

TDK-dolgozat

Tardi Tamás

**Debrecen
2022**

**Debreceni Egyetem
Informatika Kar**

**OPTIMÁLIS SZENZORELHELYEZÉS
MEGERŐSÍTŐ TANULÁSSAL A NAPRENDSZER
VÉDELMEHEZ**

Témavezető:
Dr. Hajdu András
egyetemi docens

Készítette:
Tardi Tamás
programtervező informatikus

**Debrecen
2022**

Tartalomjegyzék

1	<i>Bevezetés</i>	4
2	<i>Alapfogalmak</i>	7
2.1	Geometriai leírás	7
2.2	Feladat definiálása	8
2.3	Felhasznált adatok	10
2.4	Megerősítő tanulás	11
3	<i>Optimális elhelyezés meghatározásának módszertana</i>	13
3.1	A Föld megfigyelése (1. példa)	13
3.2	A Naprendszer megfigyelése (2. példa)	15
3.3	Egy aszteroida megfigyelése (3. példa)	17
3.4	Unity ML-Agents implementáció	17
4	<i>Eredmények</i>	21
4.1	A Föld megfigyelése	21
4.2	A Naprendszer megfigyelése	22
4.3	Egy aszteroida megfigyelése	25
5	<i>Diszkusszió</i>	27
5.1	Geometria	27
5.2	Az űr modellezése	27
5.3	Felhasznált technológiák	28
5.4	Továbbfejlesztési lehetőségek	29
6	<i>Összefoglalás, kontribúció</i>	31
7	<i>Irodalomjegyzék</i>	33
8	<i>Köszönetnyilvánítás</i>	36
9	<i>Saját munka leírása:</i>	37

1 Bevezetés

A gépi tanulás és mesterséges intelligencia fejlődésének köszönhetően jelentősen növekedett a megerősítő tanulóval foglalkozó megoldások gyakorisága. Ez, ahogy a [1] is mutatja, megmutatkozott a publikált cikkek növekvő számában is.

A megerősítő tanulás előnye abban nyilvánul meg a gépi tanulás többi módszerével szemben, hogy nincs szüksége tanító adathalmazra. A tanulás középpontjában egy ágens áll, amely döntések sorozatával, próbálgatás útján találja meg az optimális megoldást. Ennek tudatában nem is meglepő, hogy a kezdetekben az egyik legnépszerűbb kutatási terület a jól definiált szabályrendszerrel rendelkező játékok megoldása volt. Már az 1960-as évek elején megjelentek például a kezdetleges megerősítő tanulást alkalmazó algoritmusok a Tic-tac-toe játékra [2], de a köztudatba a módszertan is a 2016-ban megjelent AlphaGo [3] sikereinek köszönhetően tört be, az akkori Go világbajnok legyőzésével.

Napjainkra a megerősítő tanulás felhasználása már kiterjed a hétköznapi életben használatos eszközökre is. Használják már közlekedési lámpák optimalizálására [4], önvezető autók fejlesztésére, de a robotikában is megállja a helyét.

A dolgozat a megerősítő tanulás egyik típusfeladatával, az optimális elhelyezés problémájával foglalkozik. A cél úgy elhelyezni objektumokat a térben, hogy azok egy bizonyos jutalomfüggvény szerint maximális eredményt érjenek el. Ez a problémakör, mint megerősítő tanulóval megoldható feladat az utóbbi években kiemelt figyelmet kapott. A legkülönbözőbb témákban folytattak kutatásokat, mint például szenzorok elhelyezése az üvegházakban páratartalom és hőmérséklet mérésére [5], jelerősség maximalizálása a térben pilóta nélküli repülőgépek használatával [6], de jelent meg a probléma általánosításáról is közlemény [7]. A legnagyobb tudományos elismerést ezek közül talán mégis a Google Brain csapata vívta ki, akik a [8] [9] munkákban egy olyan megerősítő tanulást használó algoritmust mutattak be, ami a szakértőknél lényegesen gyorsabban és több szempontból is optimálisabban tudja a komponensek elhelyezését megtervezni egy nyomtatott áramkörön.

A feladattípust demonstráló megközelítéseket tanulmányozva megfigyelhető egy nagyjából egységes módszertan, ami alapján hatékonyan lehet kezelni az ilyesfajta problémákat. A megoldás a feladat Markov döntési folyamattá átírásával kezdődik. A kutatásokban az ágens

mindig egy a szenzorokat lehelyező/mozgató entitás, a döntés pedig annak lehelyezése/mozgatása a kiválasztott pozícióba. A környezet előállítása tipikusan a tér diszkrét felírásával kezdődik, így a probléma általánosan megfogalmazható a következő módon: a cél N darab szenzor elhelyezése véges sok lehetséges helyen úgy, hogy azok optimális lefedettséget biztosítsanak.

Bár a módszertan a főbb pontokon megegyezik a különböző kutatásokban, közel sem mondható el, hogy egy teljesen szabványos rendszert adna. Több megközelítés létezik például a szenzorok elhelyezésére. A legtöbb megoldás egy üres térből indul ki. Bizonyos problémákban segít, ha az ágens egyesével választja a szenzorok helyzetét, mert számolható a félkész állapotra is jutalom, léteznek viszont olyan kutatások is, ahol a legjobb eredményeket úgy érték el, hogy ezek egy lépésben kerülnek lehelyezésre. Ismertek azonban olyan problémák is, ahol az objektumok helyzete adott, és a feladat az optimális pozícióba való mozgatás az előre definiált maximális lépésszámmal [6]. Hasonlóképpen eltérő a különböző célok miatt a jutalomfüggvény is, hiszen feladattól függően mást jelenthet az optimális szó. A chipok gyártásánál például ugyanabban a környezetben maximalizálható a chip komponensek száma, a teljesítmény, de minimalizálható a vezetősáv hossz, az áramhasználat, és ezeknek kombinációja is.

A dolgozat a megerősítő tanulás alkalmazhatóságát egy időben változó, folytonos, geometriai takarásokat is tartalmazó modellben vizsgálja. Nevezetesen, három példán keresztül mutatja be, hogyan lehet megfigyelő kameraszensorokat optimálisan elhelyezni úgy, hogy azok a lehető legnagyobb lefedést biztosítsák a Föld felszínének/Naprendszernek a monitorozására.

A megfigyelőállomások elhelyezése a Föld vizsgálatára nem egy teljesen új probléma. Már a 80'-as évek elején elkészült egy módszer a műhold-konstellációk optimalizálására úgy, hogy az általuk definiált globális lefedettség maximális legyen [10], de azóta is folyamatosan jelennek meg a különféle lefedési problémákra szolgáló megoldások [11] [12]. A dolgozatban problémakört a teljes Naprendszerre kiterjesztjük. A feladat ekkor különböző égitestek (bolygók) megfigyelésével foglalkozik, aminek a nehézsége abban mutatkozik meg, hogy az előző példákkal ellentétben a geometria központi szerepet játszik. Kiemelt figyelmet kell fordítani a Naprendszer pontos modellezésére, az égitestek takarására, és a Naprendszer méretéből adódó kihívásokra.

Általánosságban kijelenthető, hogy a vizsgált problémakör már annyira komplex, hogy elméleti modellekkel kvázi nem leírható. A bemutatott megerősítő tanulást használó eljárás viszont képes azt kezelni, sőt, még összetettebb feladatokra is kiterjeszthető.

A példák feldolgozása a nyílt forráskódú Pythorch alapú Unity ML-Agents Toolkit segítségével történt. Ennek fő előnye, hogy tartalmazza az optimális lefedés problémájára legtöbbet használt PPO [13] algoritmust, valamint egyszerű implementációt biztosít Unity-val létrehozott környezetben az ágensek tanítására, de a példákban előírt geometriai feladatok szimulálására is tökéletes megoldást nyújt a játékmotor.

A dolgozat szerkezetét tekintve a feladatok bemutatása az alapfogalmak ismertetésével kezdődik, különös figyelmet fordítva a megerősítő tanulásra és a feladatban használt vonatkoztatási rendszerre. Ezt követi a feladatok Markov döntési folyamattá való átírásának módszertana, majd az eredmények bemutatása. Mivel ismereteink szerint a Naprendszer szintjén ilyen vizsgálatok még nem történtek, a megerősítő tanulással talált megoldásainkat a természetes elhelyezésnek tűnő, naiv megközelítésekkel hasonlítjuk össze. A dolgozat az implementációkból fakadó potenciális pontatlanságok tárgyalásával és továbbfejlesztési lehetőségekkel zárul.

2 Alapfogalmak

2.1 Geometriai leírás

Ahhoz, hogy a Naprendszer vizsgálható legyen, szükség van a különböző égitestek pozíciójára. A Nasa egyik űrközpontja, a Jet Propulsion Laboratory (JPL) évszázadokra visszamenőleg és a jövőre is nézve adatokat szolgáltat az égitestek helyzetéről. Ezeket táblázatokba összefoglalva csillagászati évkönyvekben jelenítik meg. A feladat megoldásához a DE421-es [14] évkönyv került felhasználásra, ez magába foglalja a Naprendszer bolygóinak, a Napnak és a Holdnak a pozícióit 1900-tól 2050-ig; példánkban erre az időintervallumra fókuszáltunk.

A csillagászati évkönyvekben az égitestek pozíciói tipikusan három értékkel vannak leírva, Alpha, Delta (szögek) és adott központtól (origó) való távolság. Mivel Unity-ben a legegyszerűbb 3D-s koordinátákkal (X/Y/Z koordinátahármas) számolni, célszerű ezeket konvertálni. A csillagászatban különböző problémákhoz különböző vonatkoztatási rendszereket használnak [15]. Az általunk választott keret a The International Terrestrial Reference System (ITRS) [16]. Ez egy olyan koordinátarendszer, aminek az origója a Föld tömegközéppontja, és ahogy a Föld forog, forog vele együtt maga a koordinátarendszer is.

Az évkönyvben leírt koordináták az égitestek tényleges helyzetét jelölik az űrben. A példákban viszont arra az információra van szükség, hogy a Földről megfigyelve az adott égitest hol jelenik meg az égen. Ezen pozíciók kiszámításában segít a Skyfield Python csomag [17]. A Skyfield arra lett tervezve, hogy egyszerűen meg lehessen kapni a Föld körül keringő műholdak, vagy egy csillagászati évkönyv alapján különböző bolygók, égitestek helyzetét. A DE421-ből kapott értékek ITRS-re konvertálásának menete a következő:

1. Föld tömegközéppontjának kiválasztása, mint megfigyelési pont
2. Vizsgálni kívánt égitest asztrometriai koordinátájának kiszámítása. Ez számításba veszi az időt, míg a fény a Földre ér. Például, ha egy objektum 10 fénypercre van, akkor arra a pozícióra van szükség, ahol az 10 perce volt. Ez a fontos korrekció biztosítja, hogy a megfigyelőállomások a valós képet lássák.
3. Látszólagos hely kiszámítása. További hatásokat is figyelembe vesz, mint például az aberráció, vagy a Föld gravitációjának hatása a fényre.

4. Az így kapott Geocentric Celestial Reference System (GCRS) XYZ koordináta konvertálása ITRS koordinátákká.

```
astrometric = barycentric_earth.observe(planet)
apparent = astrometric.apparent()
rotated_position = apparent.frame_xyz(itrs)
```

1. ábra. Adott bolygóhoz tartozó koordináták kiszámítása az ITRS vonatkoztatási rendszerben (2., 3. 4. lépés)

2.2 Feladat definiálása

A dolgozatban három példán keresztül kerül bemutatásra az optimális elhelyezés problémája. Az első, egy már elméleti és gyakorlati eszközökkel évtizedek óta vizsgált probléma átvezetése megerősítő tanulásba. Bár valódi újítást itt nem vezetünk be, a példa kiválóan alkalmas volt arra, hogy meggyőződhessünk róla, hogy a megerősítő tanuláson alapuló megközelítésünk kellően hatékony. A cél a Föld köré egy olyan műhold-konstelláció elhelyezése (lásd 2. ábra), ami a lehető legnagyobb globális lefedettséget garantálja.



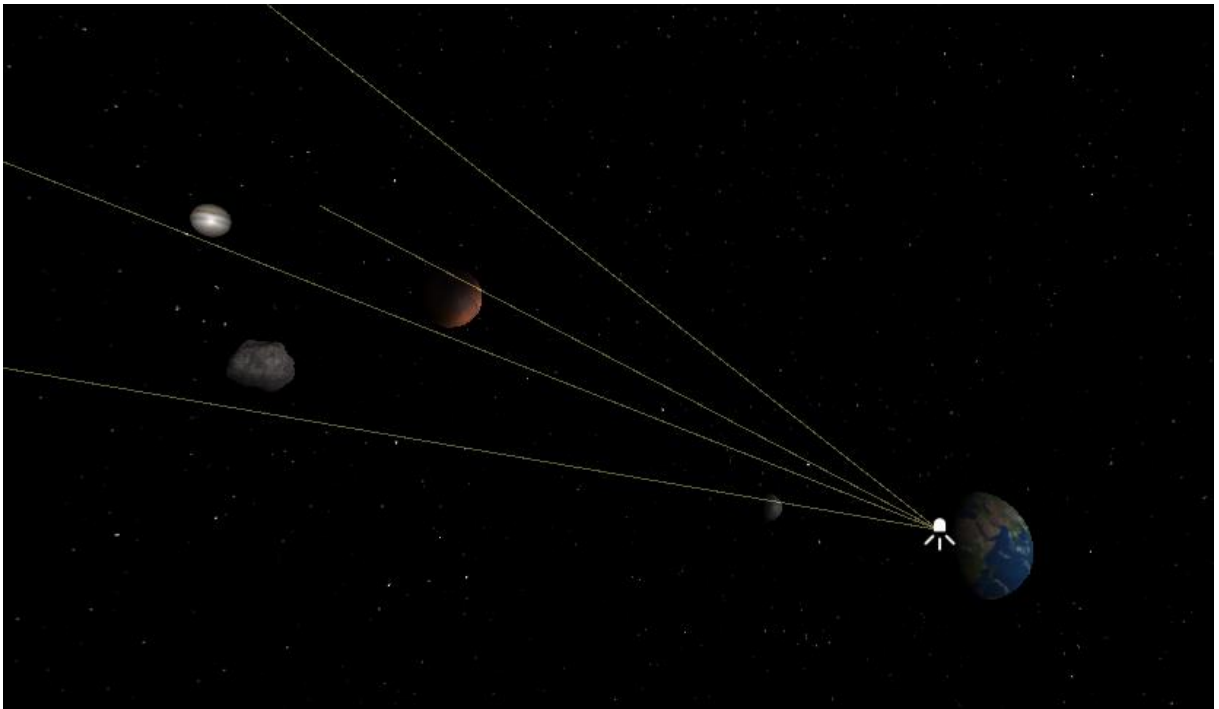
2. ábra. A Föld megfigyelése műhold-konstellációkkal Unity-ban.

Erről a területről már rengeteg közlemény jelent meg, van több megoldás a globális és regionális lefedettség optimalizálására is, használtak már genetikus algoritmusokat [12], de

voltak már hasonló témában megerősítő tanulós megközelítések is. Ezzel az egyszerű esettel kapcsolatban a célunk Unity-val létrehozott környezetben való szimuláció és megoldhatóság ellenőrzése volt.

A második és a harmadik példa a Naprendszerre – általánosabban a világűrre – fókuszál. A cél úgy elhelyezni a Földön megfigyelőállomásokat, hogy:

- azok minél több égitestet tudjanak egyszerre vizsgálni (2. példa), lásd 3. ábra,
- minél jobban meg tudják figyelni egy Föld felé tartó idegen objektum (például aszteroida) pályáját (3. példa)



3. ábra. A Naprendszer megfigyelése a Földről; Unity implementáció

A célunk tehát egy olyan védelmi megfigyelőállomás-rendszer elemei optimális helyének meghatározása, amely képes minél nagyobb lefedéssel folyamatosan monitorozni a Naprendszer égitesteit vagy a belépő objektumokat.

2.3 Felhasznált adatok

Ahhoz, hogy számszerűen mérni tudjuk az obszervatóriumok által biztosított lefedettséget, szükségünk van a megfigyelni kívánt égitestek számára, méretére, és minél több megfigyelésre. Ezek a paraméterek egy CSV fájlba kerültek legenerálásra. Ennek felépítése a következő:

- 1. sor: Az elhelyezendő állomások látószöge. Például 30; 20; 10 azt jelenti, hogy három darab megfigyelőállomást helyezünk el, amelyeknek látószöge sorra 30°, 20° és 10°.
- 2. sor: A megfigyelendő testek átmérője. A látószöghöz hasonlóan itt is annyi érték jelenik meg, ahány égitestet vizsgálunk. Az égitesteket szabályos gömbbel reprezentáljuk.
- 3. sor: A megfigyelendő testek fontossági értéke. Ha a Naprendszer bolygóit vizsgáljuk, megadhatjuk például, hogy a Jupiter utáni bolygók megfigyelése annyit érjen, mint az előtte lévők.
- 4. és későbbi sorok: Megfigyelések ITRS vonatkoztatási rendszerben X / Y / Z formátumban. Egy sor jelöl egy megfigyelést, aminek egy oszlopa az adott égitest helyzetét írja le. Például 2000 január elsején a Nap helyzete: -135,326901092604 / -1,79590187717909 / -57,6462921870569

A Naprendszer vizsgálatakor a 2. példában a megfigyelések a rendszer nagybolygóira, a Napra, Holdra, továbbá a Plutóra vonatkoznak. Ez a lista a 3. példában tovább bővül az aszteroida pályájának vizsgálatakor egy kitalált, 3000 km átmérőjű, a Föld felé érkező égitesttel. Az ezekre vonatkozó adatok a már az előző fejezetben említett Skyfield csomaggal kerültek legenerálása. A megfigyelések a 2. példában 2000 január 1.-től 2049 december 12-ig napi háromszor készültek, UTC időzónában 00:00-kor, 08:00-kor és 16:00-kor vizsgálva (54729 sor). A 3. példa 2025 február 22.-ei megfigyeléssel kezdődik és 2031 július 26.-aival zárul, ugyanolyan rendszerességgel, mint az előző példában (6945 sor). Az aszteroida helyzete a vizsgálat kezdetén megegyezik a Neptunusz 2000. január 1.-ei pozíciójával, a Földet pedig a záródátumra éri el, így összesen 31,0211 csillagászati egységnyi utat megtéve.

2.4 Megerősítő tanulás

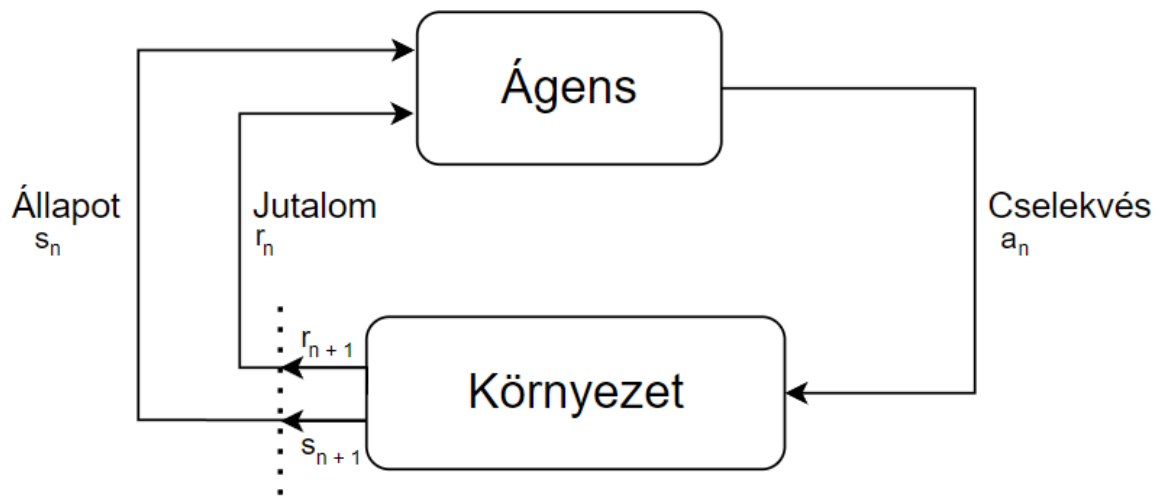
A megerősítő tanulás a gépi tanulás egyik ága. Olyan típusú feladatok megoldására szolgál, ahol egy ágensnek egy jól meghatározott környezetben kell döntések sorozatát meghoznia. Az optimális viselkedést a környezet visszajelzései alapján próbálgatás útján sajátítja el, tanulva a saját cselekvéseiből és azoknak a környezetre gyakorolt hatásából. A megerősítő tanulás alappillérei:

- Ágens: a cselekvéseket választó és végző entitás
- Környezet: a tér, amiben az ágens tevékenykedik
- Állapot: a környezet egy leírása az adott időpillanatban (megfigyelések halmaza)
- Cselekvés: az állapot ismeretében meghozott döntés, amivel az ágens interakcióba lép a környezettel
- Jutalom: egyfajta visszajelzés a környezet felől, ami alapján mérhető a választott cselekvés jósága
- Stratégia: a módszer, ami alapján az ágens a cselekvéseket választja

Az ilyen feladatokat Markov döntési folyamattal (továbbiakban MDP) szokás leírni. Ennek részei:

- Állapottér
- Cselekvéstér
- Jutalmak halmaza minden állapot – cselekvés kombinációra

A tanulás során az ágens az aktuális állapotban (s_n) választ egyet a rendelkezésre álló cselekvések közül (a_n), aminek a hatására a környezet a következő állapotba (s_{n+1}) kerül, és a jutalma a jutalomfüggvényben megfogalmazott módon nő (r_{n+1}). Ezt a folyamatot lépésnek, a kezdőállapotból egy végállapotba való eljutást pedig epizódnak nevezzük. A cél olyan stratégia megtalálása, ami az epizód során összegyűjtött jutalmat maximalizálja. A 4. ábra ezt a folyamatot mutatja be.



4. ábra. MDP folyamat ábrázolása

A példák megoldásához két mély megerősítő tanuláson alapuló technikát használtunk. Az első ilyen algoritmus a PPO [13] (Proximal Policy Optimization), aminek a mély mivoltát az adja, hogy neurális hálót használ a megfigyeléseket az optimális cselekvésre leképező függvény approximálásához. Az eljárás on-policy, tehát ugyanaz a stratégia kerül felhasználásra új cselekvések választására, mint a stratégia kiértékelésre nézve.

A másik módszer a SAC (Soft-Actor Critic) [18], ami a PPO-val ellentétben off-policy, tehát a tanulás közben a cselekvések választására más stratégiát használunk, mint a kiértékeléshez. Az algoritmus előnye, hogy a futás során menti a tapasztalatokat (milyen információt tudott meg, mikor az s_n -ből s_{n+1} -be lépett a_n -el, r_{n+1} jutalmat kapva), és ezeket a tanulás során véletlenszerűen felhasználja a stratégia frissítéséhez. Ez azt jelenti, hogy bár egy lépés az MDP-ben több értéket biztosít a SAC-nak (hiszen azt többször is felhasználhatja), a stratégia frissítése több időt vesz igénybe éppen e miatt.

3 Optimális elhelyezés meghatározásának módszertana

Annak érdekében, hogy meg lehessen találni az állomások optimális elhelyezését megerősítő tanulással, először is a problémát fel kell tudni írni MDP-ként. A három példa – mivel hasonló a cél – ilyen szempontból több ponton is megegyezik. Ilyen közös elem például a cselekvés. Bár a konkrét választható cselekvések eltérnek, minden feladatban a szerepe az lesz, hogy lehelyezzen egy objektumot (műhold/megfigyelőállomás). Hasonló módon közös rész az epizód menete is. Mindig ugyanúgy kezdődik, adott a környezet, ahol nincs lehelyezve egy objektum se, és addig tart, ameddig ezek el nem fogynak. Ekkor történik a jutalom számítás.

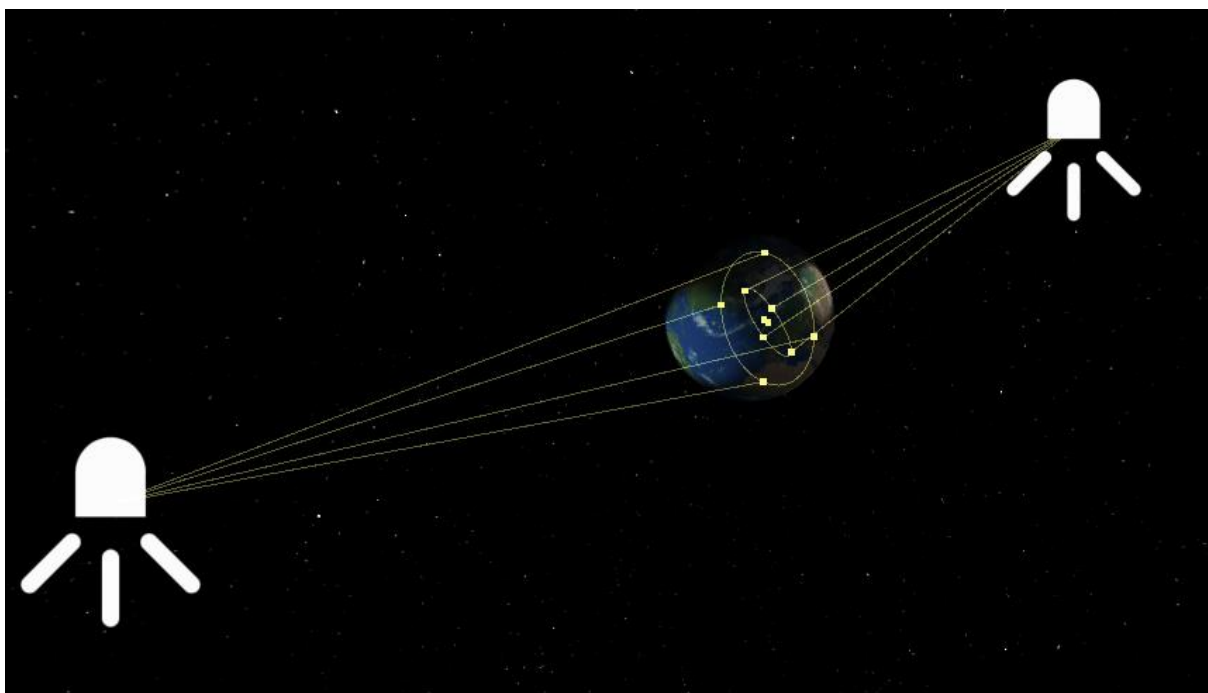
3.1 A Föld megfigyelése (1. példa)

A feladat a paraméterek megadásával kezdődik. Szükség van arra az információra, hogy hány darab műholdat kell lehelyezni, ezek milyen sebességgel mozognak, milyen távolságból, és milyen látószöggel figyelik a Földet. Miután ezek letisztázódtak, el lehet kezdeni az MDP konstruálását. Ebben az esetben a környezet egyszerű, a középpontban a Föld helyezkedik el; és köré kerülnek lehelyezésre a műholdak.

Ahogy az előző bekezdésben ismertetésre került, egy-egy lépésben a cselekvés egy műholdat helyez le. Mivel ezek a Föld körül keringenek, így egy pályát kell definiálni. Ezt az elvárt pontosságtól függően több módon is meg lehet tenni. A műholdak tipikusan kör, vagy körhöz közeli pályán keringenek a Föld körül. Ha nem megengedett az elliptikus pálya, elég megtudni, hogy honnan indul (ez lehet egy előre megadott pont, de tanulható is), és milyen irányba halad a műhold (összesen 5 paraméter). Ellenkező esetben viszont vagy a kezdeti koordinátát és a lendület-koordinátát kell tanulni (6 paraméter), vagy a pályaelemeket (6 paraméter). A cselekvés utáni megfigyelés tartalmazza minden műhold pályáját, valamint azt, hogy le lett-e már helyezve.

Miután már nincs több műhold, amit pályára lehet állítani, a jutalom számítása következik. Mivel a műholdak pályája ismert, így egyszerű egy szimulációt létrehozni, aminek a középpontjában a Föld forog, és körülötte keringenek a műholdak. A jutalom számítása megtörténhet bizonyos időközönként vagy akár minden alkalommal, amikor a Unity Update függvény meghívásra kerül (képkocka váltásonként).

A megerősítő tanulásnál kiemelt figyelmet kell fordítani arra, hogy a jutalom belátható időn belül számítható legyen, hiszen az optimális megoldás eléréséhez akár több millió epizód is szükséges lehet. Azt, hogy a műholdak optimális lefedést biztosítanak több módon is lehet definiálni. Ebben a bekezdésben bemutatásra kerül, hogy hogyan lehet a maximalizálandó jutalomfüggvényt kiváltani egy approximációval abban az esetben, ha annak a számítása túl költséges lenne. Ahogy az 5. ábra is mutatja, a műholdak felfoghatóak úgy, mint irányított fényforrások, amik a megadott látószögben világítják meg a Földet.



5. ábra. Műholdak, mint irányított fényforrások

Ekkor a következő lehetőségek a szimuláció során ekvivalensek:

- i. A fényforrások látószöge által definiált tölcsérek metszete minimális.
- ii. A Föld lehető legnagyobb százaléka van megvilágítva.
- iii. A Földön megjelenő körök távolsága páronként maximális (amennyiben megegyezik az összes műholdnak a látószöge, és a Földtől való távolsága).

A felsorolás a megvalósíthatóság szempontjából növekvő, a számítási költség szerint csökkenő sorrendben van.

Az i. megoldás legnagyobb hátránya, hogy a Unity beépített alakjai között nem jelenik meg a tölcsér. Bár ez külső programokból (Blender, Maya), vagy különböző csomagokból (probuilder) egyszerűen importálható, de nem tagja az alapformáknak, így a matematikai

függvények nincsenek külön erre optimalizálva. Ha a műholdak különböző látószöggel rendelkeznek, a formákat egyesével le kell generálni, mert a külső forrásból importált tölcések nem paraméterezhetők (látószög/magasság/hatótáv). Ezen kívül még a tölcések metszetének térfogatára is saját implementációt kell írni, ami a jutalom számítás gyakoriságának függvényében jelentősen le tudja lassítani a tanítást.

Az ii. megoldás erőforrásigénye szintén az elvárt pontosság mértékétől függ. Mivel a megvilágítás nem vizsgálható shader technológia nélkül (, ami rendkívül költséges), így más eszközt kell használni. Ismerve a látószöget és a Föld-műhold távolságot, meg lehet mondani, hogy mekkora a sugara annak a gömbnek, amin belül a Földet alkotó háromszögek/csúcsok meg vannak még világítva. Ezek után össze kell számolni az összes megvilágítottat. Minél részletesebb a gömb, annál több erőforrást igényel a kalkuláció.

Amennyiben tudjuk, hogy azonos távolságra helyezkednek el a műholdak a Földtől, és a látószögük is megegyezik, a iii. módszerrel a legegyszerűbb a számítás. Mivel ismert, hogy ezek a Föld középpontja felé néznek, elég egy sugarat küldeni, hogy megtudjuk a Föld felszínén kirajzolt kör középpontját. Ezen középpontok páronkénti távolságának maximalizálása választ ad az eredeti kérdésre.

3.2 A Naprendszer megfigyelése (2. példa)

A feladat megoldása a CSV feldolgozásával kezdődik. Mivel az egyetlen biztos pont a Föld, így kizárólag ebből áll a környezet az inicializálás előtt. A tanítás kezdetekor a program felolvassa a megoldandó problémát, és elmenti a megfigyelőközpontokhoz és az égitestekhez kapcsolódó adatokat. Ezek után mivel már ismeri az testeket, létre tudja hozni a környezetben a hozzájuk tartozó gömböket a megfelelő méretben.

A tanítás menete az előzőekben ismertetett módon zajlik, minden lépésben újabb megfigyelőállomás kerül elhelyezésre. Ennek módja, hogy a program cselekvésként szélességi és hosszúsági értékeket generál, ami definiál egy, a Föld felszínén lévő pontot. Ez a későbbi egyszerűbb felhasználás érdekében XYZ koordinátaként is mentésre kerül. A megfigyelés több módon lehetséges. Amellett, hogy megkapja a program, hogy le van-e helyezve az adott megfigyelőállomás, a helyzetét az előbb említett koordináta hármassal és a

szélességi/hosszúsági koordinátákkal is meg lehet adni. Tanítás során ez a választás nem volt szignifikáns teljesítmény szempontjából.

Az első példával ellentétben itt nem a szimuláció a legmegfelelőbb eszköz a jutalom számítására. A Föld forgásából és a műholdak sebességéből adódóan ott elég volt néhány nap alapján kiszámolni a lefedettséget (hiszen bizonyos idő után ismétlődik a szimuláció). Ebben az esetben viszont akár több év adatát is fel kell tudni dolgozni. Helyette a megfigyelésekre támaszkodik a jutalomért felelő algoritmus:

- Egyesével végigveszi a megfigyeléseket: az egyes bolygókat a megfigyelt pozícióba helyezi.
- Végigmegy a lehelyezett megfigyelőállomásokon.
- Megvizsgálja, hogy milyen égitesteket lát az adott megfigyelőállomás: ha az állomás és a test által bezárt szög kisebb, mint a látószög, akkor látja.
- Küld egy sugarat az égitesthez az állomástól: ha az beleütközik egy harmadik objektumba, akkor a test takarásban van.
- Ha mindkét vizsgálaton megfelel, és még nem tartalmazza a látott bolygók listája (tehát az eddig vizsgált obszervatóriumok közül egyik se látja), akkor hozzáadja.
- A lista alapján kiszámítható az egy megfigyelés alatt összeszedett jutalom: különböző megfigyelt bolygók és a hozzájuk rendelt fontossági értékek szorzata
- A végső jutalom a megfigyelések jutalmának átlaga osztva az egy megfigyelés alatt elérhető maximális pontszámmal. A tanításban a megfigyelések és a jutalmakat $[-1, 1]$ tartományba normáltuk.

```
for (int index = 0; index < problem.GeneratedPositions.Count; index++)
{
    List<String> distinctPlanetsSeen = new List<string>();
    setPlanetsToPosition(problem.GeneratedPositions[index]);
    foreach (var observatory in problem.Observatories)
    {
        List<String> planetsInCone = getAllPlanetsInCone(observatory, problem.GeneratedPositions[index],
            observatory.Angle);
        distinctPlanetsSeen.AddRange(collection: planetsInCone.Except(distinctPlanetsSeen));
    }
    reward += getRewardForPlanetsSeen(distinctPlanetsSeen);
}
```

6. ábra. Jutalom számításának algoritmusai

A választott vonatkoztatási rendszer előnye itt jelenik meg. Ha egy olyan koordinátarendszerben operálna a környezet, ahol a Föld forgását vagy keringését is figyelembe kell venni, akkor mivel a megfigyelőközpontok elhelyezkedése minden egyes epizódban változik, ezeknek pozícióját a Föld elfordulásának függvényében újra kellene számolni.

Az implementáció során voltak próbálkozások arra, hogy az algoritmus részeredményeket is kapjon lépésenként, viszont ez nem segítette az optimális megoldás megtalálását. Így a jutalom csak az epizód végén kerül kiszámításra.

3.3 Egy aszteroida megfigyelése (3. példa)

Mivel a feladat ugyanazt a környezetet használja, mint a 2. példa, a Naprendszer megfigyeléséhez, így a tanítás folyamatában nincs különbség. Ebben a példában az eddig vizsgált égitestek csak kitakaró szerepet játszanak, a fókuszban az aszteroida pályája van. Ez a változás csak a CSV fájlban jelenik meg, ami bővült a Földhöz közelítő kisbolygóhoz tartozó megfigyelésekkel, valamint átírással kerültek a fontossági értékek, miszerint jutalom csak akkor jár, ha nincs takarásban az aszteroida. Értelemszerűen, a belépő aszteroidát egy adott távolságon belülről kezdjük vizsgálni, amit a trajektóriaadatok egyébként definiálnak is.

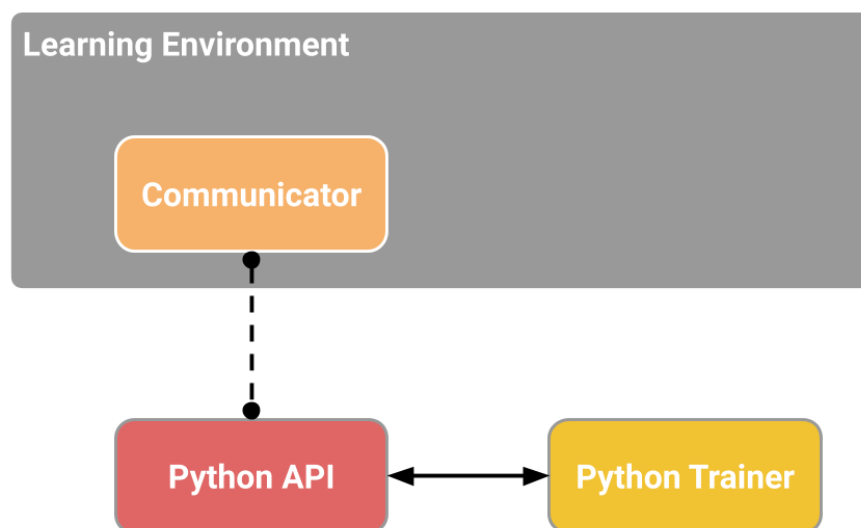
3.4 Unity ML-Agents implementáció

A Unity ML-Agents egy teljes keretrendszert kínál a megerősítéses tanulás implementálására. A folyamat központjában az ágens és annak a viselkedése áll, ezek alkotják a tanulási környezetet. Az ágens ebben az esetben egy játékbeli objektumhoz csatolt Agent osztály, ezt kiterjesztve tudjuk a saját implementációnkat megvalósítani. Ez tartalmazza az MDP nagyrészt, itt kerülnek értelmezésre a cselekvésekhez kapott értékek, az ágens itt kapja meg a megfigyeléseket a környezetéről, de a jutalomért felelős függvény is tipikusan itt van megvalósítva.

A viselkedés több különböző, a tanuláshoz elengedhetetlen attribútumot foglal magában. A tanítás megkezdése előtt meg kell adni a megfigyelések számát, valamint a cselekvések

paramétereit. Az utóbbinál külön ki kell térni arra, hogy ezek közül hány folytonos, és mennyi diszkrét. A viselkedés feladata, hogy az ágenstől kapott megfigyelésekre és jutalmakra válaszoljon a cselekvéshez szükséges információval. A Naprendszer megfigyelésénél például ez úgy jelenik meg, hogy a viselkedés megkapja az összes megfigyelőállomás helyzetét, és hogy az le van-e helyezve, erre válaszul legenerálásra kerülnek a cselekvéshez szükséges adatok, viszont ezeknek értelmezése (, hogy a következő állomást a kapott hosszúsági és szélességi értékekre helyezze) már az ágens feladata. Ez a két komponens alkotja a 5. ábrán bemutatott Learning Environment-et.

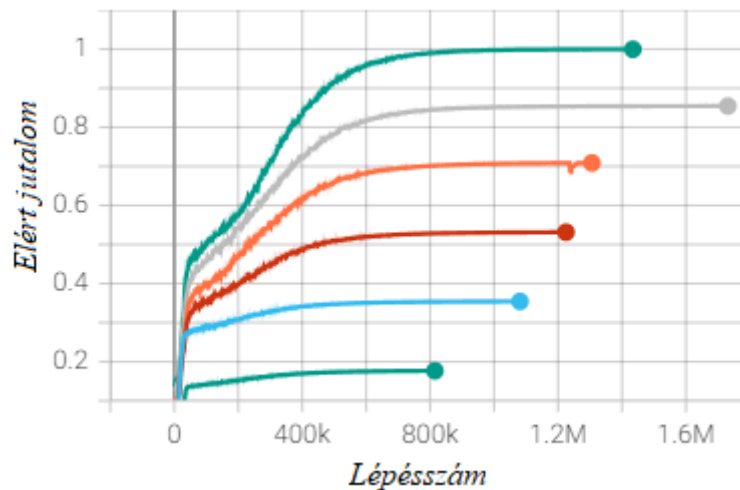
A tanítást az mlagents Python csomag használatával lehet megkezdeni, ez tartalmazza ugyanis az összes tanulásért felelős algoritmust. A Unity oldal ezeket csak a Communicator segítségével a Python API-n keresztül éri el.



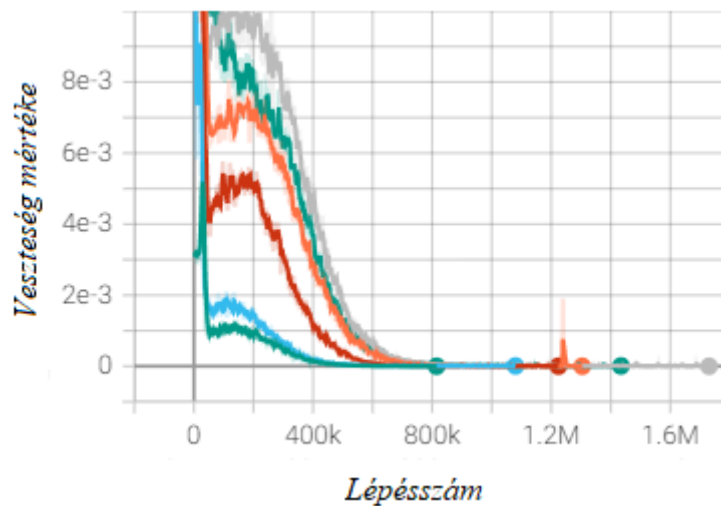
7. ábra. Unity ML-Agents felépítése

Ahogy ez az 6. ábrán látható, a tanulás sebessége és eredményessége nagyban függ a hiperparaméterektől. Bár már megszokott, hogy a gépi tanulás többi ágában rengeteg eszköz létezik ezeknek az optimalizálására (Scikit Optimize, Ray-Tune, stb), megerősítő tanulás esetében a komplexitás és a szimulációk szükségessége miatt ezek nem használhatók, így a hiperparaméterek hangolása manuálisan történt.

Támpontul szolgált ehhez az ML-Agents erre létrehozott dokumentációja, ahol külön ajánlást tesznek az egyes paraméterek értékére a használt algoritmus, a probléma nehézsége, és a cselekvéshez szükséges értékek típusára való tekintettel. Segített továbbá a Tensorboard, amivel végigkövethető a tanulási folyamat az eltelt idő függvényében. Olyan dolgokról kaphatunk információkat; mint az összeszedett jutalom (lásd 8. ábra), a stratégia veszteségfüggvénye (lásd 10. ábra), vagy a tanulási ráta alakulása.



8. ábra. Tanulás során felhalmozott jutalom a 3. példában (ábra a Tensorboard-ról)



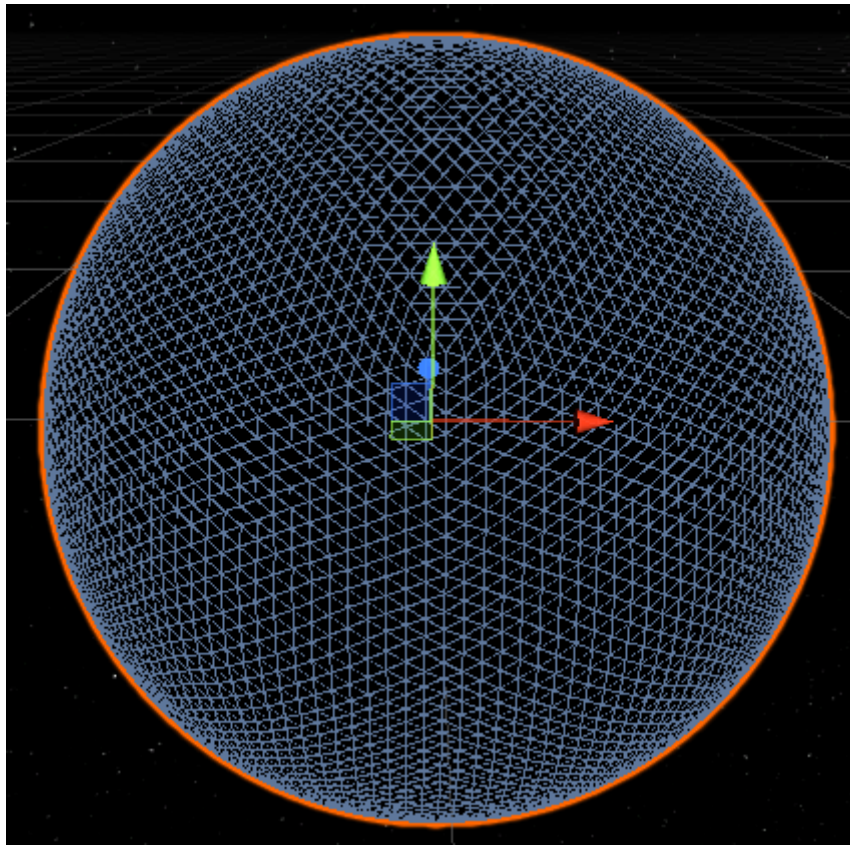
9. ábra. A veszteség mértékének alakulása a 3. példában. (ábra a Tensorboard-ról)

A tanítás során bizonyos időközönként mentésre kerül az aktuális neurális háló. Mivel az algoritmus futása közben, nehéz szemmel követni annak a helyességét (, az űr figyelésénél például szabad szemmel nem is látható minden égitest), így ezeket felhasználva könnyen betekintést nyerhetünk a tanítás aktuális állapotába. Az eredményeket a futás végén kinyert neurális háló felhasználásával kaptuk.

4 Eredmények

4.1 A Föld megfigyelése

Mivel ez a példa leginkább egy már megoldott probléma MDP-vé formalizálásának bemutatására szolgál, így elsősorban demonstratív jellegű. Az implementáció a PPO algoritmus felhasználásával sikeresen megtalálta az optimális, egyenletes kihelyezést 3 és 4 elemű műhold-konstellációk esetében is (egy pozícióból indítva, azonos magassággal). A jutalom számítására a felsorolt módszerek közül a második és a harmadik is implementálásra került. A Földet felszínét borító háromszöglapok megvilágítottságának számítása túl költségesnek bizonyult, a 10. ábra bemutatja a legnagyobb felbontású gömböt, ami fel lett erre a célra használva. Az ettől kevesebb háromszöget tartalmazóak nem tükrözték volna a tényleges lefedést a kívánt mértékben.



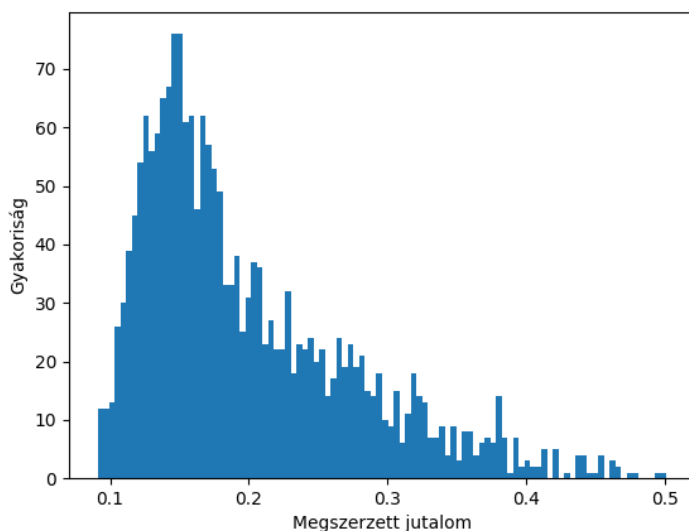
10. ábra. A Föld felszínét borító 61440 háromszöglap.

A Földre vetített sugarak középpontjainak páronkénti távolságának maximalizálásánál nem állt fenn ez a probléma.

4.2 A Naprendszer megfigyelése

Mivel ezt a komplex feladatot elméleti modellel leírni lényegében nem lehet, és mivel ismereteink szerint még nem készült rá megoldás, így számszerűleg nem tudjuk mihez hasonlítani a megerősítő tanulással kapott eredményeinket. Ilyen esetekben egy nagyon jó gyakorlati megoldásjavaslat [19] egy észszerűnek tűnő naiv modellhez való hasonlítás. Ennek megfelelően megvizsgáltuk, hogy mi lehet egy egyszerű, de jó viszonyítási alap a módszerünk kiértékeléséhez.

A legegyszerűbb – bár kevésbé hatékonyak ígérkező – módszer a megfigyelőállomások lehelyezésére az, ha a pozíciójukat a Föld felszínén véletlenszerűen állítjuk be. Tudjuk azonban, hogy minél közelebb vannak egymáshoz az állomások, annál nagyobb az általuk definiált tölcsérek metszete is, tehát nagyobb azon égitestek száma, amiket többen is megfigyelnek egyszerre. Mivel ez felesleges erőforrás-pocséklás, ezért N állomás esetén olyan pont- N -esek lettek véletlenszerűen legenerálva, amiknek a páronkénti távolsága egyenlő. A 11. ábra bemutatja az ilyen módon elhelyezett 2000 ponthármas ($N=3$) megszerzett jutalmának eloszlását. Vegyük észre, hogy ez a konstelláció egyenletes elhelyezést tekint. A 2000 tesztet már statisztikailag is szignifikáns eredményt mutat.



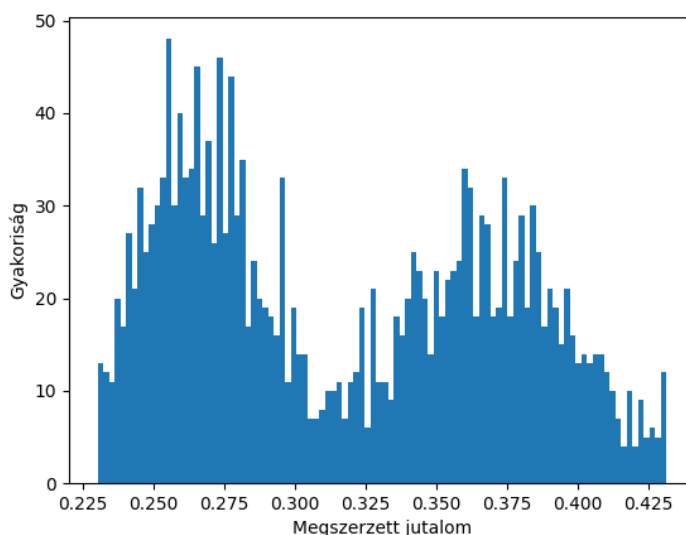
11. ábra. Jutalom eloszlása a Föld felszínén egyenlő távolságra elhelyezett megfigyelőállomások ($N = 3$) esetén

Ahogy az 1. táblázatban is látszik, ezzel a módszerrel a maximálisan elért jutalom 0,5 körüli, ami azt jelenti, hogy a megfigyelőállomások átlagosan valamivel több, mint 5 bolygót látnak a példában használt megfigyelésekben. Látható viszont az is, hogy az átlag és a medián is ennél lényegesen alacsonyabb.

1. táblázat. A jutalom fő statisztikai számai egyenlő távolságra elhelyezett megfigyelőállomások (N=3) esetén.

Minimum	Maximum	Átlag	Medián	Szórás
0,090928	0,501111	0,198783	0.174148	0.077069

Ismert, hogy a Föld Nap körüli keringése meghatároz egy, az egyenlítővel $23,5^\circ$ -ot bezáró síkot, amit ekliptikának neveznek. Mivel a Naprendszer formája leginkább egy lapított korongra hasonlít, így a nagybolygók is többé-kevésbé ebben a síkban mozognak. Ezen bolygók pályája az ekliptikával legfeljebb 3° , kivéve a Merkúr, ami 7° , a törpebolygónak minősített Pluto pedig 17° -os szöget zár be. Az előző megközelítésben annak köszönhetőek az alacsony átlag és medián értékek, hogy ez nem lett figyelembe véve, így több olyan pont-N-es is keletkezhetett, ami a sarkkörökhöz közeli. Ezek alapján tovább lehet fejleszteni a pont-N-esek naiv generálását: hozzunk létre olyan lehelyezéseket, amik az ekliptikával maximum 7° -os szöget zárnak be. Ahogy a 12. ábra is mutatja, így szignifikánsan jobb eloszlást lehet elérni.



12. ábra. Jutalom eloszlása a Föld felszínén az ekliptikával maximum 7° -ot bezáró sakra egyenlő távolságban elhelyezett megfigyelőállomások (N=3)

Ez megfigyelhető a 2. táblázatban is. Bár a maximum érték picit csökkent, látszik, ez az eljárás átlagosan lényegesen jobb eredményt tud elérni.

2. táblázat. A jutalom fő statisztikai számai az ekliptikával maximum 7°-ot bezáró síkra egyenlő távolságra elhelyezett megfigyelőállomások (N=3) esetén.

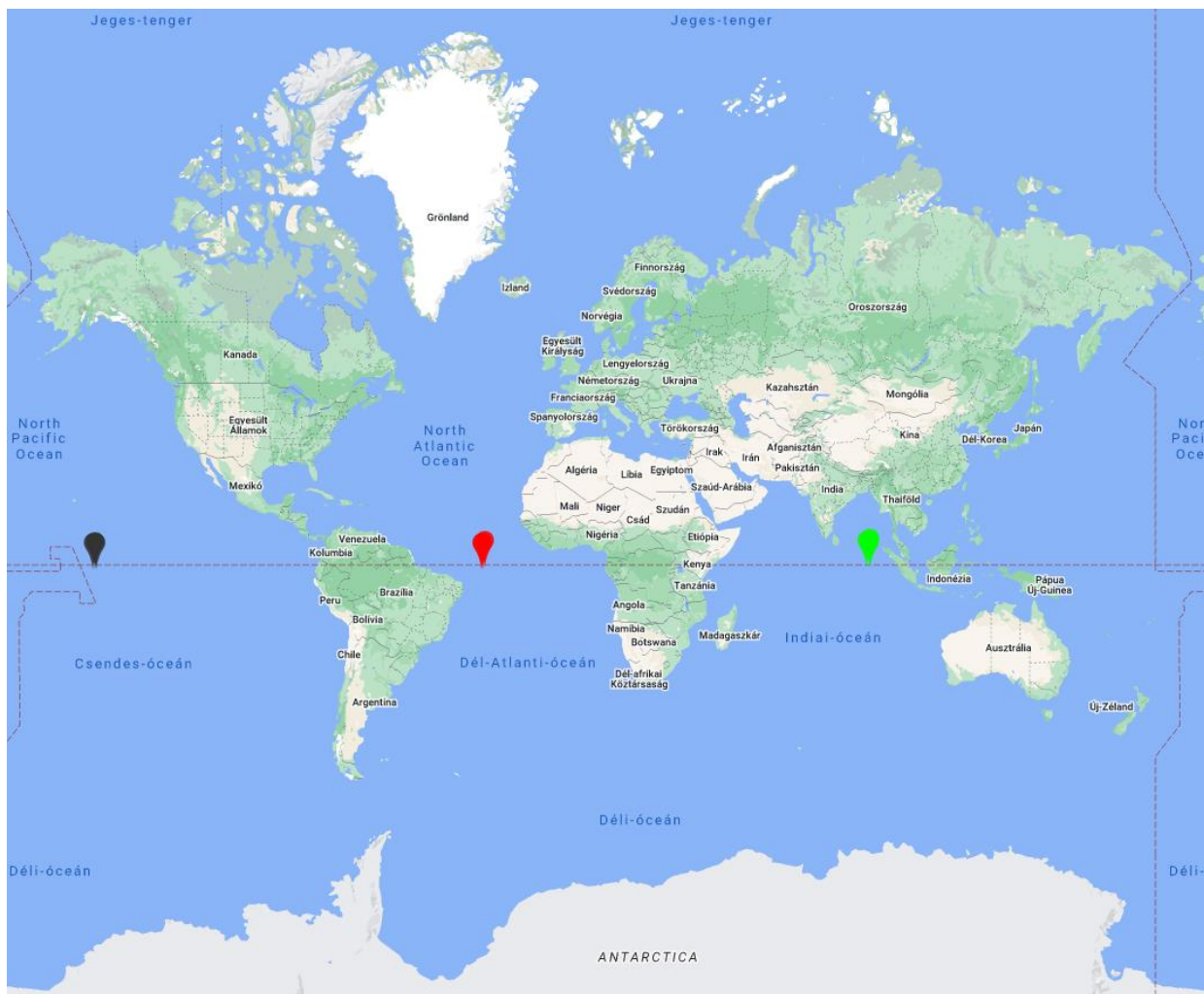
Minimum	Maximum	Átlag	Medián	Szórás
0,230212	0,431345	0,317839	0,311007	0.055732

A megerősítő tanulást használó módszerünk a már ismertetett módon keresi az optimális megoldást. A 2. példában az ML-Agents által nyújtott algoritmusok közül a PPO teljesített legjobban. Bár a SAC a tanítás korai szakaszában optimálisához közeli eredményt ér el, a maximálisan elérhető jutalomhoz csak nagyon lassan konvergál.

Ez a megoldás átlagosan 0,52 értékű jutalmat ért el 0,001-es szórás mellett. Az eredményt az 1. és 2. táblázattal összehasonlítva látható, hogy bár a naív megközelítés sem tűnik rossznak, a megerősítő tanulás alapú eljárás jobb és stabilabb eredményt biztosít. Az algoritmus által meghatározott modellt felhasználva megkaphatóak a konkrét állomás-pozíciók, amiket az optimálisnak ítélt. A 3. táblázatban megfigyelhető, hogy kisebb eltérésekkel, de alapvetően egy rögzített konstellációt talált az algoritmus optimálisnak.

3. táblázat. A PPO algoritmussal talált Föld felszíni pozíciók a 2. példára N=3 állomás elhelyezéséhez.

	1. állomás	2. állomás	3. állomás	Jutalom
szélesség	0.037468	-1.5047225	-0.8833312	0,519938231
hosszúság	91.5376511	-28.7876186	-149.139023	
szélesség	-0.9209747	-1.4075241	-1.275795	0,520126462
hosszúság	91.5599747	-29.0415173	-148.667511	
szélesség	-0.5494843	-0.7496718	-1.5331115	0,520119131
hosszúság	90.3300095	-28.2769871	-149.618423	



13. ábra. A 3. táblázatban megjelenő állomások elhelyezkedése a Földön

A 13. ábra bemutatja a 3. táblázatban ismertetett ponthármassok elhelyezkedését a Föld térképén. A naiv megközelítésekkel összehasonlítva az is látszik, hogy az állomások távolsága nem teljesen egyenletes, aminek oka, hogy a bolygómozgás sem az a vizsgált intervallumban; éppen ez az, amit elméleti modellel nagyon nehéz lenne leírni.

4.3 Egy aszteroida megfigyelése

Míg a Naprendszer megfigyelésénél az volt a cél, hogy megtudjuk, hogy adott számú megfigyelőállomással hány darab égitestet tudunk átlagosan megfigyelni, ebben a példában fontosabb az, hogy hány állomással lehet a legjobban figyelemmel kísérni az aszteroida pályáját. A 4. táblázat bemutatja az N lehelyezett obszervatórium által elért lefedettséget.

4. táblázat. Különböző számú megfigyelőközpontok által elért lefedettség egy aszteroida követéséhez

Megfigyelőközpontok száma (N)	Elért lefedettség (%)
1	17,8
2	35,5
3	53,1
4	68,3
5	85,4
6	99,9

A táblázatból látszik, hogy közel 100%-os lefedettség érhető el 6 megfigyelőközponttal. Ezek után akár úgy is tűnhet, hogy az égitestek takarásának nincs nagy jelentősége a jutalom számításában. Ennek kiderítése érdekében ideiglenesen kivételre került a takarás vizsgálatáért felelős kódrészlet, és újra vizsgálva lett ez az eset 5 db obszervatóriummal. Míg az eredeti példában így 85,4%-os lefedettséget értünk el, az egyszerűsített feltételekkel ez az érték 3%-kal nőtt.

A megfigyelőállomások helyének meghatározása ugyanazzal a megerősítő módszertannal történt, mint a 2. példában.

5 Diszkusszió

A megoldás kidolgozása és az implementáció során számos nehézség felmerült, a fejezet célja az ezekből fakadó potenciális hiányosságok és pontatlanságok ismertetése. A keretrendszer kidolgozása során az is fontos szempont volt, hogy könnyen adaptálható lehessen a modell bővülése esetén; a dolgozatot a potenciális továbbfejlesztési lehetőségekkel zárjuk.

5.1 Geometria

A geometriai pontatlanságok jelentik az első kihívást. Ilyen egyszerű megszorítás például, hogy minden égitest egy gömbbel van reprezentálva. Ennek több oka is van. Az egyik, hogy így sokkal egyszerűbb és gyorsabb számolni a hosszúsági és szélességi körök alapján az obszervatóriumok helyzetét, ami a tanítás szempontjából igen hasznos. A másik ok, hogy tovább bonyolítaná az algoritmust annak definiálása, hogy mit is jelent különböző komplex testeknél az, hogy “látja” azt a megfigyelőközpont. A gömbök esetében a példák azt vizsgálják, hogy a test középpontja benne van-e a látószögben, és hogy a középpontjához küldött sugár nem ütközik-e másik testbe. Ilyen szabályt nehezebb definiálni összetett objektumokra, speciális megoldásra lenne szükség, például:

- Csillagképekre: Ahhoz, hogy “lássa” ezt egy állomás, az összes alkotó csillagot meg kell tudni figyelni?
- Távoli galaxisokra: Komplex, korong formák általában az égen, látószögtől függően más szabály vonatkozhat rájuk. A Messier 104 galaxisra az éléről látunk rá például.
- Üstökösökre: A csóva is része? Idő függvényében változik az átmérője?
- Bolygók gyűrűjére: Külön figyelni kell a bolygó forgására és a látószögére.

5.2 Az űr modellezése

A következő kategória az űr modellezéséből és annak nehézségeiből ered. Az első példában a műholdak elhelyezésénél több változót is figyelmen kívül hagytunk. Ilyen például az, hogy a különböző Föld körüli pályák tipikusan más-más tulajdonsággal rendelkeznek, és ezért

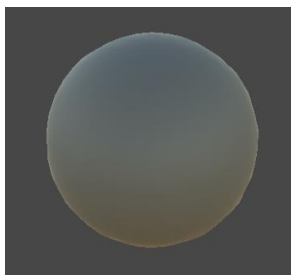
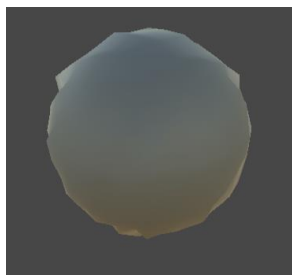
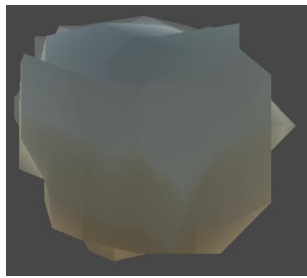
jellemzően más célt szolgáló műholdak kerülnek ezeken elhelyezésre. A magasság befolyásolhatja például a műholdra háruló fizikai hatásokat, annak sebességét és élettartamát is.

A későbbi példák nem hiába használják csak a Naprendszer égitesteit. Bár voltak próbálkozások további bolygók, és azok a holdjainak felhasználására, hogy teljesebb képet lehessen mutatni az űr lefedettségéről, nem állt rendelkezésre elég megfigyelés. A Mars holdjai például, bár az egyik legtöbbet tanulmányozott égitestek a naprendszerünkben, még mindig nem rendelkeznek pontos csillagászati évkönyvbéli leírással a jövőre nézve [20].

5.3 Felhasznált technológiák

Jelentős kihívás volt a feladatban használt skála kiválasztása. A Unity a pozíciókat Vector3-ként használja, ami tartalmazza az X, Y, Z értékeket. A probléma abból ered, hogy ezeket float típusként tárolja. Ez számunkra azt jelenti, hogy nem lehet az értékeket a végtelenségig kicsinyíteni, mert információt veszíthetünk. Nem lehet viszont az eredeti skálát vagy egy ahhoz közelit sem megtartani, mert ahogyan a 5. táblázat is mutatja, Unity-ben bizonyos távolság után elkezdenek torzulni a geometriai formák. A végleges arányosítás 1 Unity egység (UE) = 1 000 000 km lett, így a Föld átmérője 0,012742 UE, az X, Y, Z pedig az ~5500 maximális értéket veszi fel (Pluto X értéke).

5. táblázat. Objektumok alakjának változása (torzulása) a távolság függvényében

Pozíció (X, Y, Z)	(1000, 1000, 1000)	(5000, 5000, 5000)	(20000, 20000, 20000)
Megjelenítés			

Némi pontatlanság megjelenhet a SkyField csomagban is. Bár a tanulás menetét fel lehetett gyorsítani úgy, hogy a megfigyelések előre le lettek generálva, az obszervatórium pozícióival nem lehet így eljárni. Ez azt jelenti, hogy átlagos értékek lettek felhasználva, vagy esetenként

figyelman kívül hagyunk olyan paramétereket, amik bár minimálisan, de hatással vannak a látszólagos helyre. Ilyenek például a magasság, vagy a páratartalom.

5.4 Továbbfejlesztési lehetőségek

A bemutatott megerősítő tanuláson alapuló optimalizálási módszertan egy nagyon rugalmas keretrendszert biztosít, ami könnyen kiterjeszthető számos új irányba; ebben a szakaszban ezeket a motiváló lehetőségeket tekintjük át röviden.

A megerősítő tanulás egyik előnye, hogy ha már létezik egy jól definiált környezet a megfelelő cselekvésekkel és megfigyelésekkel, akkor, ha a cél változik, elég a jutalmat leíró függvényt módosítani. Az 1. példában, ha az ágens például nem azt kapja jutalmként, hogy a Földet felszínét alkotó háromszöglapok közül hány van megvilágítva, hanem azt, hogy az amerikai kontinens háromszögeinek hányad részét látja, akkor a tanulási folyamat már nem a globális, hanem regionális lefedettségre optimalizál.

Ennek épp ellenkezőjére is le lehet fordítani a feladatot. Ha nem az a cél, hogy minél több bolygót lehessen megfigyelni egyszerre a Naprendszerünkben, hanem hogy minél többet lássanak a megfigyelőállomások a világegyetemből, szintén csak a jutalomért felelős függvényt kell átírni. Ebben az esetben azért kap jutalmat az ágens, ha a megfigyelőközpontok által meghatározott tölcések metszete, és a látószögben lévő égitestek látszólagos átmérője minimális.

Akár össze is lehet kötni a megoldásokat további kutatásokkal. A [21]-ben bemutatásra kerülnek a gravitációs hullámokokat észlelő legfejlettebb obszervatóriumok. E közlemény célja, hogy ezek optimális lehelyezését keresse a Föld felszínén. A jutalomfüggvény átírásával maximalizálható ezen hullámok észlelése is.

Lehet változtatni a feladatot a környezet módosításával is. Ahogy láthattuk a Naprendszer megfigyelésénél az állomásokhoz megtalált optimális pozíciók mind az óceánba kerültek. Megoldható a cselekvéstér szűkítésével, hogy ezek a hosszúsági és szélességi értékek ne legyenek választhatóak. Felvetődött az is, hogy nem a Föld a legmegfelelőbb bolygó a megfigyelőközpontok elhelyezésére [22]. Könnyen átírható az implementáció, hogy egy másik

égitestre legyenek elhelyezve az állomások, ehhez a vonatkoztatási rendszer és a környezet középpontjának a megváltoztatása szükséges.

Akár össze is lehet vonni a példákat. Mivel már elkészült a jutalomfüggvény, amivel a látott objektumokat vizsgálhatjuk, és a műholdak keringése is meg lett oldva, átírható az implementáció annak optimalizálására, hogy az állomások egyszerre minél több műholdtól kaphassanak jelet.

6 Összefoglalás, kontribúció

A dolgozatban bemutatott kutatási/fejlesztési munka célja annak felderítése volt, hogy hogyan lehet a megerősítő tanulást, azon belül is az optimális lehelyezés problémáját felhasználni a Naprendszer megfigyelésére. Ennek bemutatása három példán keresztül történt meg. Az első, egy már évtizedek óta vizsgált, megoldottnak tekinthető probléma: hogyan lehet úgy elhelyezni a Föld köré egy műhold-konstellációt, hogy az a lehető legoptimálisabb globális lefedést biztosítsa. A rövid szakirodalmi áttekintés után a példa bemutatta az MDP-vé formalizálás módszertanát, annak nehézségeit, valamint a megoldás menetét a Unity ML-Agents Toolkit segítségével.

A további példák leírására kialakításra került egy paramétereizhető keretrendszer, amiben megadhatóak az állomások, a vizsgálni kívánt égitestek jellemzői és a hozzájuk tartozó megfigyelések. Az utóbbiak legenerálására készült egy algoritmus, ami a csillagászati évkönyvben szereplő pozíciókat Unity-ban is könnyen felhasználható, Földközpontú vonatkoztatási rendszerbeli (ITRS) 3D koordinátává tudja konvertálni. Ezek feldolgozásával a játékmotor inicializálni tudja a környezetet, amiben a Unity ML-Agents Toolkit felhasználásával intelligens ágensek taníthatók az optimális lehelyezés problémájának megoldására.

A keretrendszer sokszínűségét bemutatják a további példák, hiszen csak a paraméterek változtatásával két külön problémát tudunk megfogalmazni. A 2. példa azt vizsgálta, hogyan lehet a Föld felszínén úgy elhelyezni obszervatóriumokat, hogy azok minél többet tudjanak megfigyelni a Naprendszer égitestei közül egyszerre. Mivel nem áll rendelkezésre elméleti modell ennek becslésére, naiv megközelítésekkel került összehasonlításra a megerősítő tanulást alkalmazó megoldás. Míg az általunk felhasznált naiv technikák által legenerált legeredményesebb ponthármas 5 égitestet tudott megfigyelni átlagosan, a megerősítő tanulás ehhez képest sokkal konzisztensebben, 5,2 objektumot tudott egyszerre vizsgálni.

Az utolsó példa egy esetlegesen a Föld felé tartó aszteroida nyomon követéséhez határozza meg az állomások optimális helyét. A példában egy a 31 csillagászati egység távolságból érkező, 1500 km sugarú kisbolygót vizsgáltunk. Ezt 6 darab megfigyelőközponttal az idő közel 100%-ban meg tudtuk figyelni.

Bár a Unity tökéletes választásnak tűnt a geometriai feladat szimulálására, bemutatásra kerültek a fejlesztés során felmerült geometriai, modellezési, és a felhasznált technikákkal kapcsolatos egyéb problémák.

A dolgozat a továbbfejlesztési lehetőségek listájával zárul, megmutatva, hogy milyen alternatív feladatokat lehet megoldani pusztán az MDP különböző részeinek módosításával.

7 Irodalomjegyzék

- [1] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup és D. Meger, *Deep Reinforcement Learning that Matters*, arXiv, 2017.
- [2] D. Michie, „Experiments on the Mechanization of Game-Learning Part I. Characterization of the Model and its parameters,” *The Computer Journal*, pp. 232-236, 1963.
- [3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel és D. Hassabis, „Mastering the Game of Go with Deep Neural Networks and tree search,” *Nature*, pp. 484-489, 2016.
- [4] H. Wei, G. Zheng, H. Yao és Z. Li, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, United Kingdom, Association for Computing Machinery, 2018, p. 2496–2505.
- [5] D. D. Uyeh, B. I. Bassey, R. Mallipeddi, S. Asem-Hiablie, M. Amaizu, S. Woo, Y. Ha és T. Park, „A Reinforcement Learning Approach for Optimal Placement of Sensors in Protected Cultivation Systems,” *IEEE Access*, pp. 100781-100800, 2021.
- [6] E. Krijestorac, S. Hanna és D. Cabric, *Spatial Signal Strength Prediction using 3D Maps and Deep Learning*, arXiv, 2020.
- [7] Z. Wang, H.-X. Li és C. Chen, „Reinforcement Learning-Based Optimal Sensor Placement for Spatiotemporal Modeling,” *IEEE Transactions on Cybernetics*, pp. 2861-2871, 2020.
- [8] A. Goldie és A. Mirhoseini, *Placement Optimization with Deep Reinforcement Learning*, arXiv, 2020.

- [9] A. Mirhoseini, H. Pham, Q. . V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio és J. Dean, *Device Placement Optimization with Reinforcement Learning*, arXiv, 2017.
- [10] „Satellite Constellations,” *Journal of the British Interplanetary Society*, p. 559, 1984.
- [11] H. W. Lee, S. Shimizu, S. Yoshikawa és K. Ho, „Satellite Constellation Pattern Optimization for Complex Regional Coverage,” *Journal of Spacecraft and Rockets*, pp. 1309-1327, 2020.
- [12] C. Circi, T. Savitri, Y. Kim, S. Jo és H. Bang, „Satellite Constellation Orbit Design Optimization with Combined Genetic Algorithm and Semianalytical Approach,” *International Journal of Aerospace Engineering*, p. 17, 2017.
- [13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford és O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv, 2017.
- [14] W. M. Folkner, J. G. Williams és D. Boggs, „The Planetary and Lunar Ephemeris DE 421,” *Interplanetary Network Progress Report*, pp. 1-34, 2009.
- [15] P. Seidelmann és J. Kovalevsky, „Application of the new concepts and definitions (ICRS, CIP and CEO) in fundamental astronomy,” *Astronomy & Astrophysics - ASTRON ASTROPHYS*, pp. 341-351, 2002.
- [16] G. Petit és B. Luzum, „IERS conventions (2010),” *Tech. Rep. DTIC Document*, pp. 31-42, 2010.
- [17] B. Rhodes, *Skyfield: High precision research-grade positions for planets and Earth satellites generator*, Astrophysics Source Code Library, record ascl:1907.024, 2019.
- [18] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel és S. Levine, *Soft Actor-Critic Algorithms and Applications*, arXiv, 2018.
- [19] F. Chollet, *Deep learning with Python*, Simon and Schuster, 2021.

- [20] V. Lainey, A. Pasewaldt, V. Robert, P. Rosenblatt, R. Jaumann, J. Oberst, T. Roatsch, K. Willner, R. Ziese és W. Thuillot, „Mars moon ephemerides after 14 years of Mars Express data,” *A&A*, p. A64, 2021.
- [21] S. E. Gossan, E. D. Hall és S. M. Nissanke, „Optimizing the Third Generation of Gravitational-wave Observatories for Galactic Astrophysics,” *The Astrophysical Journal*, p. 231, 2022.
- [22] J. Silk, „Put telescopes on the far side of the Moon,” *Nature*, p. 6, 6 01 2018.

8 Köszönetnyilvánítás

Ezúton szeretném megköszönni Dr. Hajdu András témavezetőmnek, akinek szakértelme és tapasztalata nélkül nem sikerülhetett volna a dolgozat. Köszönettel tartozom Bogacsovics Gergőnek, aki, ha megakadtam az implementáció során mindig útba tudott igazítani. Hálás vagyok továbbá Dr. Tomán Henriettának, aki a segítette a dolgozatot a geometriai kihívások tisztázásával.

9 Saját munka leírása:

- Környezet kitalálása (vonatkoztatási rendszer, megfigyelések generálása Skyfield-el, CSV paraméterezhetőség)
- Példák MDP-vé alakítása
- Példák implementálása (Unity C# programozás, környezet kialakítása: ágens, viselkedés)
- Példák tanítása
- Naiv példák legenerálása

Lezárás dátuma: 2022.06