Univerzitet „Džemal Bijedić“
Fakultet informacionih tehnologija – FIT
Predmet: Umjetna inteligencija & Mašinsko učenje

Wumpus World

Umjetna inteligencija & Mašinsko učenje

Datum: 17/7/2023

Autori, studenti: Ermin Maksumić, IB190087
Haris Hindić, IB180015

Mentor, profesor: dr.sc. Migdat Hodžić

Sadržaj

Abstrakt

Wumpus World je klasična igra u kojoj agent istražuje lavirint ispunjen jamama, smradom, povjetarcem, čudovištem Wumpus i skrivenim blagom. Osnovni cilj igre, tj. Agenta jeste pronalazak blaga kroz donošenje odluka koje su u direktnoj korelaciji sa trenutnim okolnostima u kojima se agent nalazi. U ovom radu će se koristiti Q-tabela za implementaciju umjetne inteligencije koja će se kasnije kroz istu metodologiju iskoristiti i za mašinsko učenje. Kako bi se osigurao kontinuitet, radit će se pohranjivanje podataka, koji će se nakon sljedeće grupe iteracija učitati i iskoristiti kako bi agent izgradio znanje na prethodnim iskustvima i tako značajno smanjio vrijeme potrebno za ponovno učenje. Na temelju Q-tabele u radu će se pokušati prikazati kako umjetna inteligencija i mašinsko učenje u kombinaciji mogu pomoći pri treniranju agenta.

Ključne riječi: Wumpus World, mašinsko učenje, umjetna inteligencija, Q-tabela, iteracije

1. Uvod

U proteklih dvadeset godina, primjena umjetne inteligencije i mašinskog učenja je doživjela veliku ekspanziju, donoseći time velike benefite raznim industrijama, od automobilske, preko robotske, pa sve do gejming industrije. Primjenom raznih tehnika i metoda, igre su postale veoma značajan faktor pri upotrebi umjetne inteligencije i mašinskog učenja.

Kako je tema ovog rada Wumpus World, rad će se fokusirati na implementaciju Wumpus World igre, kroz prizmu umjetne inteligencije i mašinskog učenje.

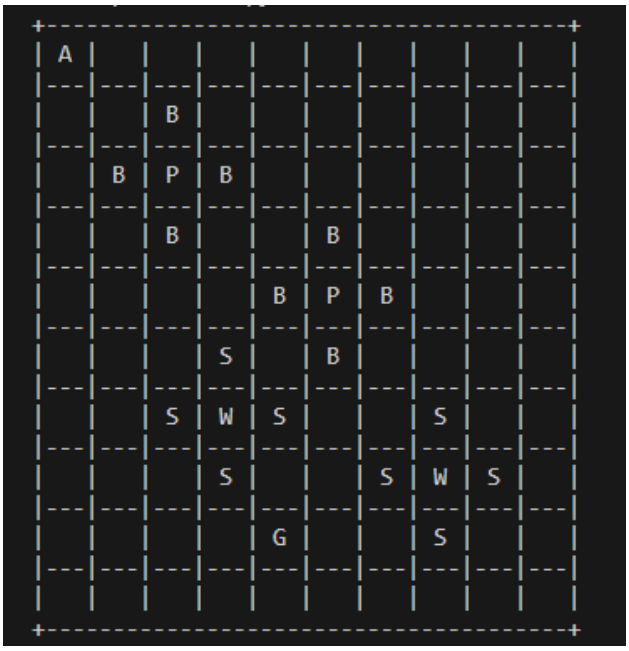
Wumpus World je veoma popularna igra kada je riječ o naprednim treniranjima agenta, kako bi na osnovu naučenog donosio ispravne odluke, dolazeći tako do željenog cilja. Spomenuta igra će biti sastavljena od iterativnog procesa učenja. Nakon svakog skupa iteracija (pr. 100 iteracija), sačuvat ćemo vrijednosti Q tabele, koje će se kasnije učitati i koristiti za nove grupe iteracije, kako bi agent imao što više prikupljenog znanja, koje će iskoristiti za donošenje optimalnijih odluka. Ovim će se postići značajan kontiunitet učenja agenta, koji se nalazi u neizvjesnom okruženju. Spomenuto neizvjesno okruženje je glavni izazov pri kretanju agenta, koji će, kroz nastojanje

implementacije umjetne inteligencije i mašinskog učenja, dobiti veliku pomoć u vidu kontinuiranog učenja.

2. Wumpus world kao igra

Wumpus world je jednostavna igra, koja se ipak može veoma usložniti ako se u nju uključe i napredne tehnike rješavanja problema. Wumpus World se može posmatrati kao lavirint, ili pećina, koja krije velike prepreke, u vidu provalija i Wumpusa. Da agent ipak ima značajne šanse za pronalazak zlata su se pobrinule strijele, kojima može ubiti Wumpusa, ali također i smrad i povjetarac koji ga mogu upozoriti da se u neposrednoj blizini nalazi ili Wumpus, ili provalija, te time biti upozoren na neposrednu opasnost.

Bodovi: 1000 bodova će agent dobiti ako pronađe zlato. Isto toliko bodova će mu se oduzeti ako ga uhvati Wumpus ili ako padne u provaliju. Jedan bod ću mu biti oduzet za svaki napravljeni korak, a izgubit će 10 bodova ako iskoristi strelicu u pokušaju da ubije Wumpusa.
Matrica igre: Igra se sastoji od matrice 10x10, dok se svi objekti igre stvaraju nasumično na matrici.



Slika 1. Matrica igrice

Za treniranje i učenje agenta koristi se Q tabela. Agent će time dobiti značajno unaprijeđenje performansi s obzirom da će moći iskoristiit naučene podatke u svakoj novoj iteraciji.

3. Q learning tabela

Kao što je već rečeno, za donošenje optimalnih strategija agenta, koristit će se Q tabela.
Q tabela je matrica koja pohranjuje vrijednosti akcija koje agent može poduzeti.
Svaki od elemenata u Q tablici predstavlja vrijednost Q(stanje, akcija). Vrijednost Q(stanje, akcija) predstavlja očekivanu nagradu koju agent može očekivati kada izvrši akciju a, u stanju s.
Početak učenja označava matrica koja ima sve vrijednosti postavljene na nula (0). Na osnovu agentovih kretanja, bilježe se podaci, te sam agent ažurira vrijednosti u tabeli koristeći Q learning algoritam.
Ova tehnika pomaže agentu da postepeno poboljavaša strategije donošenja odluka u igri. Agent će birati akciju sa najvišom vrijenošću Q (stanje, akcija) iz Q tabele. Međutim, ovo samo po sebi ne znači puno ako agent nema mogućnost istraživanja, koje se postiže odabirom nasumične akcije umjesot najbolje akcije. Epsilon (c) je vjerovatnost istraživanja koja se koristi prilikom ove implementacije.

Kako bi se postigli što bolji rezultati, i na najbolji način prikazala moć korištenja Q algoritma, u ovom radu će se nakon svake grupe iteracije pohranjivati postojeći podaci iz Q tabele, koji će se učitati prilikom iduće grupe iteracije, te time značajno poboljšati učenje i treniranje agenta, koji će temeljem postojećih podataka donositi optimalnije odluke tokom potrage za zlatom.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Slika 2. Primjer Q tabele

4. Implementacija umjetnog učenja u radu

Kroz ovu sekciju će se prikazati samo dva najvažnija dijela koda, bez kojih implementacija umjetne inteligencije ne bi bila moguća.

1. Ažuriranje Q tabele – ažuriranje q tabele se dešava nakon svakog novog koraka agenta. Za ažuriranje Q tabele je potrebno proslijediti stanje agenta, akciju, next state (pozicija agenta), nagradu, kao i learning i discount faktore.

```
def update_q_table(self, state, action, next_state, reward, learning_rate, discount_factor):
    x, y = state
    x_next, y_next = next_state

    current_q = self.q_table[x, y, action]
    max_q_next = np.max(self.q_table[x_next, y_next])

    new_q = (1 - learning_rate) * current_q + learning_rate * (reward + discount_factor * max_q_next)
    self.q_table[x, y, action] = new_q
```

Slika 3. Funkcija ažuriranja Q tabele

2. Automatiziranje igre – Automatiziranje igre se postiže ovim dijelom koda, gdje je važno spomenuti 3 parametra, koja su definisana na početku.

Learning rate je parametar koji kontroliše koliko brzo agent ažurira vrijednosti u Q tabeli na temelju novih informacija. Obično ima vrijednosti između 0 i 1.

Discount factor je parametar koji se koristi za vrednovanje budućih nagrada u odnosu na trenutne nagrade. Kada agent izvrši akciju u trenutnom stanje igre, za očekivati je dobivanje nagrada ne samo za tu akciju, već i za sve sljedeće akcije. Kao i learning rate, ima vrijednosti između 0 i 1.

```python
def automate_game(self,map:GameMap):
    learning_rate = 0.01
    discount_factor = 0.95
    epsilon = 0.1

    iterations = 100

    self.load_qtable_from_csv()

    for _ in range(iterations):
        self.initialize_game(map)
        self.display_grid()

        while True:
            state = self.agent_pos

            if self.is_wumpus_nearby() and self.arrows > 0:
                self.shoot_arrow(self.get_direction(self.agent_pos, self.get_next_position(random.randint(0, 3))))

                # Check if the agent found the gold or the game ended
                if self.agent_pos == self.gold_pos:
                    print("Congratulations! You found the gold.")
                    break
                elif self.score <= -1000:
                    print("Game over! You lost.")
                    break

            # Explore or exploit
            if random.random() < epsilon:
                # Explore: choose a random action
                action = random.randint(0, 3)  # 0: up, 1: down, 2: left, 3: right
            else:
                # Exploit: choose the action with the highest Q-value
                x, y = state
                q_values = self.q_table[x, y]
                action = np.argmax(q_values)

            direction = self.get_direction(self.agent_pos, self.get_next_position(action))
            self.move_agent(direction)
            self.display_grid()
            print("Score:", self.score)

            next_state = self.agent_pos
            reward = self.calculate_reward()

            # Update the Q-table
            self.update_q_table(state, action, next_state, reward, learning_rate, discount_factor)

            # Check if the game ended
            if self.game_ended:
                break
```

Slika 4. Automatizacija igre

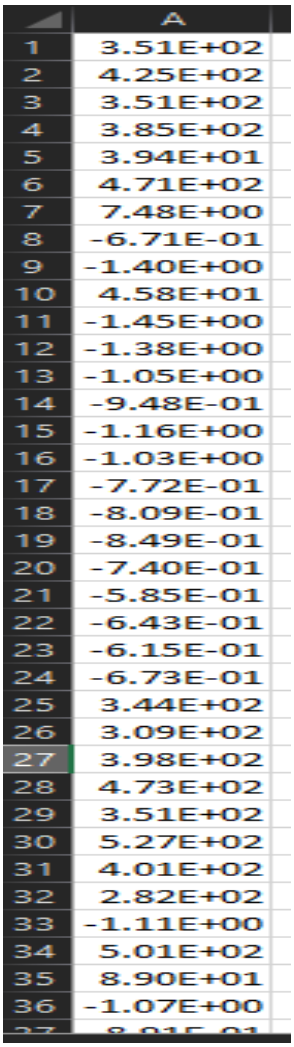
Epsilon je parametar koji se koristi za regulisanje istraživanje i iskorištavanje tog istraživanja u kontekstu donošenja odluka. U konkretnom slučaju se koristi za vrjerovatnost će agent odabrati nasumično akciju ujesto akcije sa najvećom vrijednošću Q (stanje, akcija).

5. Pohranjivanje i učitavanja podataka u svrhu korištenja istih za mašinskog učenje

U kodu se korištenjem pomoćnih funkcija pohranjuju podaci iz posljednje grupe iteracija, kako bi se prilikom idućeg pokretanja igre dobile prethodno naučene informacije, koje će agentu pomoći da donese optimalnije odluke. Korištene su dvije jednostavne funkcije za pohranjivanje i učitavanje podataka.

```
def save_qtable_to_csv(self):
    np.savetxt('qtable.csv', self.q_table.flatten(), delimiter=',')
```

Slika 5. Upisivanje podataka u CSV fajl



	A
1	3.51E+02
2	4.25E+02
3	3.51E+02
4	3.85E+02
5	3.94E+01
6	4.71E+02
7	7.48E+00
8	-6.71E-01
9	-1.40E+00
10	4.58E+01
11	-1.45E+00
12	-1.38E+00
13	-1.05E+00
14	-9.48E-01
15	-1.16E+00
16	-1.03E+00
17	-7.72E-01
18	-8.09E-01
19	-8.49E-01
20	-7.40E-01
21	-5.85E-01
22	-6.43E-01
23	-6.15E-01
24	-6.73E-01
25	3.44E+02
26	3.09E+02
27	3.98E+02
28	4.73E+02
29	3.51E+02
30	5.27E+02
31	4.01E+02
32	2.82E+02
33	-1.11E+00
34	5.01E+02
35	8.90E+01
36	-1.07E+00

Slika 7. Primjer zapisa u CSV fajlu

Korištenje podataka učitanih u CSV file

Nakon pokretanja druge i svake sljedeće grupe iteracija, učitat će se postojeći podaci koji su iskorišteni za mašinsko učenje, a koji će pomoći agentu prilikom donošenja odluka.

```
def load_qtable_from_csv(self):
    #Read the Q-table from the CSV file
    qtable_loaded = np.loadtxt('qtable.csv', delimiter=',')

    #Reshape the loaded Q-table to the desired shape
    self.q_table = qtable_loaded.reshape((self.size, self.size, 4))
```

Slika 8. Učitavanje podataka iz CSV fajla

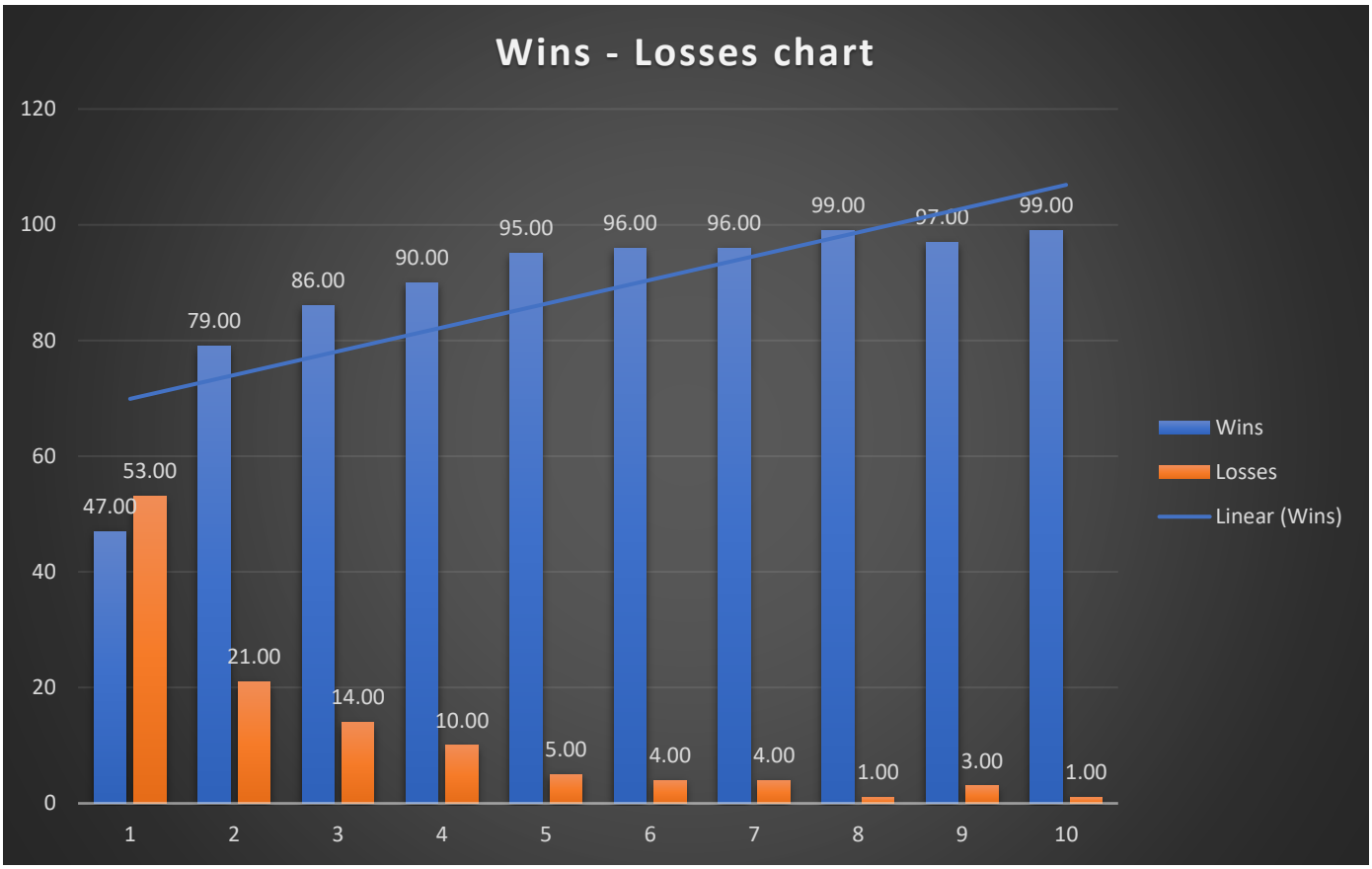
6. Korištenje dobivenih podataka

Prilikom korištenja specifične mape, izvršeno je 10 ponavljanja po 100 iteracija igrice, kako bi se prikazalo koliko je agent uspio naučiti i iskoristiti znanje stečeno iz prethodnih iteracija, uzimajući u obzir da pri prvoj iteraciji agent kreće od 0, tačnije bez ikakvog predznanja.

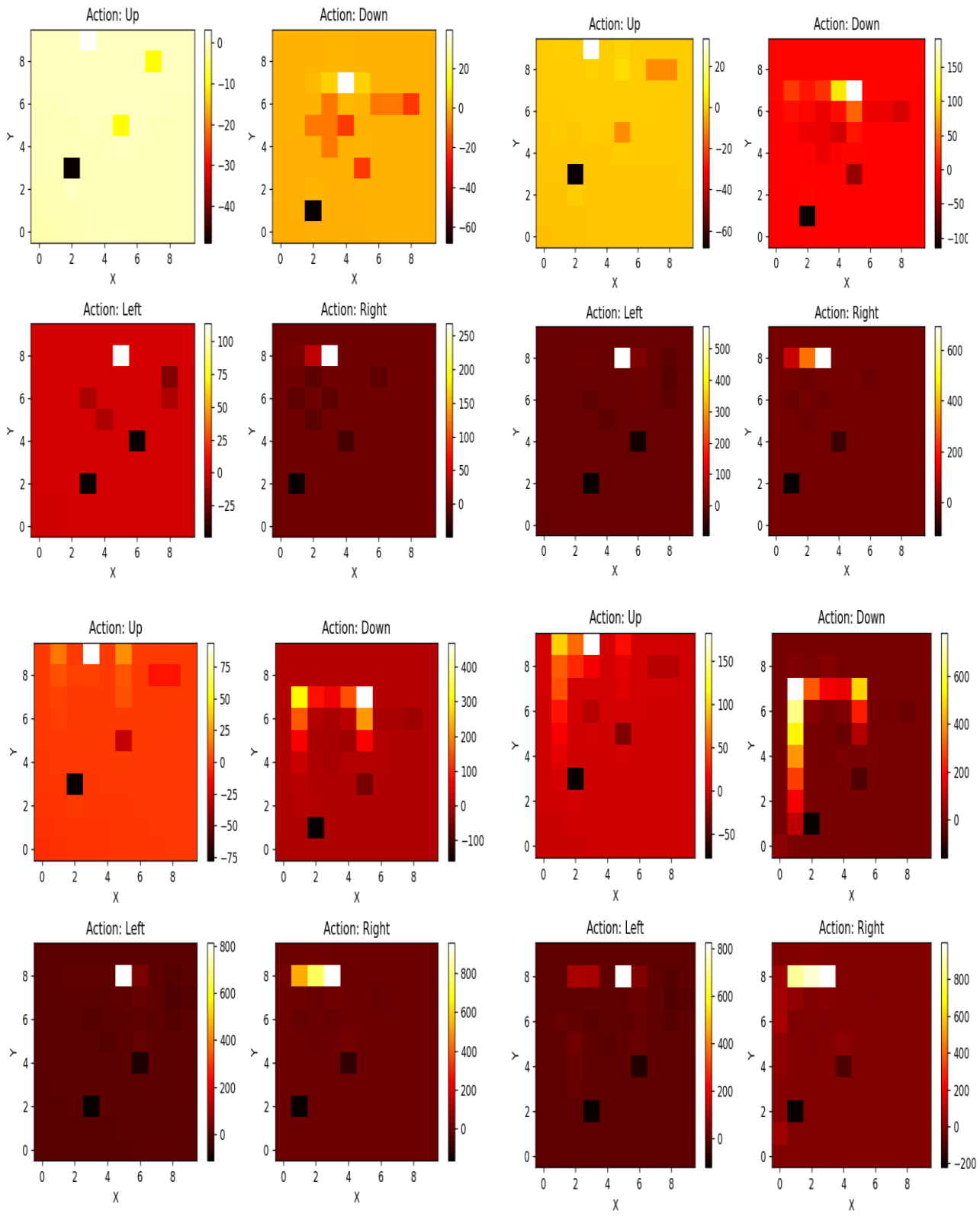
U prvoj iteraciji je agent imao više poraza (53), dok je imao 47 pobjeda, što je i očekivano s obzirom da nije posjedovao predznanje.

U svakoj od narednih iteracija je agent skupljao naučene podatke, te time postizao i bolje rezultate. Najznačajniji napredak vidimo između prvog i drugog ponavljanja, gdje se broj pobjeda skoro udvostručio, te nakon toga kroz naredne iteracije vidi se postepeni napredak, što je i očekivano.

U zadnjih par iteracija se procent pobjeda je dosegao minimum pobjeda od 96, dok je maksimum bio 99.



Slika 9. Grafički prikaz rezultata



Slike 10, 11, 12, 13. Heat map prikaz Q tabele, nakon I, III, VI i X ponavljanja

Heat mapa ima 4 podgrafa, za svaki od mogućih akcija (up, down, left, right).
Svaki podgraf pokazuje asocirane Q vrijednosti za specifičnu akciju. Redovi i kolone predstavljaju stanja u igrici. Intenzitet boje u svakoj od ćelija prikazuje opseg Q vrijednosti za dati par stanja i akcije.
Tamnije boje u ćelijama indiciraju manju Q vrijednosti, dok svjetlije boje indiciraju suprotno, tj. veće Q vrijednosti. Veća Q vrijednosit predstavlja bolji izbor akcije u specifičnom stanju. Na primjer, ako je najveća Q vrijednosti asocirana sa akcijom right (desno) u konkretnom stanju, to govori da je akcija right (desno) optimalna u tom slučaju.

7. Budući rad

Kada je riječ o budućem radu, iako je koncept igrice načelno jednostavan, postoji mnogo načina da se unaprijedi i učini boljom. Ono od čega se treba krenuti kada su u pitanju unaprijeđenja, jeste uvođenje novih algoritama učenja, poput deep Q learning, kako bismo postigli bolje performanse i još brže učenje. Nešto što dodatno može doprinijeti unaprijeđenju igre jesu i optimizacija parametara (epsilon, discount i learning faktor) kroz proučavanje najboljih kombinacija za agentovo učenje.
Kao još jedna važna karika u budućem radu jeste i proširenje funkcionalnosti igre, gdje se mogu dodati novi elementi i izazovi, kako bi AI morao donositi više odluka i više učiti. Tu se svakako može dodati više agenata, više zamki, kako bi se postigla veća složenost igre. Dodavanje više mapa je također jedna od ideja za implementaciju u budućem radu. Agent bi dobijao nasumično izabranu mapu, od na primjer njih 5, i onda bi znanje stečeno na jednoj od mapa primjenjivao i na ostalim, ostvarujući time optimalne rezultate.

8. Zaključak

Ovim radom se nastojalo pokazati kako se od jednostavne igrice može napraviti igrica u koju su uključene rapidno rastuće tehnike umjetne inteligencije i mašinskog učenja. To se postiglo dodavanjem Q algoritma, kojim se pokazuje da je moguće ostvariti veoma dobre rezultate, gdje

agent sam istražuje i na osnovu iskustva i istraženog, koristi to u pronalaženju rješenja iz raznih situacija. Osnovni cilj projekta je pokazati da se to može objasniti i na jednostavnim projektima, te kako svaka od ovih tehnika ima svrhu kako u malim tako i u velikim projektima. Rezultati govore da je agentovo učenje kroz svaku iteraciju sve uspješnije, tj. da iz svake situacije nauči nešto što će mu u idućem koraku pomoći. Procentualno, mnogo je veća razlika u pobjedama između na primjer prve i druge iteracije, nego između druge i treće iteracije, što je i očekivano, jer zbog dimenzije igre, agent u prve iteracije nauči mnogo više, čak većinu, što dovodi do toga da je stepen učenja manji u sljedećim iteracijama.

Agent u nekim slučajevima prije bira nasumični korak, nego već naučeni, kako bi istraživao nova polja i time dugoročno omogućio sebi lakšu pobjedu.

Kako je napisano u budućem radu, u konkretnom primjeru postoji još mnogobrojnih mogućnosti, kako bi se implementirale još naprednije stvari, čime bi agent dobio sasvim novu dimenziju učenja, uz naravno veći broj prepreka.

9. Reference

1. Artificial Intelligence, a modern approach – Fourth edition, Stuart Russell and Peter Norvig, April 28, 2020.
2. An introduction to Q-Learning: reinforcement learning – Freecode Camp
3. The Wumpus World in Articial Intelligence – javatpoint.com