

Nume: Cotet Teodor-Mihai

Grupa: 342C4

Pentru vizualizare interfata grafica se apeleaza programul cu python3 my_game.py –final_show

Sistemul de recompense:

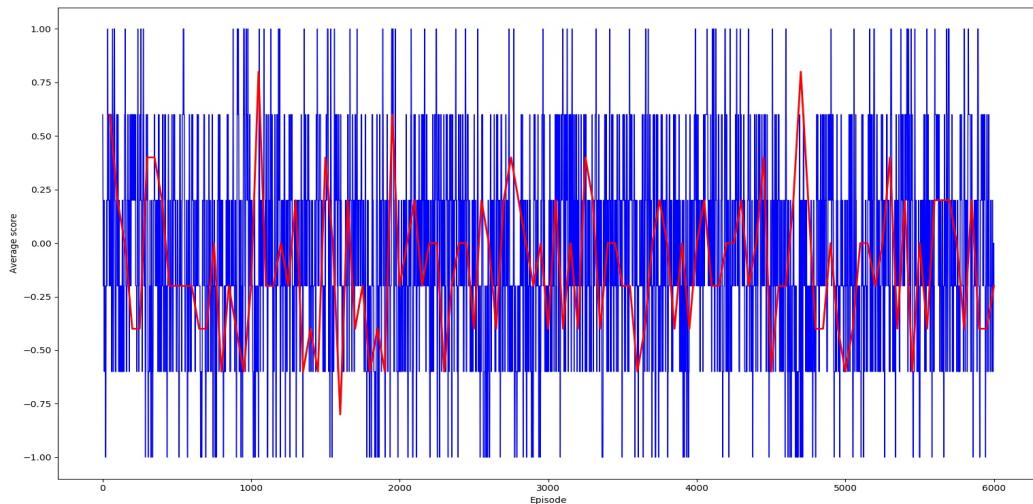
Dau 1 punct pentru un meci castigat, -1 pentru unul pierdut, in rest 0

Explicatii diverse rezultate:

- * toate graficele sunt trecute in foldere cu acelasi nume (random_random, greedy_random)
- * toate graficele sunt realizate pentru 30x12 dimensiunea tablei si 3 dimensiunea paletei, dupa 1800 de miscari se considera remiza (doar daca nu e specificat altfel)
- * toate graficele sunt pentru 6000 de episoade

1) random_vs_random: aici nu e mult de explicat, se observa din grafic ca nici un player nu domina

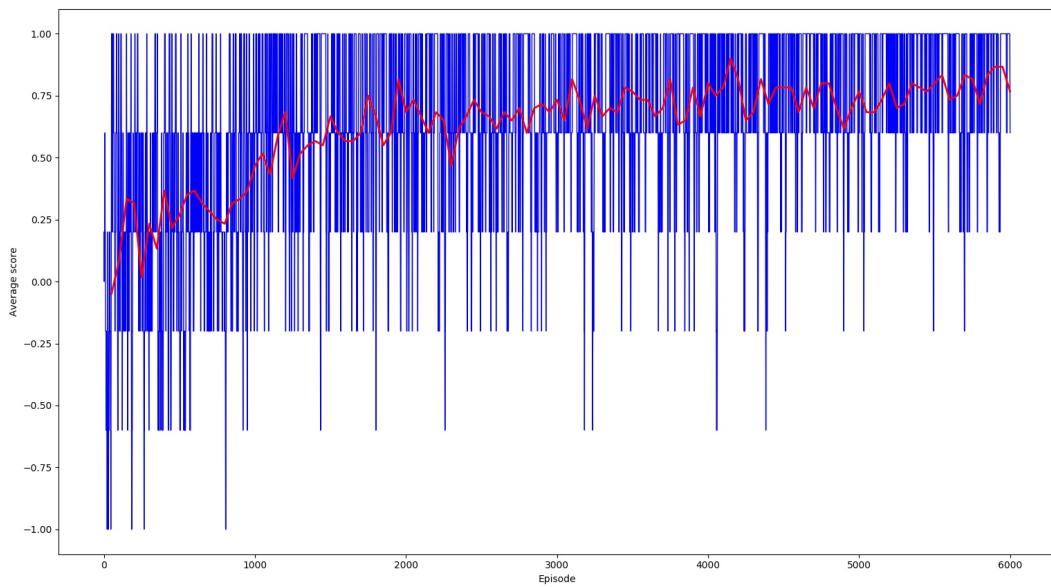
grafic:



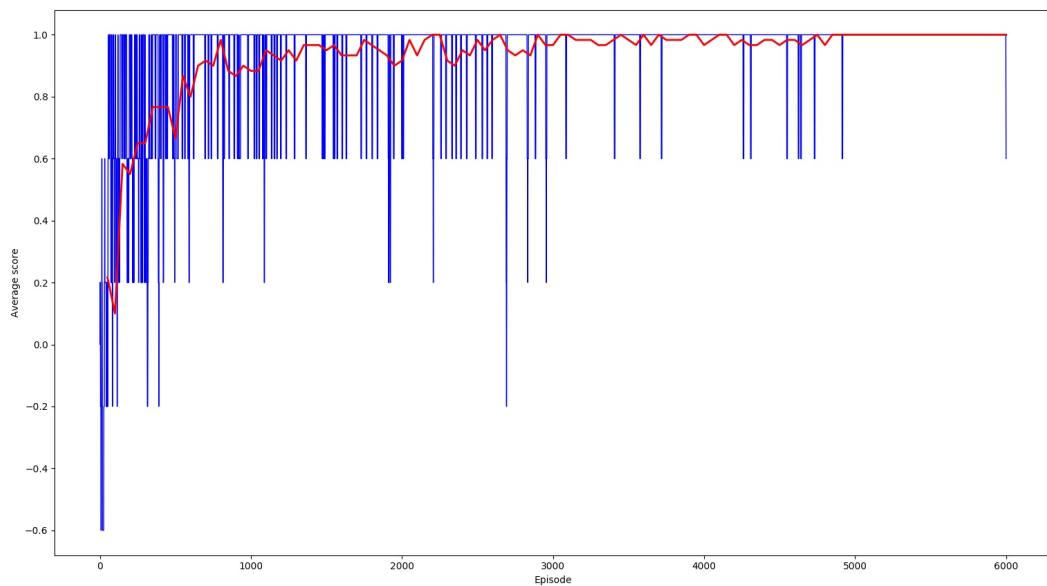
2) greedy_vs_random:

aceste tipuri de jocuri sunt impartite in 2 categorii:

- *) cand starea contine si pozitia paletei jucatorului advers:
aici m-am jucat cu diversi parametrii pentru learning_rate si discount
(0.004 cu 0.9999, 0.2 cu 0.91, 0.8 cu 0.999) dar se observa ca nu se poate trece de un 0.8 ca procentaj de castig chiar si dupa foarte multe meciuri (6000)
grafic:

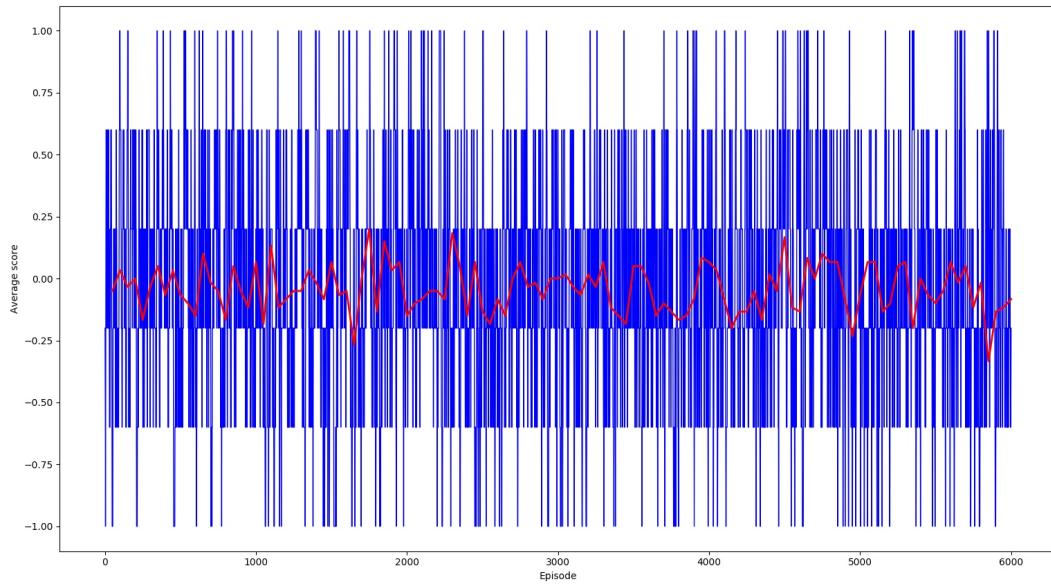


*) cand starea NU contine si pozitia paletei jucatorului advers:
 se observa o imbunatatire drastica (greedy_random_state_without_paddle2)
 dupa cateva jocuri procentajul de castig fiind 1.0, cu niste parametri pentru care
 daca as fi inclus si paleta 2 nu ar fi trecut de 0.8 - se retine ca se joaca mai bine
 daca starea nu contine si paleta adversarului
 grafic:



*) de curiozitate am incercat aici sa scot si directia mingii din starile retinute,
 rezultatele au fost cum era de asteptat, dezastruoase

(greedy_random_state_without_ball_velocity grafic), directia mingii fiind intr-adevar utila.
 grafic:

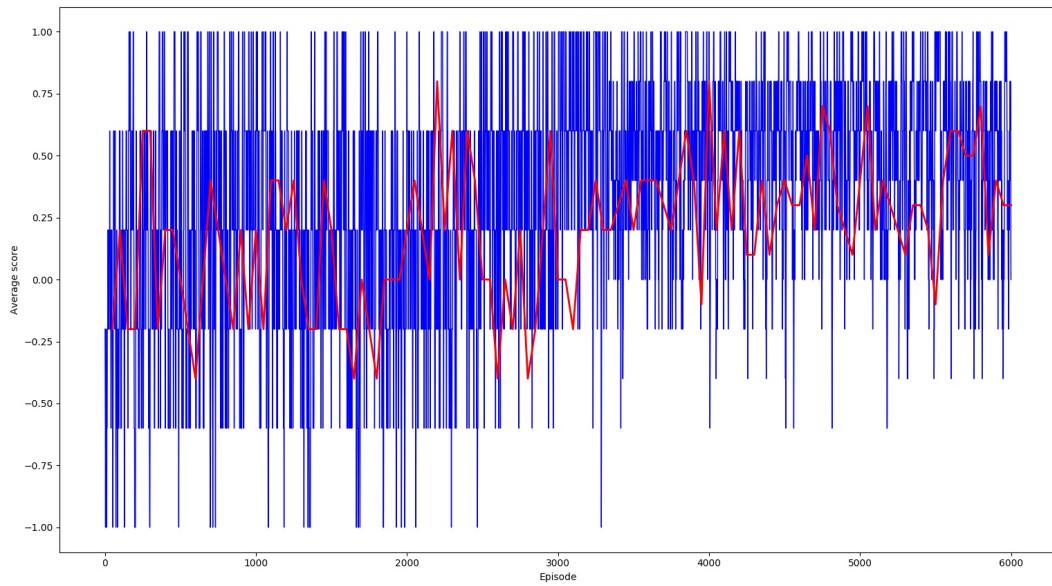


avand in vedere aceste observatii de acum incolo voi salva starile fara sa mai tin cont de pozitia playerului 2

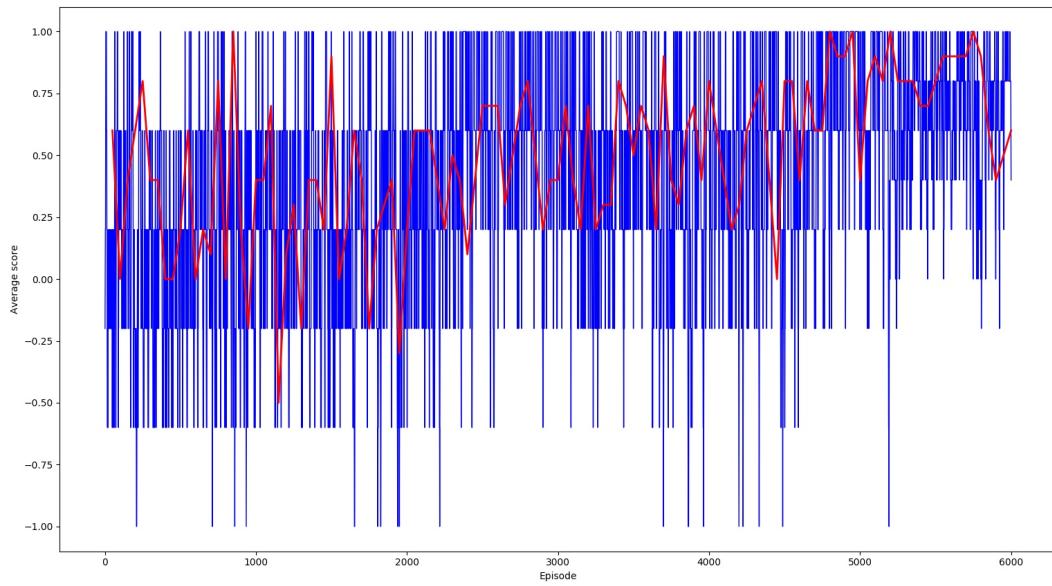
3) greedy_vs_greedy:

- * pentru player2 folosesc valorile mesei oglindide din Q atat pentru invatare cat si pentru testare aici ne-am astepta sa avem multe remize (ambii jucatori avand acelasi informatii), insa situatie e un pic diferita
- * in general playerul1 are castig de cauza pentru ca el invata fix pentru starile la care este el expus playerul2 trebuind sa aiba un grad de noroc pentru a da peste stari care au fost bine explorate de player1
 (practic asta se intampla pentru ca jocul nu este "perfect simetric")
- * am facut 2 grafice cu aceeasi parametri, doar ca intr-unul meciurile se termina remiza dupa 6000 miscari, in altul dupa 600. player1 fiind foarte putin inainte joaca mai bine in situatia cu remize la 6000 intrucat are timp sa-si puna in valoarea acel mic avantaj

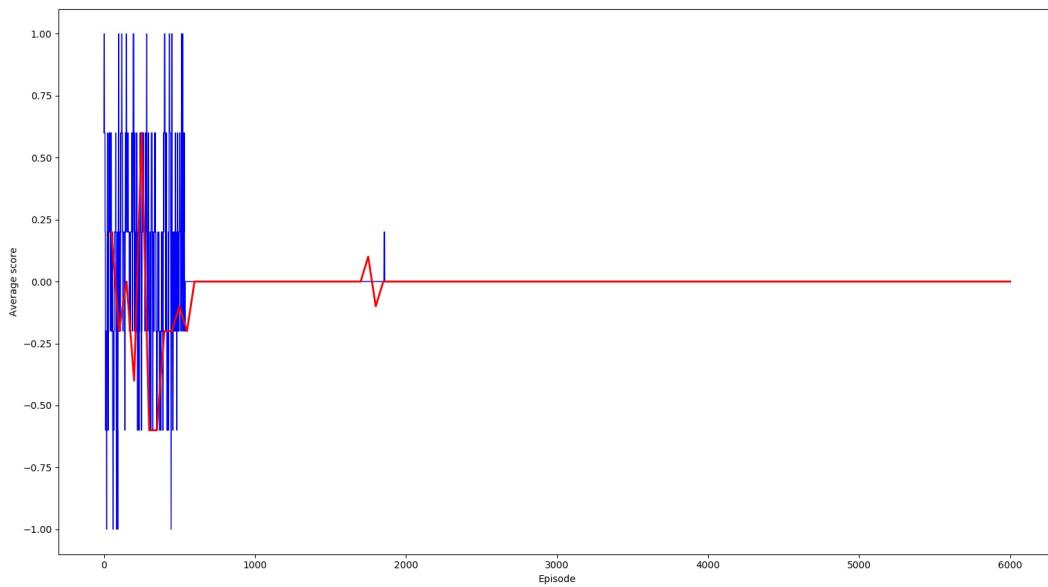
grafic-draw600:



grafic-draw6000:



* de observat si ca disputele variaza in functie de lungimea mesei: o lungime de 31

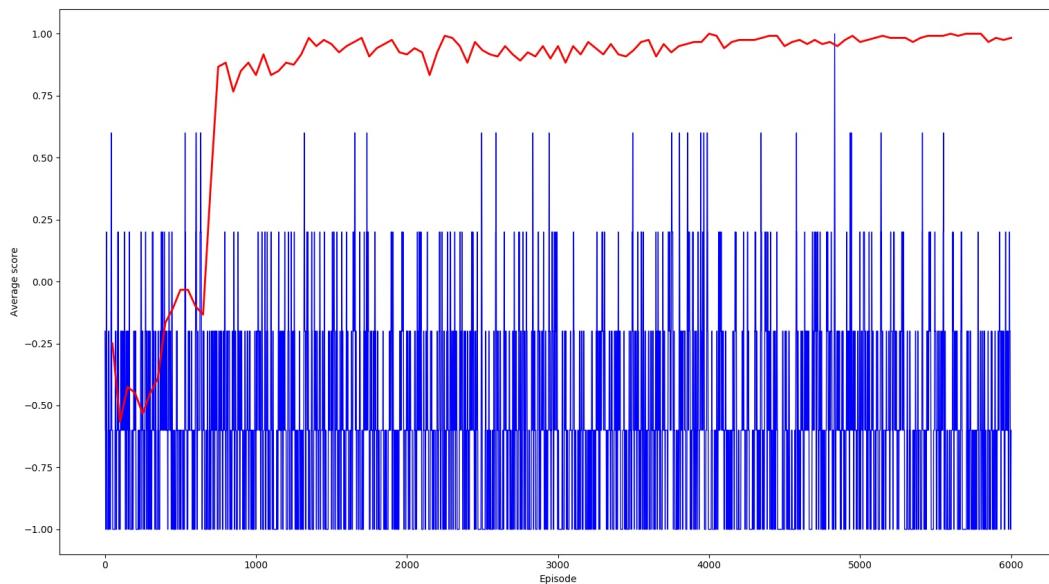


favorizand meciuri echilibrate (probabil pentru ca atunci cei 2 playeri intalnesc sitatii mai asemanatoare - greedy_greedy_31_12_3 grafic)

4) greedy_vs_almost_perfect:

- * am implementat adversarul perfect asa cum este descris in enuntul temei
- * cum un adversar perfect nu era interesant, l-am facut sa raspunda random cu o prob de 0.1
- * in general am testat cu mai multi factori de learning rate, dar graficele arata similar, agentul greedy reusind sa ajung la o rata de castig suficient de buna (dar nu perfecta) cu niste parametri normali (learn_rate=0.1, disc=0.9999)

grafic:

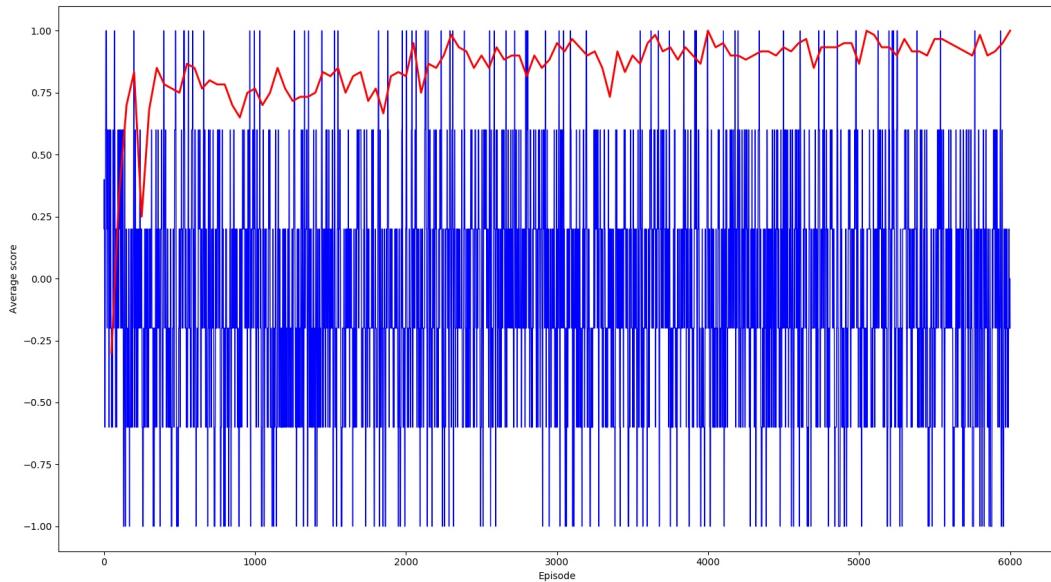


5) e-greedy_vs_random:

- * pentru niste parametrii clasici (learn_rate=0.1, disc=0.99 , eps=0.05)

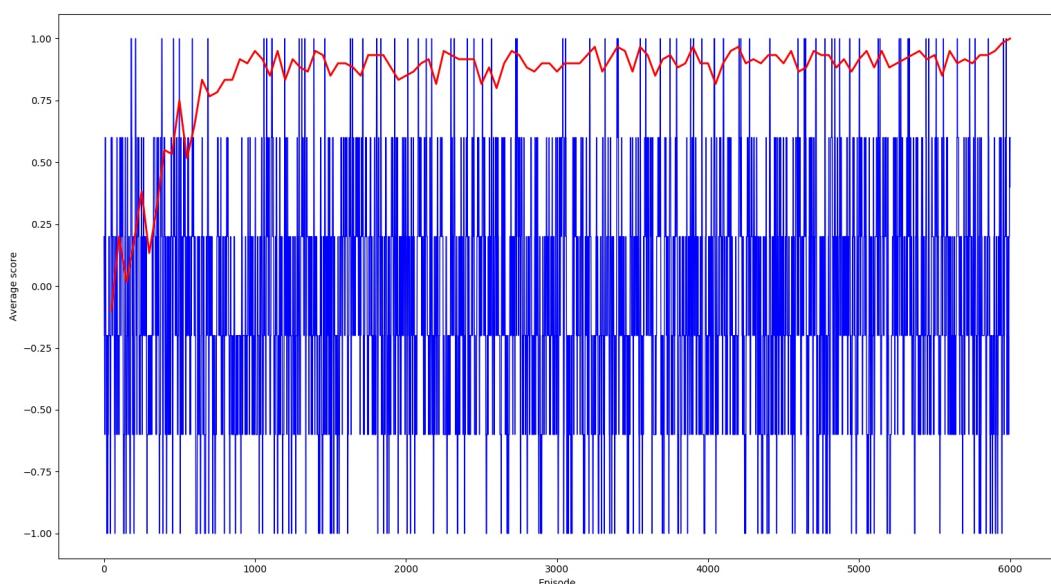
e_greedy_random_lr01_disc099_eps005 observam ca se ajunge la o rata de castig foarte buna (aproape de 1) dupa foarte putine episoade (sub 500), dar in acelasi timp observam ca nu atinge rata de castig absoluta (1.0) nici dupa 6000 de meciuri

grafic: e_greedy_random_lr01_disc099_eps005



- * daca marim eps la 0.2 observam ca tot nu se atinge o rata de castig cu multa mai buna spre final, dar observam ca initial se invata mai greu - panta e mai putin abrupta, fapt care are sens pentru ca exploram mai mult actiuni care probabil nu sunt bune

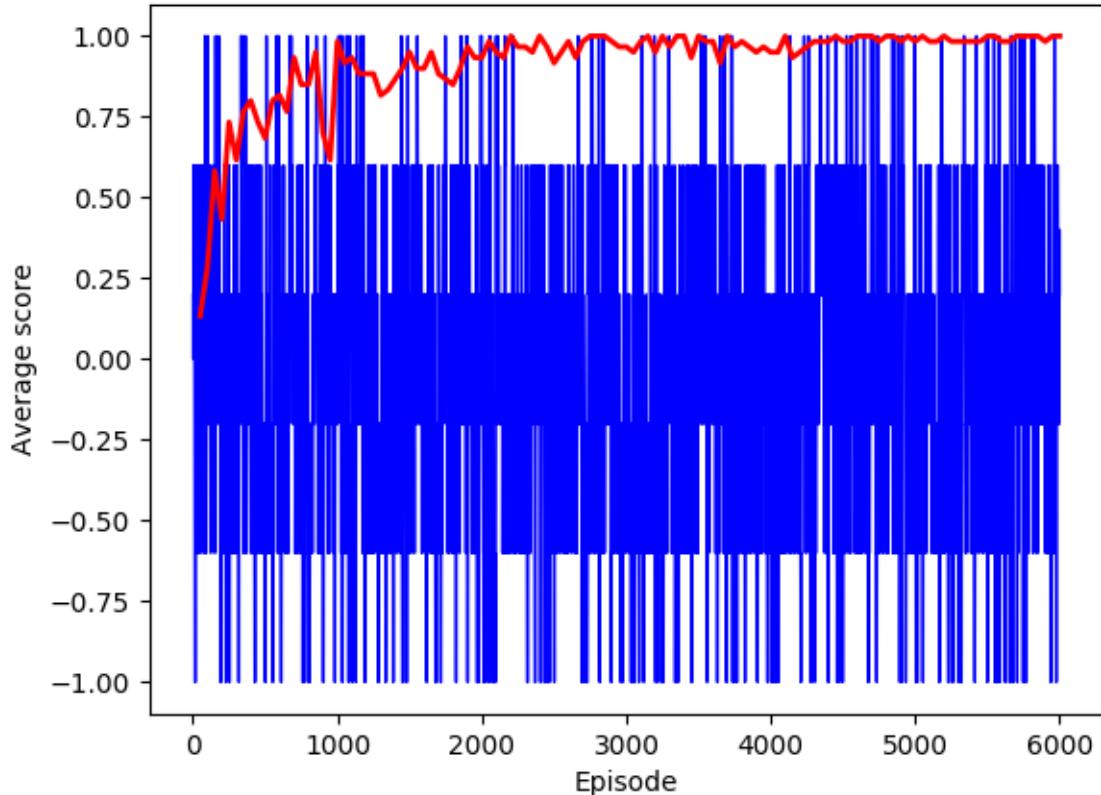
grafic: e_greedy_random_lr01_disc099_eps02



- * pastram eps=0.2 dar marim learn_rateoul la 0.2 si discountul la 0.999999, astfel obtinem

valori foarte apropiate de 1.0 dupa mai multe episoade , fapt care are sens pentru ca gandim pe termen mai lung (discount factor mare) si incercam sa fim mai sensibili la rezultate

grafic: e_greedy_random_lr02_disc099999_eps02

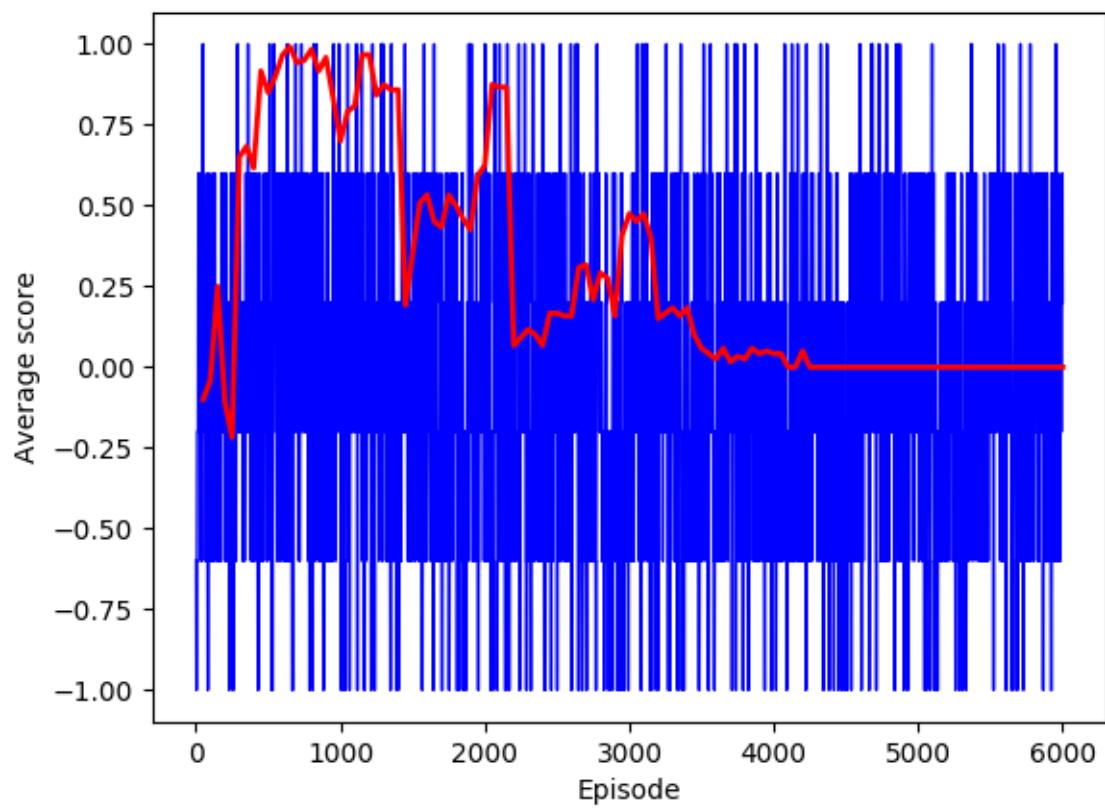


retinem aceste rezultate pentru urmatorul adversar al lui e_greedy

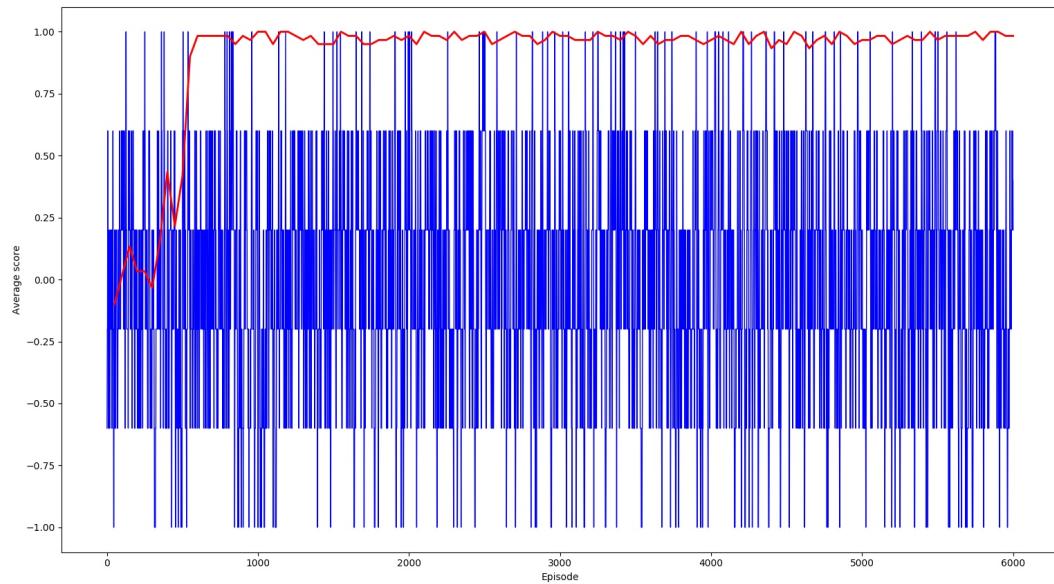
6) e-greedy_vs_greedy

* observam ca situatie este asemanatoare cu greedy_vs_greedy intrucat amandoi jucatori au acces la aceeași informație doar pentru unul informație este ceva mai utilă decât pentru celălalt. Pentru o tablă de 30x12 observăm că după ce explorăm foarte multe stări amandoi joacă la fel de bine (explorăm și stările de care are nevoie player2, dar doar după multe episoade, de astă player1 are initial câștig de cauză)

grafice: e_greedy_greedy_eps=02_30_12

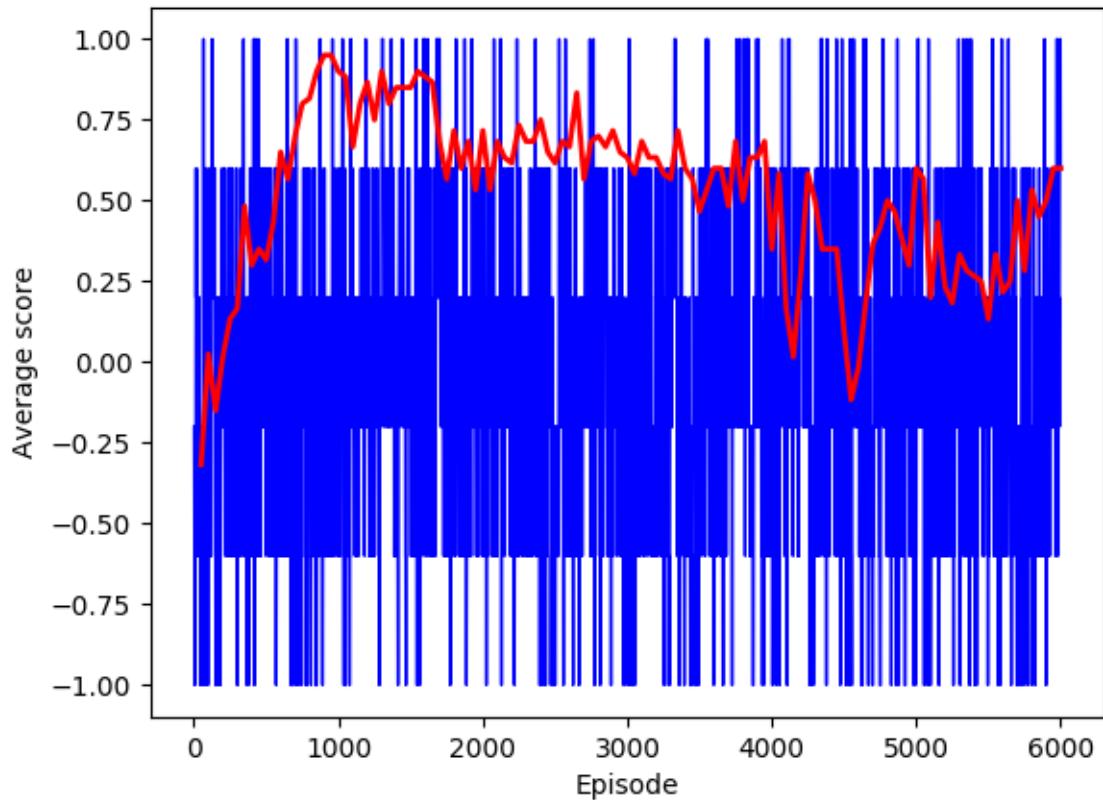


* pentru o tabla de 50*37 (cu multe stari) observam ca nu exista destule episoade si pentru player2 sa aiba stari utile explorate asa ca acesta pierde mai mereu:
grafic: e_greedy_greedy_eps=02_50_37



* pentru o tabla ceva mai mica: observam ca fiind mai putine stari player2 se descurca ceva mai bine

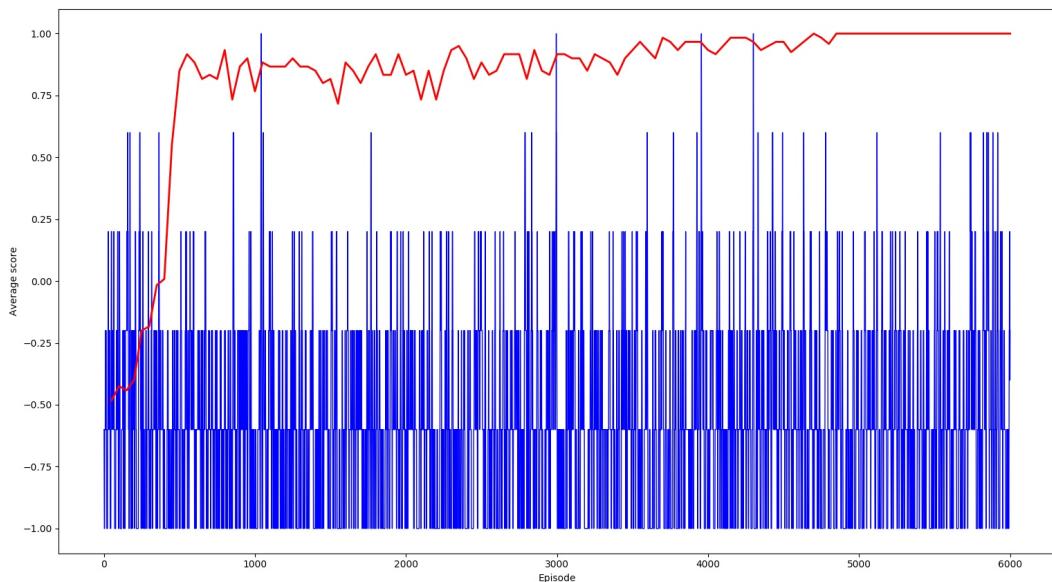
grafic: e_greedy_greedy_40_17



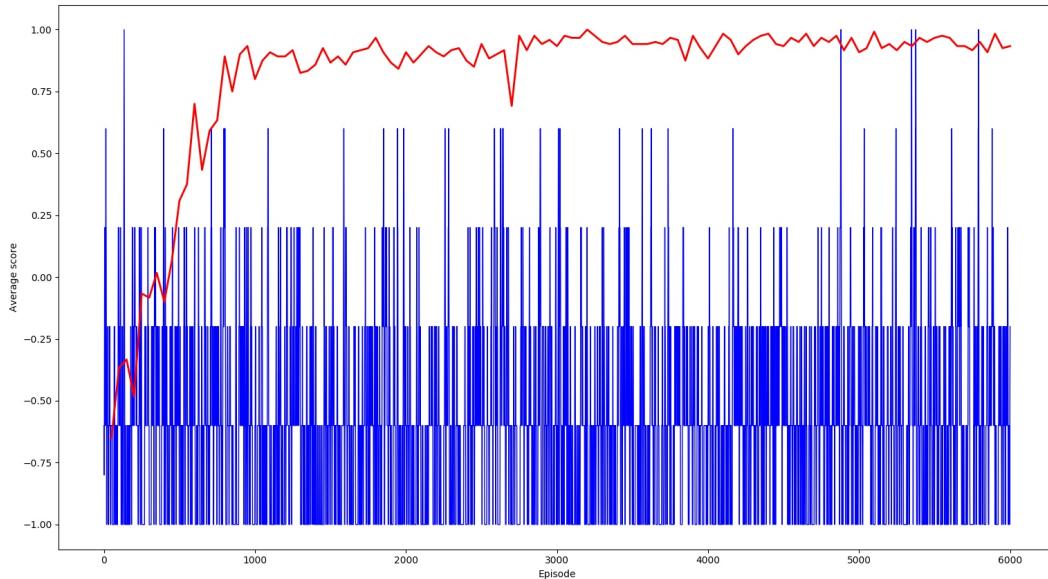
* epsilon nu influenteaza prea mult jocurile, l-am lasat 0.2

7) e-greedy_vs_perfect:

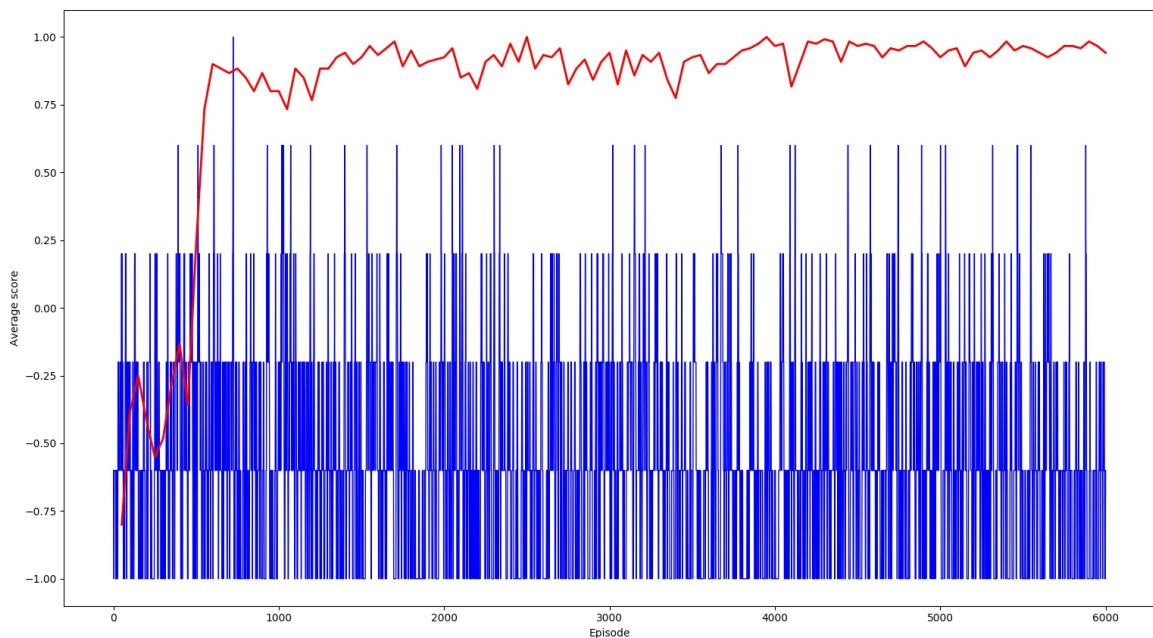
* comparam initial e_greedy cu greedy intrucat sunt foarte asemanatoare, vrem sa obtinem o rata de castig aproape de 1 incercand aceeasi parametru de la greedy, dar cu un epsilon de 0.2 observam ca nu se imbunataste cu mult rata de castig finala



* schimband insa discountul la 0.9999999 fata de de 0.999 se observa ca se ajunge la o rata de castig de 1 spre final lucru explicabil deoarece un discount mai mare permite ajustari mai fine si este exact de ce avem nevoie
grafic: e_greedy_perfect_lr=02_disc=099999_eps=02



* pentru obtinerea pante de invatare mai bune folosim urmatorii parametrii: learn_rate=0.4, disc=0.99, eps=0.04 , dar aceasta invatare mai buna vine cu un trade off, si anume faptul ca chiar si dupa multe episoade nu avem rata de castig egala cu 1
grafic: e_greedy_perfect_lr=04_disc=099_eps=006



Bonus:

Pentru bonus am facut urmatoarea clusterizare.

Clusterizezi starile dupa pozitia mingii astfel: impart table in split sectiuni (cu cat e mai mare table cu atat e mai mare numarul de sectiuni). Pentru fiecare sectiune clusterizezi starile care au pozitii similiare ale mingii in marja de eroare split. Cu cat minge este mai aproape de a lovi paleta cu atat marja de eroare este mica (cand este foarte aproape nu clusterizez deloc).

In e-greedy vs perfect, cu 3 sectiuni agentul se comporta decent, desi nu atinge rate de castig apropriate de 1.

