Aventurile domnului H(Hero/Gigel)

Stefu Teodor 341C1

Descriere implementare:

Implementarea consta in 3 module python, unul de generare harti, unul de abstractizare a jocului, iar al treilea de implementare algoritm si grafice.

Harta este mentinuta in intr-o matrice rooms x dim1 x dim2, in care fac toate update-urile pe baza acitiunilor, il mut pe H si/sau G. Pentru a nu eficientiza cautarea de portale, pentru fiecare intrare (entry, out) a portalelor, am adaugat si o intrare (out, entry), in cazul in care H doreste sa se intoarca in camera anterioara.

Am folosit metoda din laborator de serializare a starilor in string-uri, pentru a putea mapa starile in functie de raza. Astfel mereu tin in starea curenta cat vede domnul H. Pe langa asta mereu tin pozitia absoluta a lui H, si pe baza ei generez starea in care se afla. (acest lucru il fac pentru a putea avea cazul in care starile se pot confunda).

Algoritmul de miscare al gardianului este cel din laborator, care se foloseste de distanta si directie pentru a il ajunge pe H.

Generarea tabelei:

Pentru generare am pus cateva restrictii, deoarece m-am apucat tarziu de tema si am considerat ca este mai importanta analiza algoritmului.

- Peretii sunt de cate o patratica
- Numarul de portale este mai mare sau egal cu numarul de camere si ma asigur ca toate camerele fac parte dintr-un graf cu o singura componenta conexa.
- Peretii nu au alte elemente in jur initial => deci sigur exista drum de H -> F(stare finala)
- Numarul de gardieni este maxim numarul camerelor(din enunt)

Pe baza acestor restrictii imi generez harti.

Ca si reward-uri am ales:
TREASURE_REWARD = 100
FINISH_REWARD = 30
STEP_REWARD = -1

*Portalele au tot -1 reward
Comanda de rulare completa(similar laboratorului):
python q_learning.py --map_file='test_test.txt' --learning_rate=0.1 --discount=0.99 --epsilon=0.05 --train_episodes=10000 --eval_every=1000 --eval_episodes=100 --max_moves=1000 --final_show --sleep_time=0.2 --radius=2 --plot

Descrierea hartii pe care am facut graficele.

Camere de dimensiune 12 x 12.

3

_,_,_,_,_,_,_,_,T,_,_,_
_,_,_,_,_,_,_,_,P,T,_,_ (0, 1, 8) (1, 2, 2)
_,_,T,_,#,_,_,_,T,_,_,_ (1, 3, 9) (2, 6, 11)
_,#,_,_,_,T,#,_,_,_,_,_ (0, 8, 4) (1, 8, 4)
_,_,_,_,#,_,_,_,T,_,_,T (0, 5, 11) (1, 3, 4)
_,#,_,_,_,_,_,_,F,_,_,P
_,T,_,_,#,_,_,#,_,_,_,_
_,T,#,_,_,_,T,_,T,_,T,_
T,_,_,_,P,_,T,_,#,_,G,T
_,_,_,_,_,_,_,_,_,T,T,_
_,#,_,_,_,#,_,_,_,_,#,_
T,_,_,_,_,_,T,_,T,_,_,_

_,_,_,_,_,_,_,_,_,_,T,T
_,_,_,_,_,_,T,#,_,_,_,_
_,_,P,_,_,_,T,T,_,_,_,_
_,_,_,_,P,_,_,_,_,P,_,_
_,_,_,T,T,_,_,_,_,T,_,T
_,_,_,_,_,_,_,#,_,_,#,_
_,_,#,_,_,#,_,_,_,_,_,_
_,T,_,H,_,_,_,#,_,#,_,_
_,_,T,T,P,T,_,_,_,_,_,_
T,_,_,_,_,_,_,_,_,_,#,_
T,#,_,G,_,#,_,_,#,_,_,_
T,_,_,_,_,_,T,_,_,_,_,_

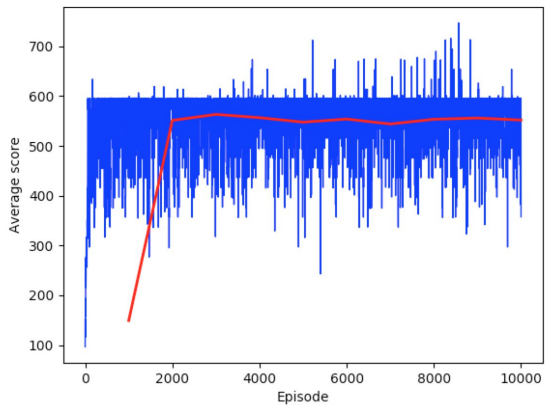
_,_,_,_,_,_,_,_,_,_,_,_
_,#,_,_,_,_,_,_,#,_,#,_,_
_,_,_,#,_,#,_,_,_,_,_,_
_,_,_,_,T,_,_,_,#,_,#,_
T,_,_,#,_,_,_,_,_,_,_,_
_,_,_,_,_,_,_,#,_,#,_,_
_,T,#,_,_,_,_,_,_,T,T,P
_,_,_,_,_,_,_,_,T,T,_,_
_,T,_,#,T,#,T,#,_,_,_,_
_,#,_,T,_,_,_,_,_,#,_,_
_,T,_,T,#,_,_,_,_,_,_,_

_’_’_’_’_’_’_’_’_’_’_’_’_

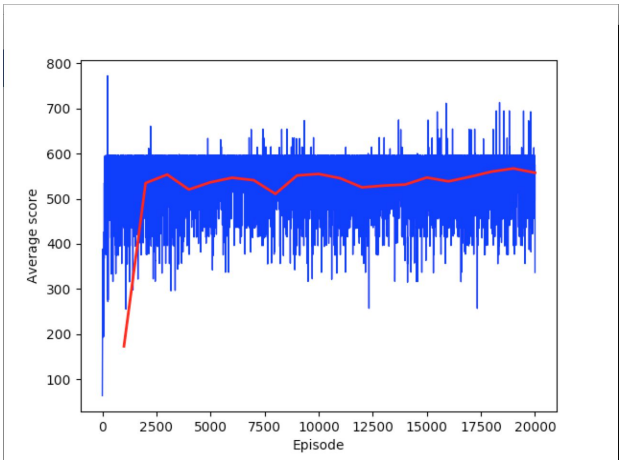
Numar de episoade:

Parametri fixi: learning rate, discount, epsilon_greedy
Politica: greedy_epsilon
python q_learning.py --map_file='test_test.txt' --learning_rate=0.1 --discount=0.99 --train_episodes=10000 --eval_every=1000 --eval_episodes=100 --max_moves=1000

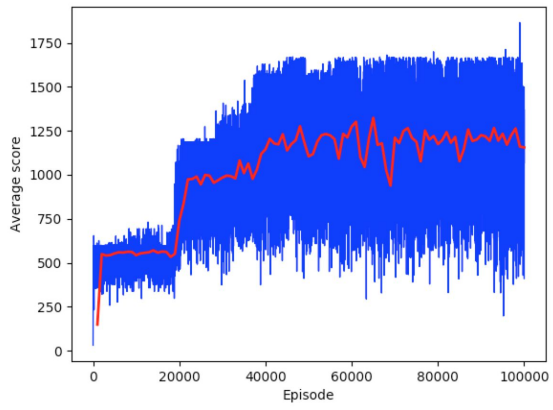
10.000



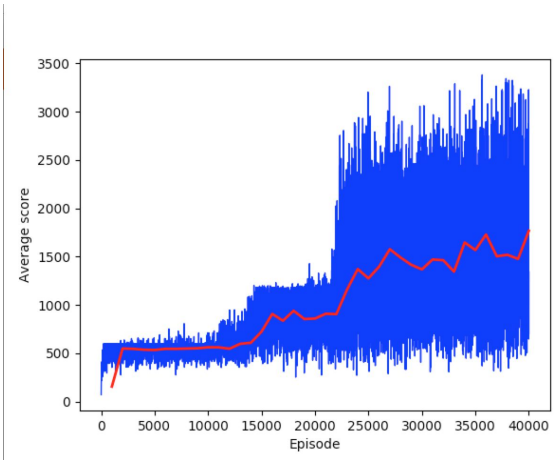
20.000



40.000



100.000



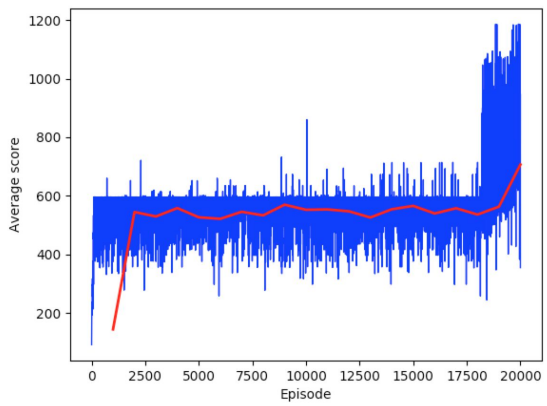
Cu cat numarul de episoade este mai mare, jocul ajunge la un scor din ce in mai bun. Acest lucru este favorizat de faptul ca agentul reuseste sa exploreze cat mai multe stari. Pentru un numar extrem de mare de episoade, agentul poate ajunge sa cunoasca toata harta.

Rata de invatare

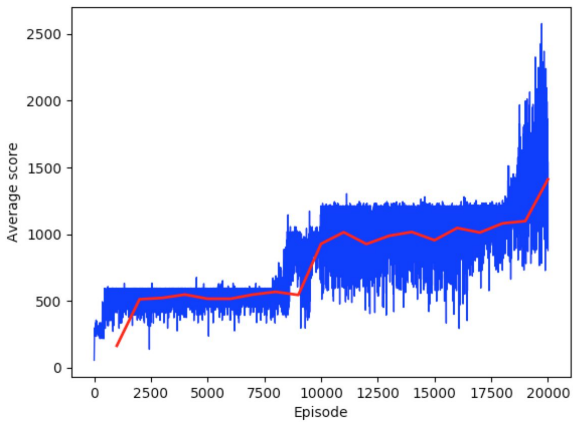
Parametri fixi: discount, epsilon_greedy, numar episoade
Politica: greedy_epsilon

python q_learning.py --map_file='test_test.txt' --learning_rate=0.1 --discount=0.99 --epsilon=0.05 --train_episodes=20000 --eval_every=1000 --eval_episodes=100 --max_moves=1000

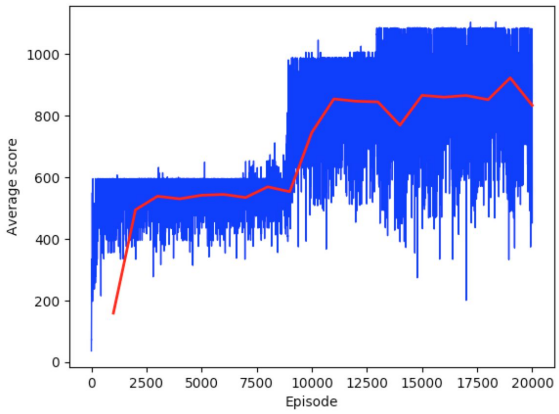
0.1



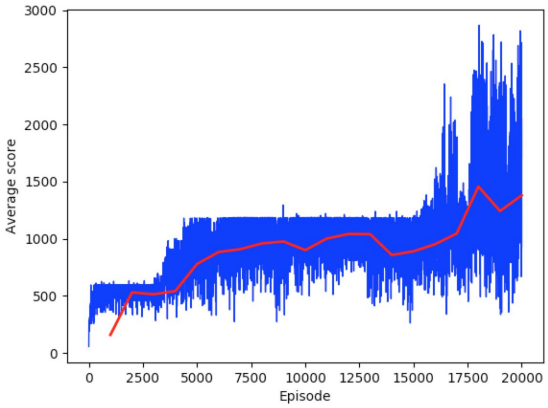
0.3



0.5



0.7

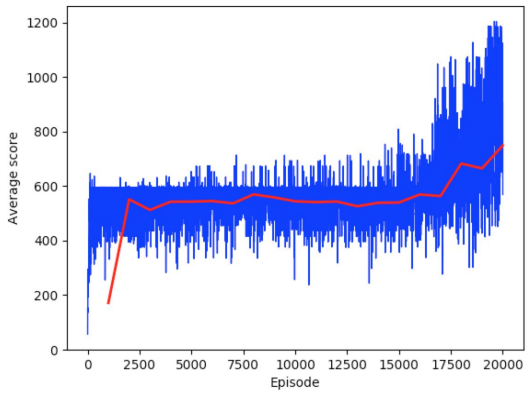


Din grafice se observa un trade-off de stabilitate a solutiilor gasite dar cu cresteri lente ale solutiei pe parcursul episoadelor(learning rate mic), fata de o crestere brusca dar mai putin stabila.(learning rate mare).

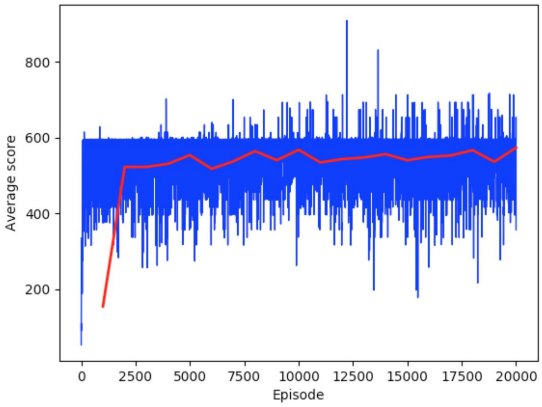
Discount:

Parametri fixi: learning rate, epsilon_greedy, numar episoade
Politica: greedy_epsilon

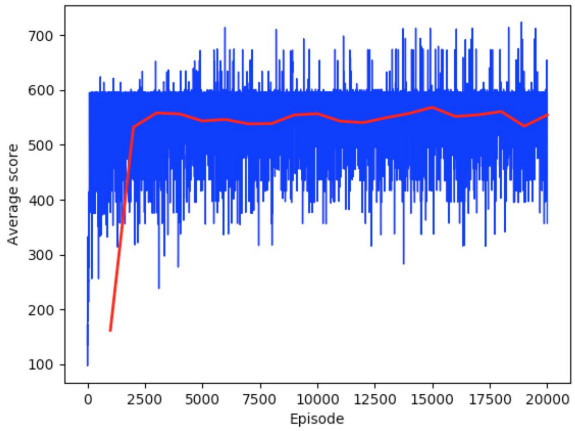
0.99



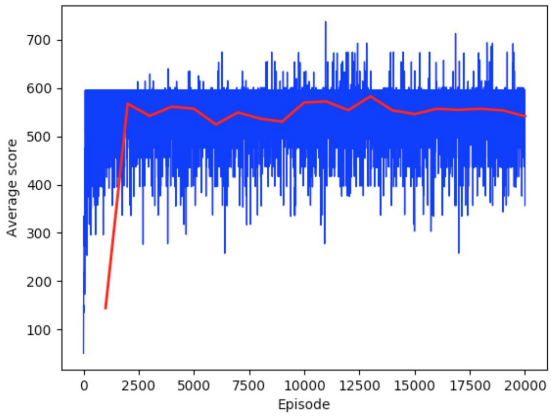
0.8



0.5



0.3

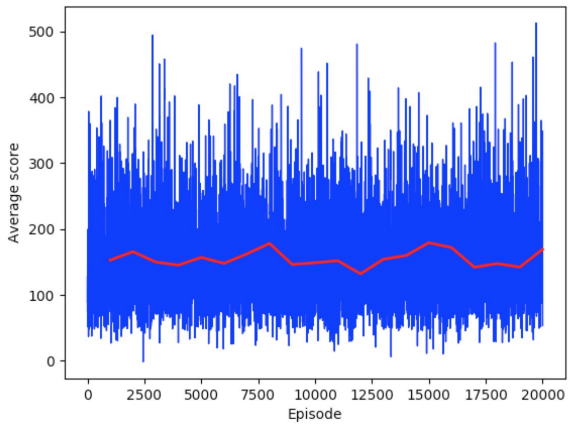


Cu cat scade discountul, se observa ca si valorile de antrenare nu mai au aceleasi (spike-uri), negativa sau pozitive. Desi valorile evaluate au dat cam aceleasi valori, se observa cum valorile de antrenarea si cele de evaluare se aproprie unele de altele.

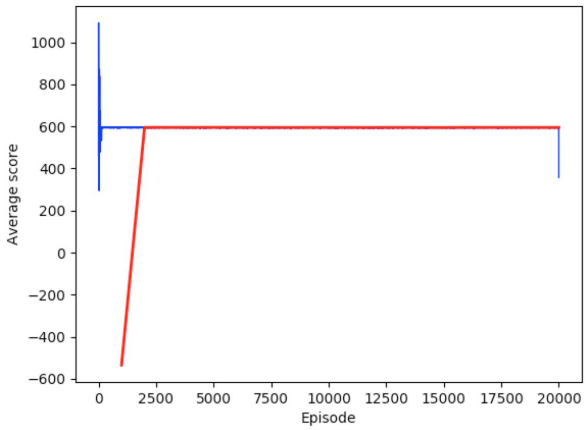
Politici:

Parametri fixi: discount, learning rate, epsilon_greedy, numar episoade
python q_learning.py --map_file='test_test.txt' --learning_rate=0.1 --discount=0.99 --epsilon=0.25 --train_episodes=20000 --eval_every=1000 --eval_episodes=100 --max_moves=1000

Random

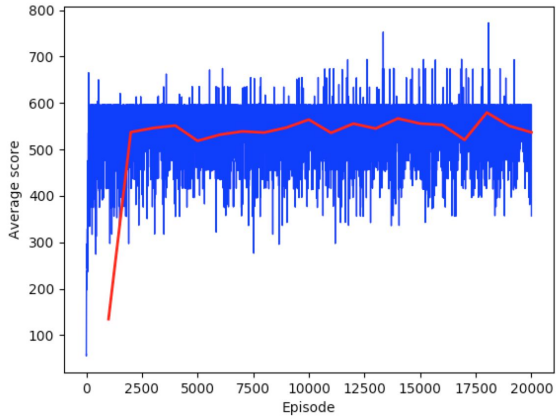


Max First

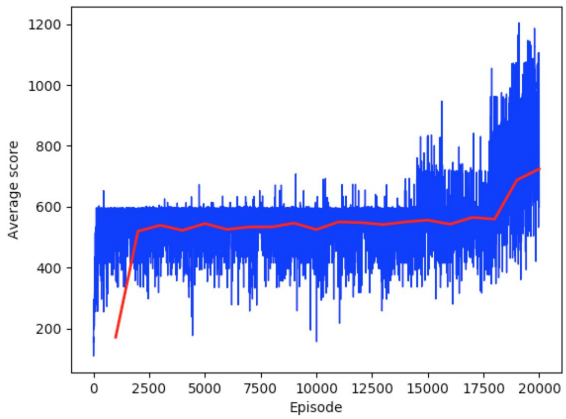


Epsilon greedy:

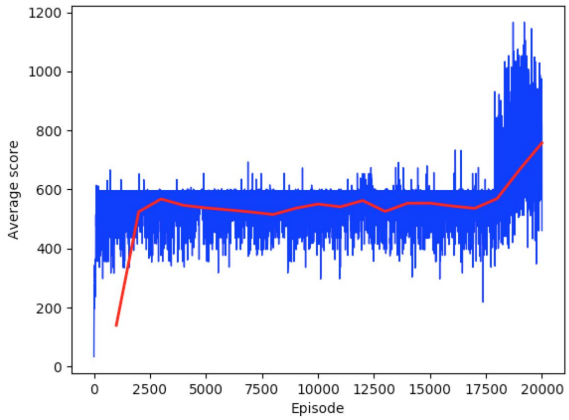
0.05



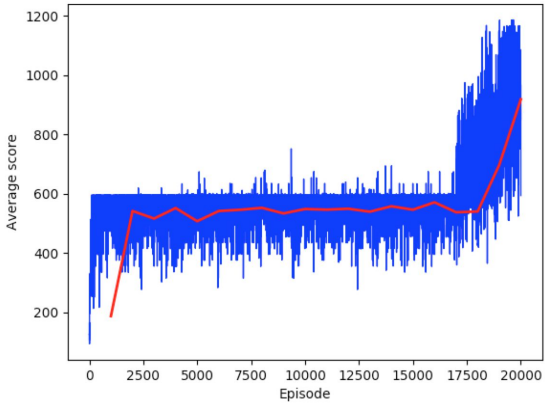
0.15



0.25



0.5



Dintre cele 3 stategii cele mai slabe rezultate sunt pentru politca de random.

Max First gaseste un maxim local si il exploateaza la nesfarsit pe acesta. Astfel mereu agentul o sa aplice actiunile care il duc spre cel mai mare punctaj, bazat pe reward-ul imediat, in limita mutarilor permise per joc.
Cele mai bune rezultate au fost obtinute de epsilon greedy deoarece face o un raport in exploatarea unor resurse si explorarea hartii (altor posibile solutii). Urmareste mai mult reward-ul indepartat, fata de cel apropiat.

Limitari ale algoritmului:

Pe langa parametrii de ajustare al algoritmului Q-Learning (discount, learning_rate si epsilon) un factor foarte important care influenteaza rezultatul final este configuratia jocului.

Parametrii de configurare: numarul de pereti, numarul de gardieni, numarul de comori, numarul de portale, raza jocului.

Raportul camere / nr_episoade

Cu cat numarul camerelor creste, numarul starilor creste expornential, avand in medie 3-4 posibilitati de actiuni viitoare in fiecare stare. Prim urmare si numarul episoadelor creste foarte mult.

In limita timpului disponibil pentru invatare, impus de situatia in care este folosit algoritmul, el il invata pe domnul H (Hero/Gigel) ce actiuni sa ia in mediul jocului.

Daca se pastreaza acelasi raport camere/nr_episoade ar trebui ca algoritmul sa se stabilizeze mereu.

Asezarea elementelor pe harta care pot crea probleme(pattern-uri):

Un caz care nefericit este cand in starea initiala gardianul este in raza agentului, si invers. Acest lucru il forteaza pe agent sa gaseasca un portal cat mai repede, lasand in urma comorile din camera de start. Sansele sa mai ia acele comori sunt in cazul unui alt portal catre camera initiala si un numar foarte mare de episoade.

Un alt exemplu nefericit in cazul unei politic MaxFirst si o zona cu multe comori printre care si starea finala, atunci el va lua doar comorile din drumul spre starea finala, nu va face un ocol si pentru celelalte comori. Acest lucru este influentat si de diferenta reward-urilor dintre comori si starea finala.

De asemenea hartile aglomerate sunt mai potrivite pentru aceasta problema, fata de hartile cu elemente putine si rare. Hartile aglomerate duc la o distinctie clara a starilor, si astfel

nu se creaza multe confunzii, dar in cazul hartilor cu elemente putin, se creaza confuzii deoarace o zona goala dintr-o harta se poate mapa pefect pe o zona goala din alta harta. In acest caz se modifica Q[state, *] cand nu ar fi cazul. Astfel se poate ajunge sa se aleaga o alta actiune fata de cea fireasca la urmatoarele jocuri.

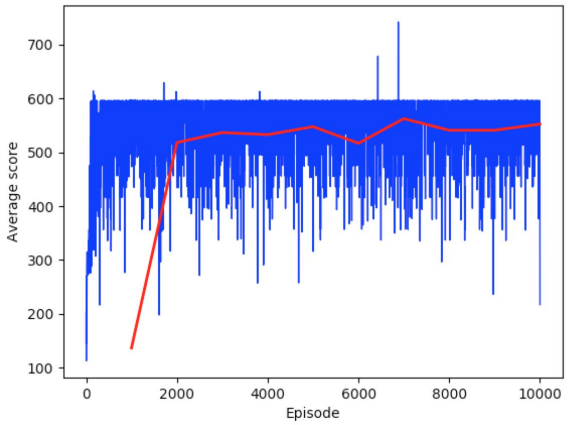
Raza:

Lungimea razei poate duce la confuzia starilor. Cazul razelor mici poate sa fie echivalent hartile neaglomerate, iar razele lungi cu hartile aglomerate. Cu cat raza este mai lunga, cu atat mai multe elemente vor intra in starea curenta, si altfel se pot distinge starile mai usor. In urma confuziei de stari apar consecintele prezentate la subpunctul anterior.

Influenta razei:

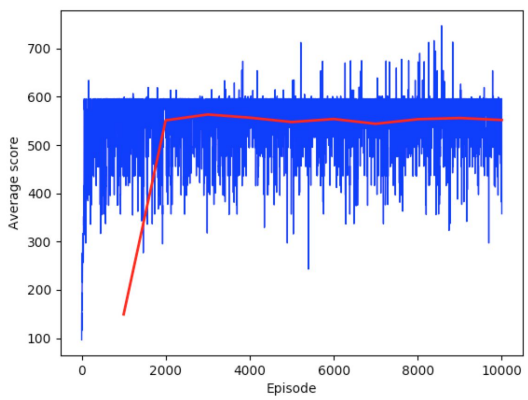
Pentru camera de 12x12.

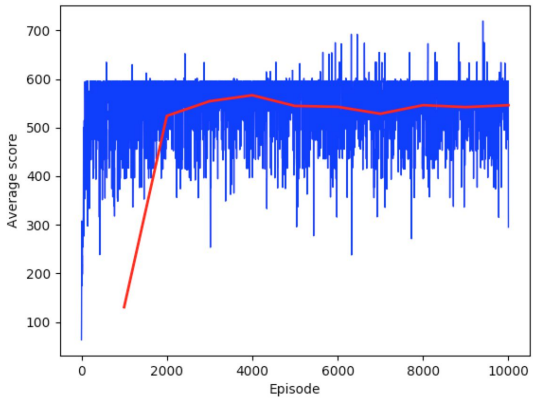
Raza 2



Raza 4

Raza 8





Cu cat raza este mai mare rezultatele sunt mai bune, ajunge mai repde la starea de stabilitate, si starea de stabilitate are spike-uri pozitive.

Gasirea solutiei:

Gasirea starii finale nu este scopul central al agoritmului, ci maximizarea reward-ului. Din acest motiv gasirea solutie nu este obligatorie.

Cazuri in care nu poate sa gaseasca starea finala:

- Raportul reward-urilor dintre comoara si starea finala nu este foarte mare. In acest caz, agentul se poate deplasa catre o zona cu foarte multe comori de unde ar avea un castig mai mare fata de directia catre starea finala.

- Politica de explorare: Daca politica este MaxFirst, el o sa porneasca spre “maxim local” care se poate sa nu fie si “maximul global”. Mereu va exploata prima solutie gasita, din cauza ca mereu alege cel actiunea care ii da cel mai bun reward in acea stare, lucru care reduce de fapt puterea de exploatare a agentului. In cazul strategiei randon, este o sanse extrem de mica sa gaseasca starea finala. In cazul politicii eps-greedy care permite explorarea de noi posibilitati pentru o anumita stare, pentru un numar foarte mare de episoade si mutari per joc ar putea sa gaseasca solutia, dar la fel daca este mai retabil ca punctaj final sa ajung in starea finala sau nu.