Toteutusdokumentti, Eclipse calculator

Yleisrakenne

Ohjelman tarkoitus on tuottaa todennäköisyyslaskelmia Eclipse lauatapelin taistelutilanteiden mahdollisista lopputuloksista aiemmin viime vuonna tekemäni käyttöliittymän tarpeisiin.

Ohjelmaan kuuluu handler tiedosto, joka käsittelee ohjelman saapuvat pyynnöt ja lähettää lasketut todennäköisyydet eteenpäin JSON muodossa.

Handler käyttää core tiedoston tarjoamia funktioita, joista toinen muuttaa saadun datan ohjelman sisäisten funktioiden käyttämään muotoon ja toinen toimii välikappaleena handlerin ja sisäisten funktioiden välillä, palauttaen handlerin kaipaamat todennäköisyydet.

Internal kansion funktiot hoitavat todennäköisyyksien laskennan simuloiden aitoa pelitilannetta kolmen (tai useamman, mutta core on asetettu kutsumaan kolmea) kierroksen ajan. Jokainen alus koostuu tietystä määrästä hull arvoa ja tuhoutuu saadessaan osumia hull + 1:n verran. Ohjelmassa ylläpidetään tietoa kaikista mahdollisista osumien jakaumista, jolloin uusia osumia voidaan helposti lisätä. Todennäköisyys aluksen säilymiseen hengissä on helposti laskettavissa jakaumista ja myös uusien osumien lisääminen käy tehokkaasti.

Todennäköisyys pelaajien voittoon lasketaan suhdelukuna todennäköisyyksistä, jossa toisen pelaajan aluksista vähintään yksi on hengissä ja jokainen vastustajan alus on tuhoutunut. Mikäli edes yksi vastustajan aluksista ei voi olla tuhoutunut, on mahdollisuus voittoon 0%. Voitto-todennäköisyyden lisäksi ohjelma palauttaa jokaiselle alukselle todennäköisyyden olla hengissä taistelun jälkeen, laskettuna alukseen tallennetuista todennäköisyysjakaumista.

Käyttöliittymä löytyy osoitteesta: http://darth.kipsu.fi/EclipseCalculator/

Suorituskyky

Ohjelman suorituskykyä on testattu yksikkötestein, lisäämällä testeihin ajastus. Omia implementaatioita on verrattu Clojuren core funktioihin, jotta nähdään mikäli niiden suorituskyvyt ovat verrattavia. Omissa implementaatioissa on pyritty pääsemän suurinpiirtein core funktioiden tasolle suorituskyvyssä.

Esimerkikis my-map funktio on verrattavissa Clojuren mapv funktioon. Ensimmäisissä suorituskykytesteissä ilmeni että oma toteutukseni on suurilla syötteillä selvästi hitaampi ja syy löytyi Clojuren immutable vektoreista, joiden

muuttaminen on hidasta. Ongelma saatiin ratkottua muuttamalla käytetty vektori väliaikaisesti mutableksi. Testausraportissa on tarkemmin suorituskykyvertailuiden tuloksia.

O-analyysi

Määrittelydokumentissa koko ohjelman aikavaativuudeksi oli ennustettu O(n+k), jossa n = alusten määrä ja k = kaikkien tykkien määrä. Tähän ei päästy, sillä määrittelyvaiheessa ei vielä ollut tiedossa kaikkea mitä ohjelman tulee tehdä tuottaakseen oikeita tuloksia.

Määrittelydokumentissa hyökkäävän aluksen kohteet oli suunniteltu tallennettaviksi kekoon jolloin yhden kohteen haku olisi nopeaa. Kuitenkin laskiessani tarkemin tuloksia, huomasin että hyökkääjän aiheuttama vainko on tarpeellista laskea jopa jokaiselle vastustajan alukselle. Näin ollen jokaista alusta kohden on käytävä läpi pahimmassa tapauksessa kaikki vastustajan alukset jokaista erilaista tykkiään kohden ja aikavaativuus näyttääkin olevan muotoa:

n = kaikki alukset, m = vastustajan alukset, t = aluksen erilaisten tykkien määrä

$O(n \cdot m \cdot t)$

Maksimimäärä aluksia yhdessä taistelussa (jossa olisi siis kaikki molempien vastustajien alukset, lukuunottamatta hyökkääjän starbaseja) on 32, jolloin vastustajan aluksia olisi keskimäärin 16. Tykkien määrä yhdessä aluksessa voi olla korkeintaan 3 (koska kaikki saman arvoiset tykit lasketaan kerralla) mutta käytännössä erilaisia tykkejä on yleensä vain yhtä laatua (jolloin aikavaativuus on $O(n \cdot m)$). Näin ollen pahimmassa mahdollisessa tapauksessa kierrosten yhteenlaskettu lukumäärä on 1536, jonka suoritysaika on vielä hyvin nopea.

Vastaava toiminta suoritetaan erikseen myös missile tyyppisille aluksille ja O(n) aikavaativuuden omaavia toimintoja alussa, kun syöte muutetaan ohjelmalle sopivaan muotoon, sekä lopussa kun todennäköisyydet voittoon ja jokaisen aluksen osumatodennäköisyydet listataan. O-analyysissa tulee kuitenkin ottaa huomioon vain suurin yksittäinen arvo.

Suurin hidastava vaikutus ohjelman toimintaan tuntuukin olevan palvelimen tarjoaja Heroku, jonka ilmaiselle palvelintilalle ei ole luvattu nopeimpia toiminta-aikoja ja varsinkin ohjelman ensimmäinen käynnistäminen on hidasta. Ohjelma olisi hyvä siirtää omalle palvelimelleni mahdollisimman pian, mutta tällä hetkellä siellä ei ole tuke Clojurelle.

Parannusmahdollisuudet

Todennäköisyydet lasketaan nykyisellään kolmelta kierrokselta. Voisi olla parempi jos kierroksia laskettaisiin ”tarvittava” määrä, eli niin kauan kunnes jomman kumman pelaajan alukset saavuttavat tietyn todennäköisyyden ollakseen kaikki tuhoutuneita, tai vaihtoehtoisesti määritellä jokin toleranssi ja mikäli todennäköisyydet eivät enää muutu siitä suuremmilla arvoilla kierroksesta toiseen, voitaisiin päätellä että se on riittävän lähellä todellista arvoa.

Toinen vaihtoehto olisi antaa käyttäjälle mahdollisuus valita, montako kierrosta simuloidaan, jolloin olisi myös käyttäjän hallinnassa, kuinka kauan tuloksen iteroimisessa kestää.