyttöliittymä, yksikkötestit ohjelmalle, satunnaisuutta ja ohjelma näyttää pelin loputtua optimaaliset peliliikkeet/siirrot/määrät (nämä vähintään)

3. Käyttötapauskuvaus ja käyttöliittymän luonnos:

Oletetaan, että meillä on kaksi yritystä jotka kilpailevat keskenään. Nämä yritykset edustavat pelin pelaajia. Voimme luoda mallin, jossa molemmat pelaajat edustavat ’’oikeita’’ yrityksiä, missä molemmat ’’pelaajat’’ todella pelaavat peliä ilman koneellista apua ’’eli ns tekoälyn avulla’’, tai sitten voimme luoda mallin missä toinen pelaajista on ns ’’oikea pelaaja’’ ja toinen ’’tekoälynä’’ avulla luotu pelaaja, jonka toiminta perustuu valmiisiin koneellisiin malleihin parhaista strategioista ja ’’liikkeistä’’ pelin sisällä. ’’Oikean pelaajan’’ tapauksessa pelaaja itse vastaa kaikista tehdyistä valinnoista pelin sisällä eikä ole avustettu ’’tekoälyn’’ avulla.

Itse pelin tilanteet ja strategiset liikkeet määrittelemme peliteorian avulla. Nämä peliteoriaan perustuvat mallit tarvitsevat useita komponentteja, joista tärkein on pelin pelaajat. Pelaajat joita yritykset edustavat pelin sisällä oletetaan olevan rationaalisia yksityisiä päätöksentekijöitä. Pelaajat toimivat ja tekevät päätöksiä rationaalisin perustein yrittäen maksimoida pelin voitot tai ns ’’payoffit’’. Pelin tapauksessa voitot ovat yritysten tuotot, eli tuoton maksimointi on pelin tavoite.

Yksittäisen pelin voitot pohjautuvat pelaajien valintoihin tai ns ’’moveihin’’. Pelaajan valintoihin voidaan viitata ns. ’’Strategisena muuttujana’’. Yksittäisen pelaajan strategia on laajempi käsite kuin vain yksittäinen liike pelissä, sillä strategia sisältää jokaisen mahdollisen pelaajan liikkeen pelissä, pelin jokaisessa tilanteessa. Pelin tapauksessa, joka on epätäydellinen tietopeli, tai ns ’’yksinkertainen single phase move - peli’’, jokainen strategia on yksinkertaisesti todellisuudessa vain ’’liike / move’’, sillä pelaajilla ei ole dataa / tietoa, mihin pohjustaa liikkeensä.

4. Ohjelman rakennesuunitelma:

Kansiot ja luokat;

Projektin luomiseen hyödynnämme yleiseltä kannalta yläluokkia Game, Strategy ja Situation. Algoritmit määrittelemme erikseen luokissa BestResponse, InfoSets ja Stats. Yksittäisille peleille määrittelemme myös omat toiminta luokat DuopolyGame yms.

Luokka Game:
Perustapaus pelityypeille / Base class for game instances

Luokka Straregy:
Perustapaus strategioille / Base class for a strategy

Luokka Situation:
Käynnissä olevan pelin pelitiedot / Game state information

Luokka Stats:
Muodostaa pelin ’’play:n / liikkeen’’ annettujen strategioiden perusteella.

Luokka BestResponse:
Laskee parhaan mahdollisen vasta strategian peliteorian avulla. (Game tree reversal)

Luokka InfoSets:
Apuluokka, joka ottaa dataa tietojoukoista ja niiden elementeistä pelin historian ja strategioiden todennäköisyyksien avulla.

Luokka DuopolyGame:
Itse pelin ajava luokka joka pohjustaa ja luo uuden meneillään olevan pelin.

5. Tietorakenteet

Projektissa käytettävät tietorakenteet ovat pääosin listoja ja yksittäisiä arvoja, joita luokat palauttavat toistensa välillä strategioiden ja reaktioiden laskemiseksi.

6. Tiedostot ja tiedostoformaatit

Projektissa voidaan käyttää ulkoisia tiedostoja, joista pohjustaa peli, esim yritysten / pelaajien alustavat tiedot pelin käynnistämiseksi.

7. Algoritmit

Projektissa hyödynnetään esimerkiksi Nashin tasapainoa, joka on peliteorian ongelmien ratkaisukonsepti. Tässä tilanteessa vähintään kahden hengen peliteorian lopputulos on sellainen, ettei yksikään pelaajista voi saavuttaa parempaa lopputulosta itselleen muuttamalla yksipuolisesti omaa strategiaansa. Pythonissa tämän tilanteen kuvaaminen perustuu kolmeen pääkomponenttiin: pelin matriisin syöttämiseen, jokaisen solun tarkistaminen Nash-tasapainon tyyppi ominaisuuksien varalta, ja solujen tulostaminen jotka ovat tässä kyseisessä Nash-tasapainossa.

Projektin algoritmit hyödyntävät myös suurelta osin if - else väitteitä, joiden avulla saadaan määritettyä haluttuja lopputuloksia.

8. Testaussuunnitelma

Testauksessa keskitytään matemaattisten algoritmien paikkaansa pitävyyteen ja siihen, että tuotetut arvot joita algoritmit antavat pätevät tulosten kanssa, mitä odotettaisiin. Erityistä huomiota kiinnitetään ohjelman Nash-tasapainon selvityksen suoritukseen, johon pelimme pohjautuu suurilta osin.

9. Kirjastot ja muut työkalut

- PyQt
- Plotly?

10. Aikataulu

1. Välipalautus: Tähän mennessä olen saanut valmiiksi ohjelman perustan, eli niin sanotun koneiston joka pyörittää ohjelmaani. Tähän ei kuulu graafisia osia vielä, tai graafista käyttöliittymää. Eli päätarkoituksena olisi saada tähän mennessä valmiiksi Strategy, Game, BestResponse, Stats, Infosets ja Situation luokat. Tämä on suurimmaksi osaksi kaikki mitä ns. Ohjelmani ’backend’ puoleen kuuluu. Käyttäjän pitäisi pystyä kommunikoimaan ohjelman kanssa.
2. Välipalautus: Tähän mennessä ohjelma on tarkoitus olla jo siinä vaiheessa että olisi melkein lopullisessa muodossaan. Ohjelman kuuluisi pystyä suorittamaan kaikki sille annetut tehtävät ja palauttaa järkeviä tuloksia, ja hyödyntää näitä tuloksia graafisen käyttöliittymän avulla esittäen käyttäjälle näitä tuloksia erilaisin tavoin. Käyttäjän pitäisi pystyä kommunikoimaan ohjelman kanssa ohjelman käyttöliittymän kanssa interaktiivisesti. Ohjelman valmiista versiosta tässä kohtaa puuttuisi vain lopullista kosmeettista hienosäätöä.

11. Kirjallisuusviitteet

https://en.wikipedia.org/wiki/Duopoly

https://en.wikipedia.org/wiki/Oligopoly

https://en.wikipedia.org/wiki/Cournot_competition

https://janboone.github.io/municipality_healthcare_expenditure/solving_model.html

https://www.johannaseesaro.fi/l/monopoli-oligopoli-ja-duopoli/

https://users.ox.ac.uk/~sedm1375/Teaching/Micro/week7.2.pdf

https://en.wikipedia.org/wiki/Nash_equilibrium

12. Liitteet