

Tema 1 ML

Nume : Sava
Prenume: Dragos
Grupa: 342C4

Cuprins

- I.Introducere
- II. Reprezentarea starilor
- III. Implementare
- IV.Obesrvatii
- V. Grafice
- VI. Comparatie SARSA vs Q-Learning
- VII.Mod utilizare
- VIII.Arhiva

I.Introducere

- Implementarea temei a fost realizata in limbajul JAVA.
- Am decis sa fac implementarea de la zero , fara a folosi serverul de joc din arhiva.
- M-am inspirat/ajutat un pic din codul din server atunci cand am implementat logica jocului(clasele Brick,Board, etc.).

II.Reprezentarea starilor

Am ales sa reprezint starile si actiunile sub forma unor siruri de caractere.

O stare este alactuita dintr-un sir de caractere (String) in felul urmator:

- primul caracter reprezinta tipul piesei
- al doilea caracter reprezinta inaltimea fantanii(h)
- al treilea reprezinta latimea(w)
- iar urmatoarele h*w caractere reprezinta configuratia fantanii(1 reprezinta celula libera si 0 reprezinta celula ocupata)

O actiune este reprezentata sub forma unui sir de 2 caractere unde:

- primul caracter reprezinta rotatia (cu cate grade va fi rotita piesa)
- al doilea carcater reprezinta "left" , cu cate spatii ne vom deplasa de peretele stang al fantanii

Am decis sa retin valorile lui Q intr-un Hashtable<String,Double> unde :

- * key = state+action
- * state = brick-type+board-configuration
- * action = rotation+left
- * value = valoarea lui Q

III. Implementare

Pentru implementarea jocului am folosit urmatoarele clase:

- **Brick**
 - aici am implementat tot ce tine de reprezentarea caramizilor
 - contine 2 campuri pentru latime si inaltime
 - reprezentarea caramizii este retinuta sub forma unei matrici bidimensionale de bytes
- **Board**
 - configuratia tablei(fantanii) a fost realizata tot sub forma unei matrici bidimensionale de bytes
- **Sarsa** - aceasta clasa contine implementarea algoritmului de invatare precum si metoda main
 - generateBrick - in functie de scenariu , va genera respectand probabilitatile din distributie o noua caramida(intoarce un int reprezentand tipul noii caramizi)
 - eGrredy - alege actiunea pentru starea respectiva
 - argmaxQ - intoarce actiunea de valoare maxima
 - learn - in aceasta metoda are loc implementarea efectiva a algoritmului de invatare(in timpul rularii acestei metode va fi populat Q)
- **QLearning**
 - pentru a testa algoritmul Q-Learning pur si simplu trebuie sa rulari Sarsa folosind epsilon = 0
 - astfel , strategia va fi 100% Greedy , eGreedy devenind practic argmaxQ , returnand mereu actiunea pentru care $Q[state,a]$ va avea valoare maxima

IV. Observatii

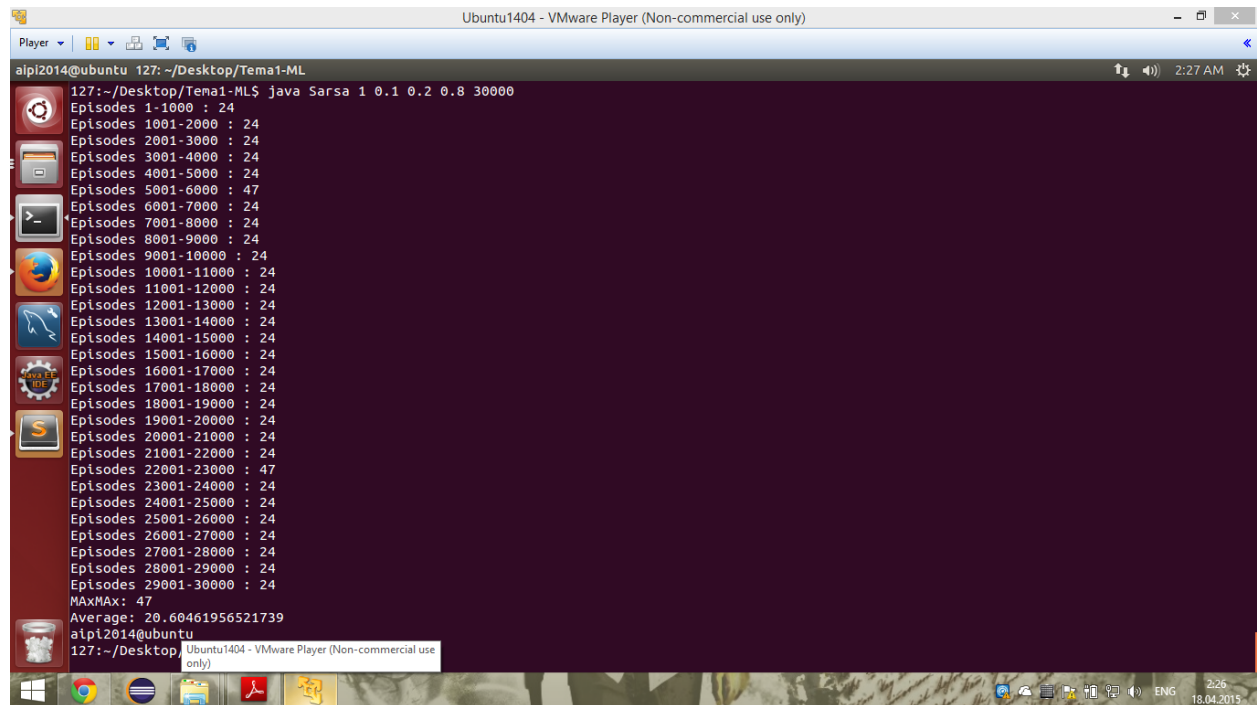
In general am observat ca cele mai bune rezultate le-am obtinut pentru epsilon = 0.95 asa ca aceasta este valoarea pe care am folosit-o in majoritatea testelor pentru generarea graficelor.

Am ales $\delta(\text{gamma}) = 0.8$ pentru a pune accent in procesul de invatare mai mult pe recompensele viitoare decat pe cele din prezent.

Am ales sa folosesc $\alpha=0.1$ (valoarea nu este prea mare , evitandu-se pericolul de a sari peste un set de valori ce ar converge spre un minim local, dar suficient de mare pentru a evita o convergere rapida).

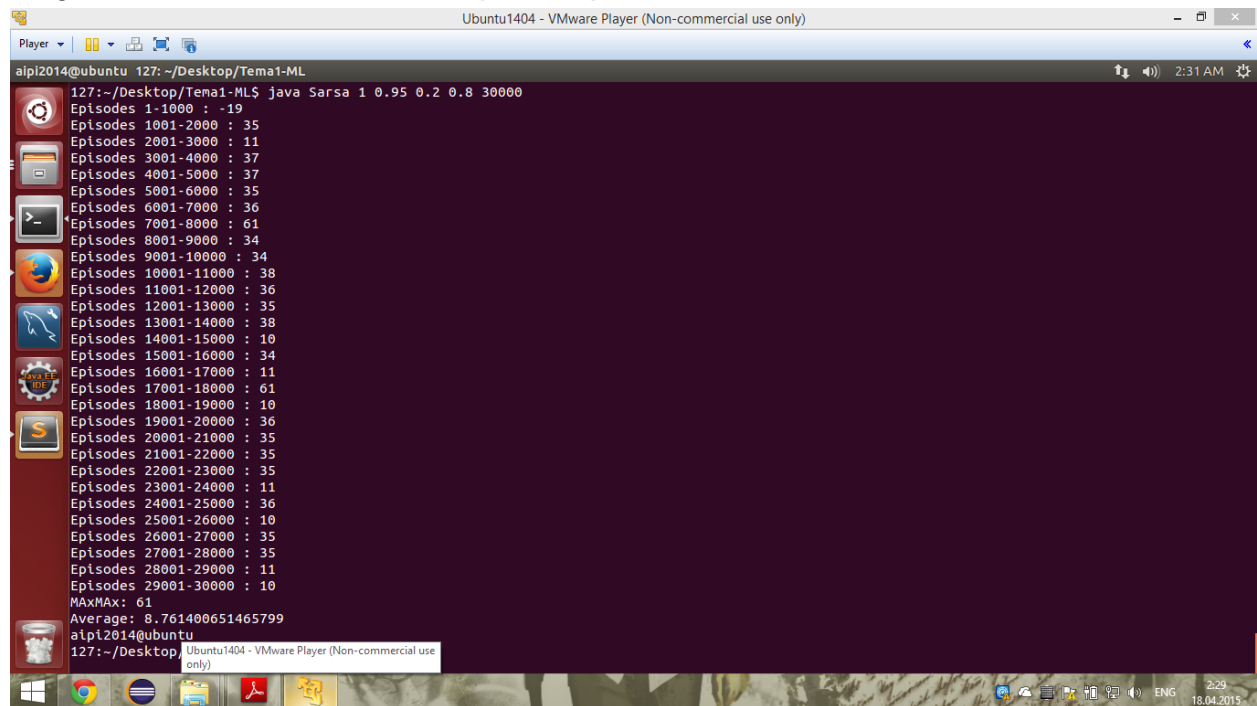
Inainte de a prezenta si analiza graficele voi prezenta rezultatele unui mic experiment. Am rulat algoritmul pe parcursul a 30000 episoade folosind $\delta=0.8$ si rata de invatare $\alpha=0.5$. Am afisat scorul maxim obtinut in fiecare interval de cate 1000 de episoade precum si Maximul general si media valorilor pozitive.

In figura de mai jos sunt rezultatele pentru $\epsilon = 0.1$



```
127:~/Desktop/Tema1-ML$ java Sarsa 1 0.1 0.2 0.8 30000
Episodes 1-1000 : 24
Episodes 1001-2000 : 24
Episodes 2001-3000 : 24
Episodes 3001-4000 : 24
Episodes 4001-5000 : 24
Episodes 5001-6000 : 47
Episodes 6001-7000 : 24
Episodes 7001-8000 : 24
Episodes 8001-9000 : 24
Episodes 9001-10000 : 24
Episodes 10001-11000 : 24
Episodes 11001-12000 : 24
Episodes 12001-13000 : 24
Episodes 13001-14000 : 24
Episodes 14001-15000 : 24
Episodes 15001-16000 : 24
Episodes 16001-17000 : 24
Episodes 17001-18000 : 24
Episodes 18001-19000 : 24
Episodes 19001-20000 : 24
Episodes 20001-21000 : 24
Episodes 21001-22000 : 24
Episodes 22001-23000 : 47
Episodes 23001-24000 : 24
Episodes 24001-25000 : 24
Episodes 25001-26000 : 24
Episodes 26001-27000 : 24
Episodes 27001-28000 : 24
Episodes 28001-29000 : 24
Episodes 29001-30000 : 24
MAXMAX: 47
Average: 20.60461956521739
alpi2014@ubuntu
127:~/Desktop/
```

In figura urmatoare sunt rezultatele pentru $\epsilon = 0.95$.



```
127:~/Desktop/Tema1-ML$ java Sarsa 1 0.95 0.2 0.8 30000
Episodes 1-1000 : -19
Episodes 1001-2000 : 35
Episodes 2001-3000 : 11
Episodes 3001-4000 : 37
Episodes 4001-5000 : 37
Episodes 5001-6000 : 35
Episodes 6001-7000 : 36
Episodes 7001-8000 : 61
Episodes 8001-9000 : 34
Episodes 9001-10000 : 34
Episodes 10001-11000 : 38
Episodes 11001-12000 : 36
Episodes 12001-13000 : 35
Episodes 13001-14000 : 38
Episodes 14001-15000 : 10
Episodes 15001-16000 : 34
Episodes 16001-17000 : 11
Episodes 17001-18000 : 61
Episodes 18001-19000 : 10
Episodes 19001-20000 : 36
Episodes 20001-21000 : 35
Episodes 21001-22000 : 35
Episodes 22001-23000 : 35
Episodes 23001-24000 : 11
Episodes 24001-25000 : 36
Episodes 25001-26000 : 10
Episodes 26001-27000 : 35
Episodes 27001-28000 : 35
Episodes 28001-29000 : 11
Episodes 29001-30000 : 10
MAXMAX: 61
Average: 8.761400651465799
alpi2014@ubuntu
127:~/Desktop/
```

Dupa cum se poate observa , in cazul unui epsilon mic , strategia tinde sa devina una de tip Greedy.Algoritmul tinde sa converge destul de rapid spre un minim local(seamana cu efectul de “platou”).Media rezultatelor este destul de apropiata si am observat ca valoarea nu mai creste chiar daca voi creste numarul de episoade(in principiu nu va mai incerca sa viziteze noi stari si alege mereu aceleasi).

In schimb , din figura 2 se poate observa ca pentru un epsilon mare(0.95) , valoarea maxima obtinuta este mult mai mare.Este adevarat ca in majoritatea cazurilor va alege o mutare random (de aici si rezulta media aritmetica destul de mica fata de cazul celalalt) , dar usor-usor , prin explorare , incepe sa invete si gaseste valori mai mari (61>24)decat minimul local gasit in primul exemplu(eps=0.1).Probabil ca in cazul in care l-as lasa sa ruleze pentru un numar suficient de mare de episoade , ar gasi scorul maxim posibil sau o valoare destul de apropiata de acesta.

Mi-as fi dorit sa realizez testele pe un esantion mai mare de episoade , dar resursele fizice ale laptop-ului nu mi-au permis.

V. Grafice

Din ce am observat , cu cat cresc mai mult valoarea lui epsilon(cu cat tinde mai mult spre 1) , numarul de stari explorate creste mai rapid.

In graficele urmatoare am tinut cont de numarul de stari explorate(X) si de valoarea maxima a lui(Y).

Scenariul 1

Deoarece numarul de stari posibile in cazul scenariului 1 este destul de limitat, am decis sa tin cont de numarul total de actualizari a lui Q si nu de numarul de stari originale vizitate. Pentru acest scenariu , numarul maxim de stari unice(prin care apoi s-a trecut de mai multe ori) este 15.

In fisierele folosite pentru generarea graficelor am pus scorurile obtinute si numarul de episoade

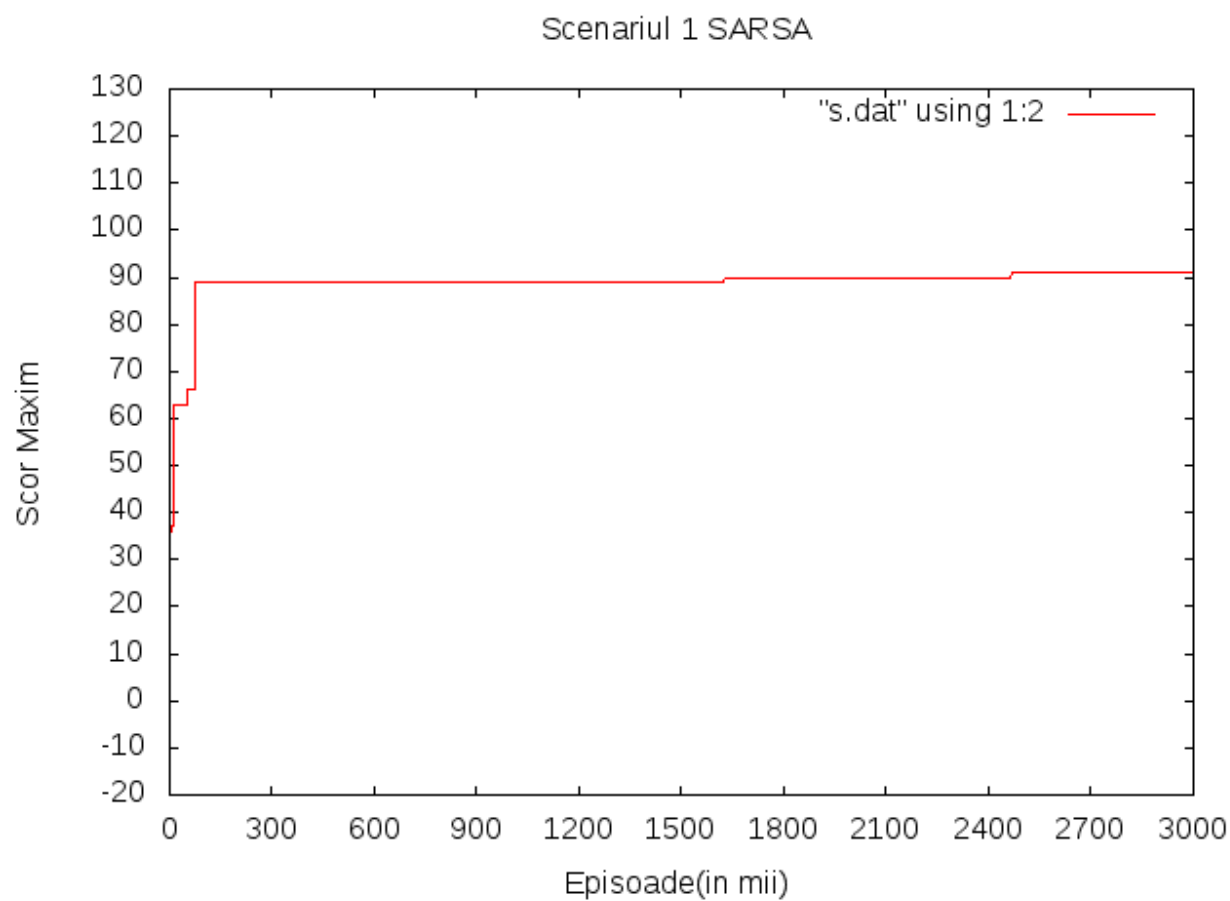
In graficul urmator am folosit :

epsilon = 0.95

alfa = 0.1

delta = 0.8

numar_episoade = 3000000



MaxValue = 92

Din grafic se poate observa ca scorul maxim se modifica din ce in ce mai greu , pentru a-si imbunatati scorul , trebuie sa isi actualizeze din ce in ce mai mult starile.Probabil ca in curand , aceasta crestere va stangna.

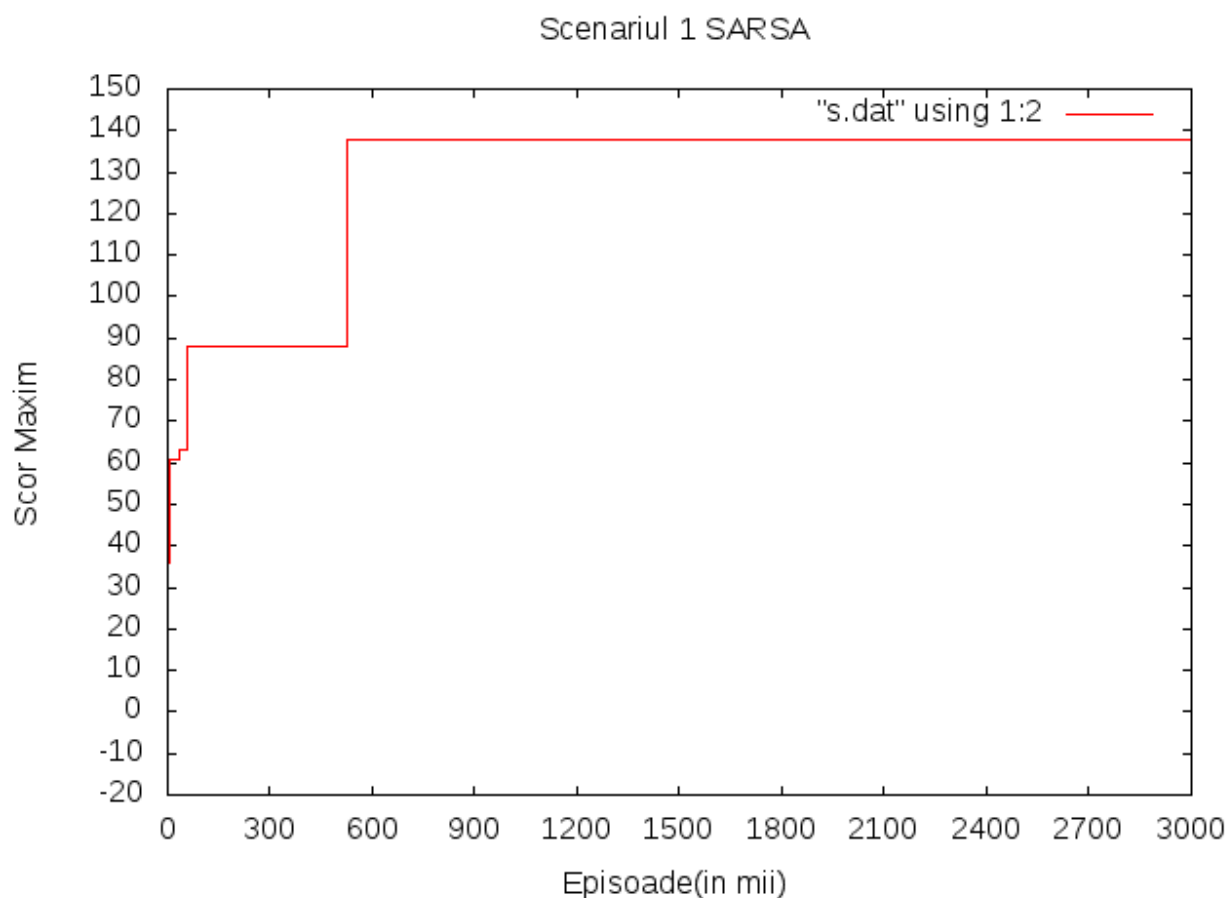
In graficul urmator am folosit :

epsilon = 0.95

alfa = 0.9

delta = 0.8

numar_episoade = 3000000



In al doilea exemplu am folosit parametri identici ca la primul , mai putin alfa. In primul grafic , pare ca algoritmul ar converge destul de greu , asa ca am decis sa folosesc o rata de invatare mare (0.9). Ca rezultat , se pare ca algoritmul ar converge mai repede spre maximul global , iar valoarea obtinuta la final este mai apropiata de valoarea ideala (optima).

MaxValue = 134

Scenariul 2

epsilon = 0.95

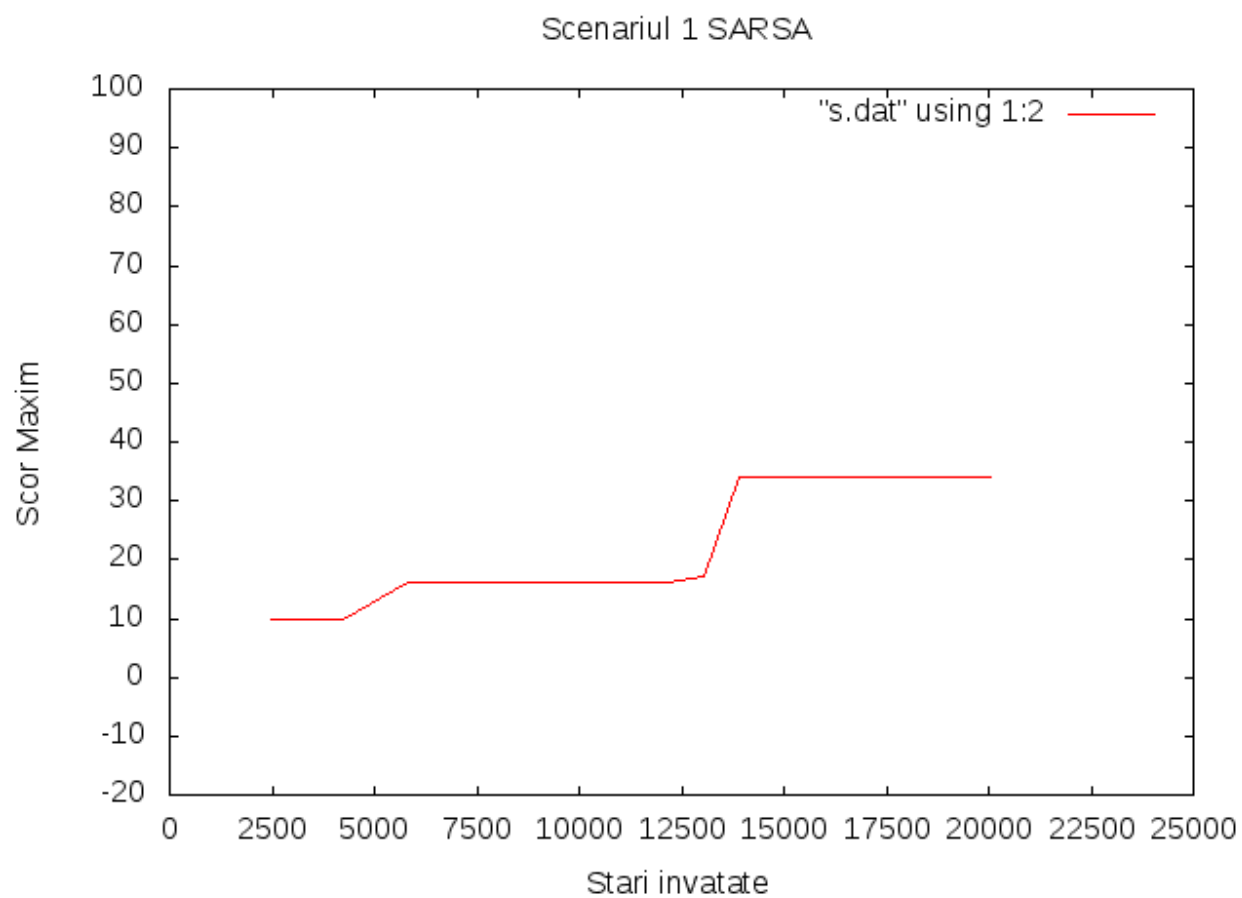
alfa = 0.9

delta = 0.8

numar_episoade = 20000

Numar stari invatate: 20027

MaxValue = 37

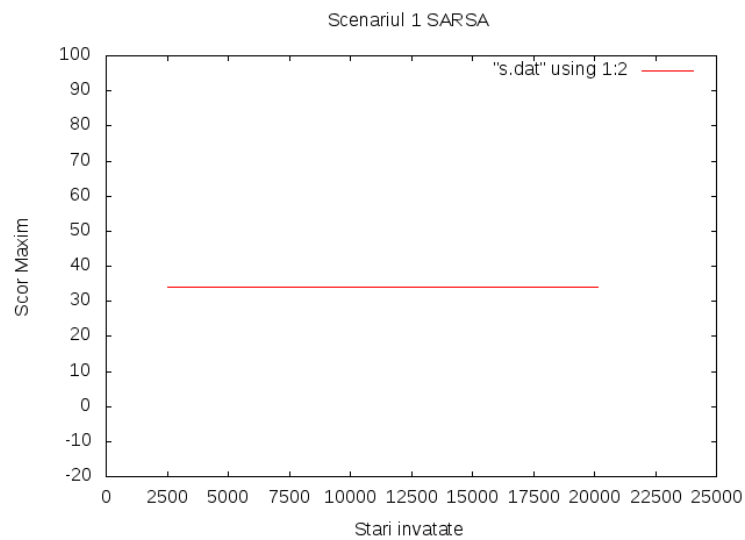


epsilon = 0.95

alfa = 0.1

delta = 0.8

numar_episoade = 20140



Numar stari invatate:20140

MAxValue = 34

Din cele 2 grafice am observat ca modificarea ratei de invatare nu modifica considerabil numarul de stari explorate. La al doilea grafic, datorita valorii mici a lui alpha , se observa tendinta de “conservare” a valorii.

OBSERVATIE !

Desi in grafic scrie Scenariul 1 , aceste grafice au fost generate folosind date obtinute prin rulara scebriului 2. Este doar o greseala de scriere. (La momentul cand am descoperit problema , nu mai aveam datele).

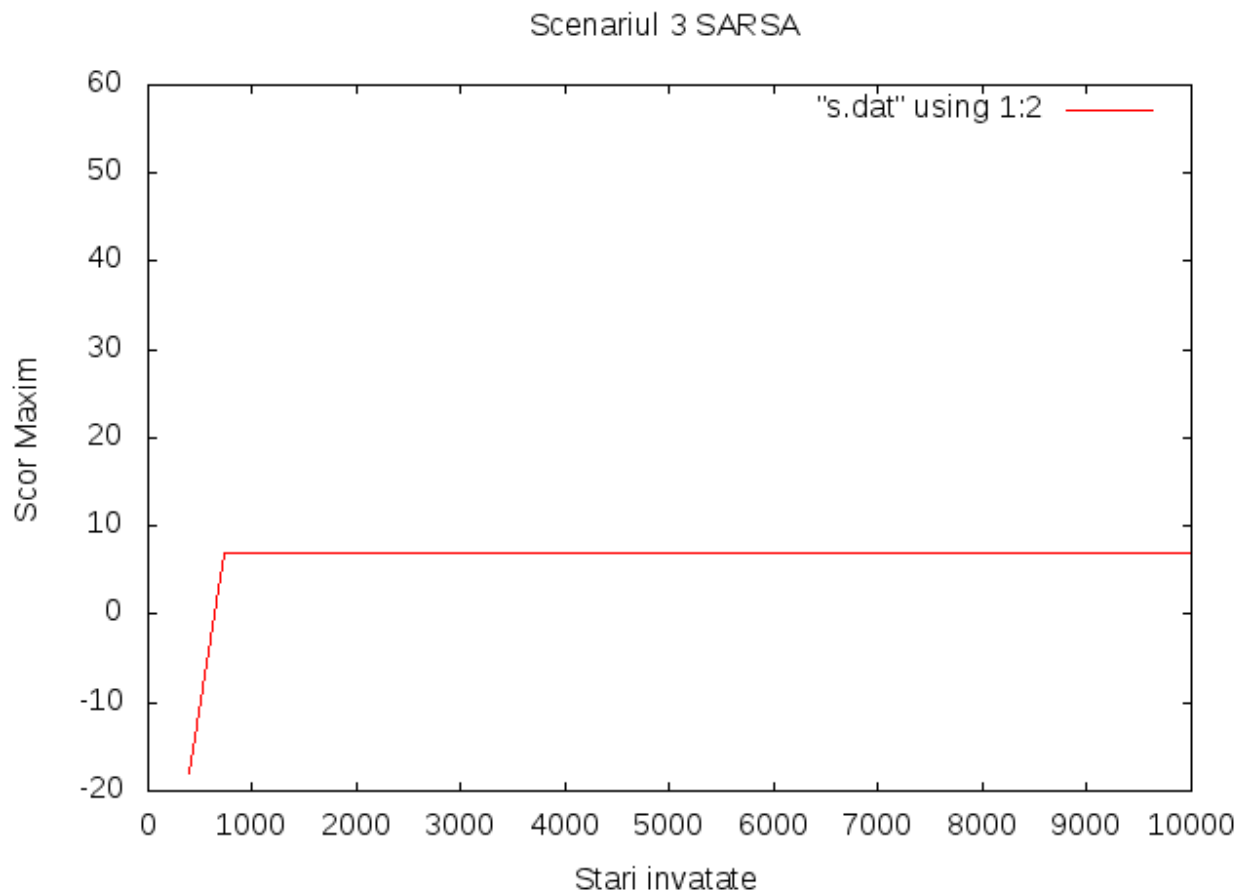
Scenariul 3

epsilon = 0.95

alfa = 0.1

delta = 0.8

numar_episoade = 10000



Numar stari invatate:23900

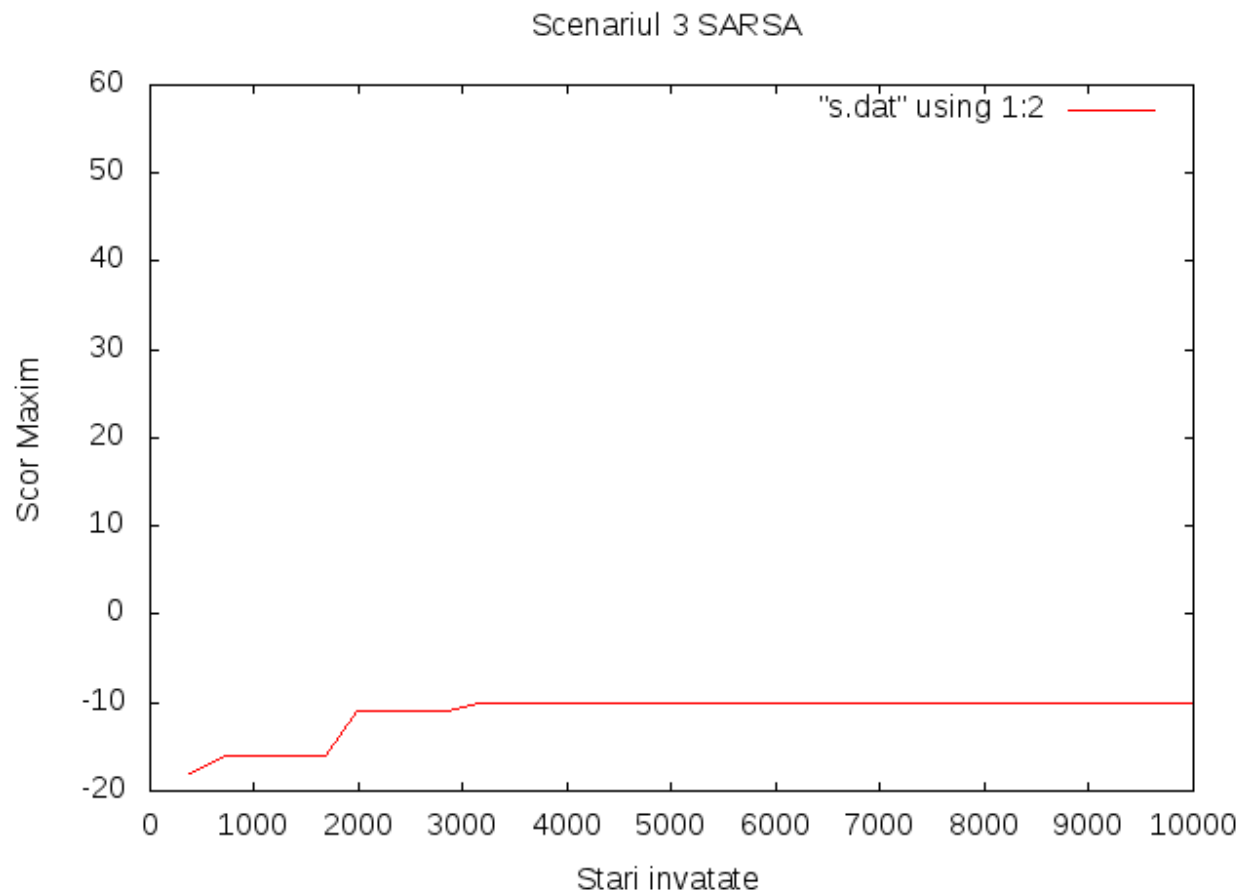
Pentru urmatorul exemplu am ales un coeficient de atenuare mai mic(0.2).Astfel , din al doilea grafic se poate observa faptul ca agentul tinde sa devina mai "oportunist" , preferand recompensa imediata celei viitoare.In acest caz , aceasta se dovedeste a fi o strategie proasta, rezultatele din primul grafic fiind mai bune.

epsilon = 0.95

alfa = 0.1

delta = 0.2

numar_episoade = 10000



Numar stari invatate:23900

Scenariul 4

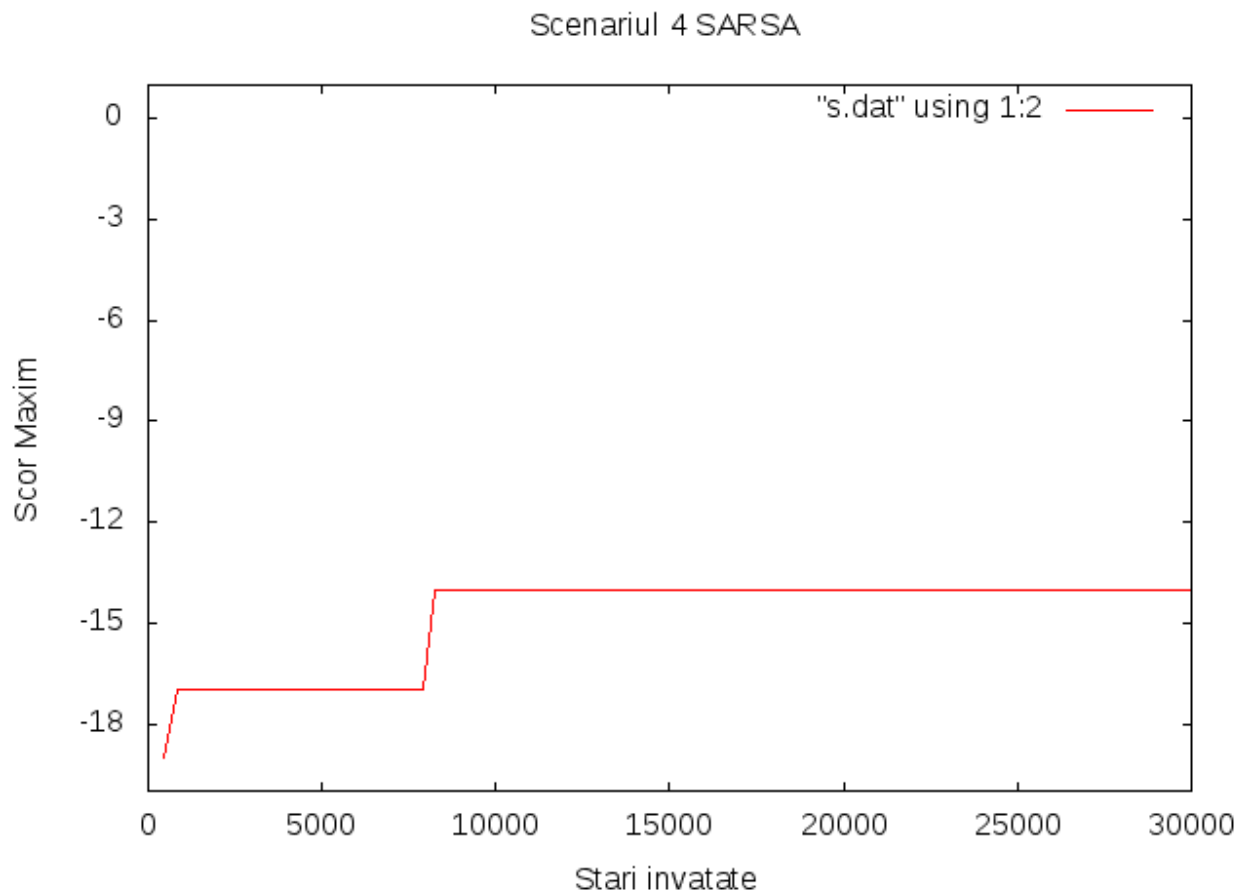
$\epsilon = 0.9$

$\alpha = 0.2$

$\delta = 0.8$

numar_episoade = 10000

În cazul scenariului 4, unde numărul stărilor posibile este foarte mare, este normal ca în timpul fiecărui "episod" să fie explorat un număr mare de stări. Din graficul de mai jos, agentul pare să învețe, scorul fiind pe un trend ascendent (este adevărat că încă este negativ dar pare să crească). Pentru a obține un rezultat pozitiv, este nevoie ca noile stări să fie parcurse de (mult) mai multe ori, pentru a actualiza (îmbunătăți) valorile din Q. Din păcate, nu am avut resursele necesare de a rula pe un număr mai mare de episoade.



Numar stari invatate:29960

VI. Comparatie SARSA vs Q-Learning (Bonus)

Scenariul 1

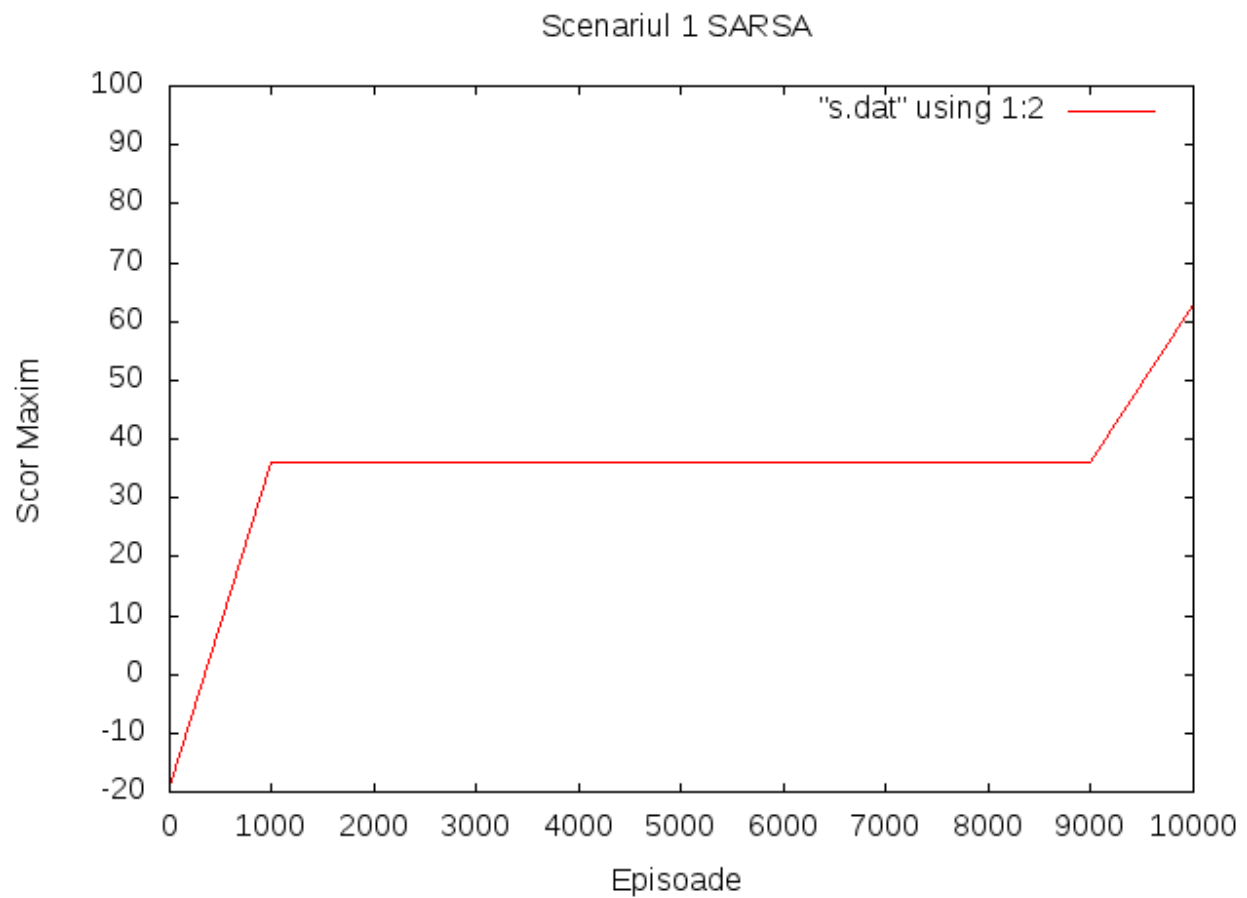
Parametri SARSA :

$\epsilon = 0.95$

$\alpha = 0.1$

$\delta = 0.8$

numar_episoade = 10000



MaxValue : 62

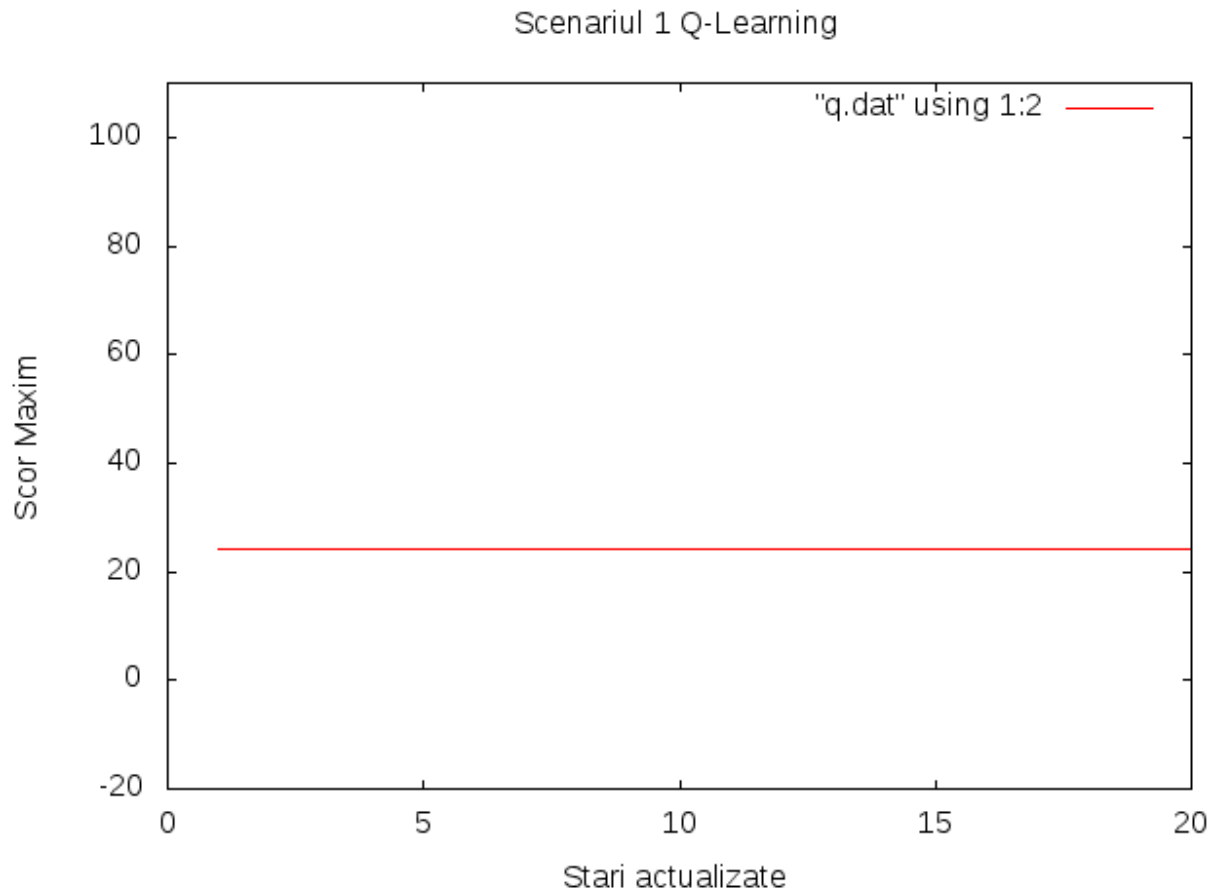
Stari(distincte) vizitate : 15

Parametri Q-Learning

$\alpha = 0.1$

$\delta = 0.8$

numar_episoade = 10000
MaxValue : 24
Stari vizitate: 1



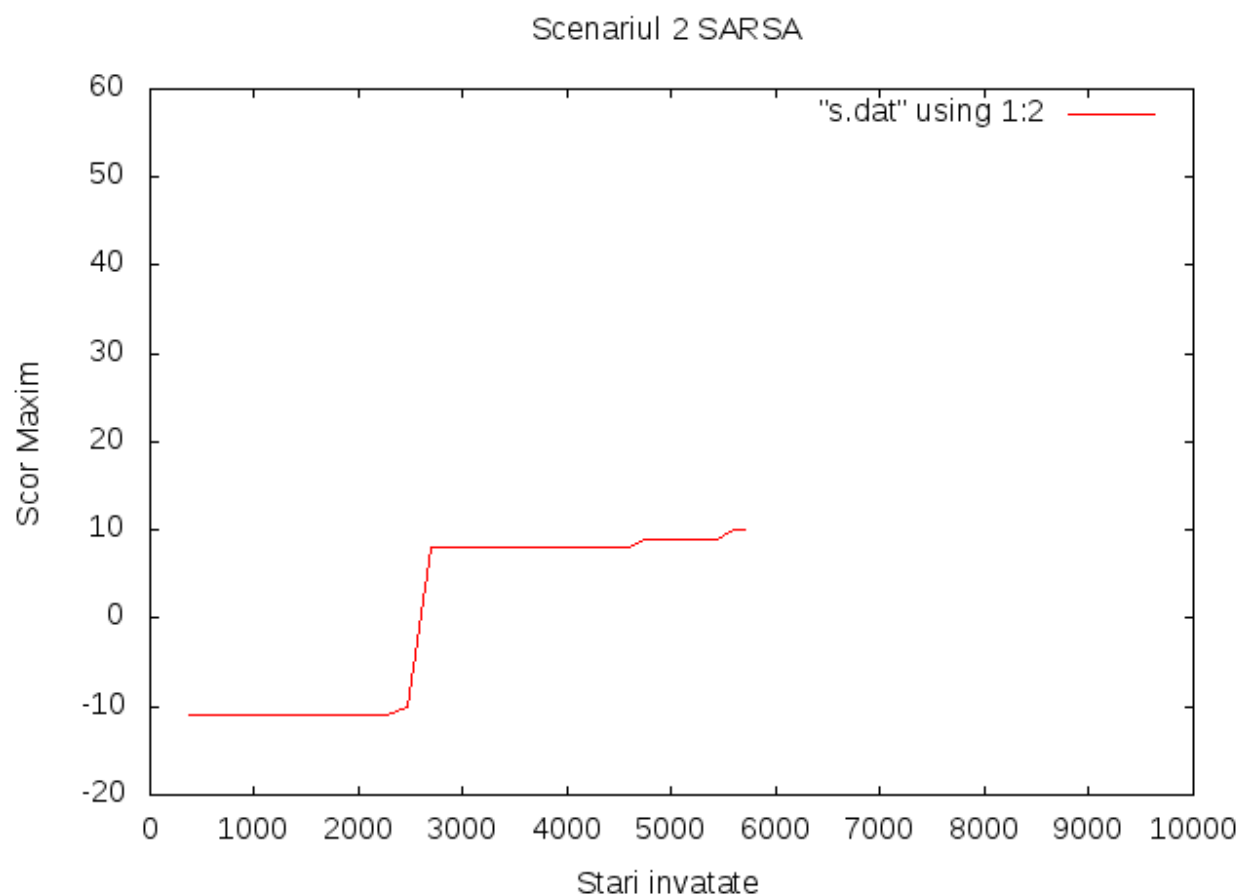
Din grafice se poate observa ca in cazul Q-Learning , numarul de stari explorate este mult mai mic decat in cazul Sarsa.Odata ce se gaseste o stare cu un castig(viitor) satisfacator , algoritmul tinde sa converga spre valoarea acesteia , mereu fiind preferata actiunea ce aduce un castig maxim in starea viitoare.

Chiar daca la inceput Sarsa va avea rezultate mai slabe(multe actiuni sunt alese random) , pe parcurs isi va imbunatati politica si va obtine rezultate mai bune.

Scenariul 2

Parametri SARSA

epsilon = 0.5
alfa = 0.2
delta = 0.8
numar_episoade = 3000



Numar stari invatate:5700

Graficul de emai sus a fost generat folosind urmatoarele date:

Nr.Stari	Valoare Scor Maxim
378	-11
665	-11
954	-11
1199	-11
1453	-11
1667	-11
1868	-11
2062	-11
2259	-11
2467	-10
2687	8
2880	8
3056	8

3249	8
3441	8
3608	8
3771	8
3932	8
4082	8
4249	8
4413	8
4577	8
4746	9
4912	9
5062	9
5186	9
5310	9
5440	9
5580	10
5706	10

Datele de mai sus reprezinta numarul total de stari explorate si valoarea scorului maxim dupa fiecare 100 de episoade. Se poate observa faptul ca algoritmul incepe sa invete mai greu (epsilon este 0.5), astfel ca daca la inceput dupa fiecare 100 de episoade explora 250-300 de stari noi, acum dupa 3000 de episoade ajunge sa exploreze cam 120 de noi stari dupa 100 de episoade.

Parametri Q-Learning

alfa = 0.2

delta = 0.8

numar_episoade = 3000

In urma rularii algoritmului am obtinut urmatoarele date:

Nr.Stari	Valoare Scor Maxim
189	-20
270	-20
339	-20
386	-20
428	-20
453	-20
476	-20
496	-20
514	-20
538	-20

567	-20
577	-20
586	-20
599	-20
609	-20
621	-20
629	-20
640	-20
655	-20
659	-20

Numar stari invatate:659

Q-Learning nu invata la fel de bine in acest caz.Numarul de stari explorat este mult mai mic fata de sarsa si tinde chiar sa ramana constant(dupa un anumit numar d episoade , acesta nu va mai explora deloc noi stari , abordand 100% o strategie Greedy).

Scenariul 3

Scenariul 4

VII.Rulare

Pentru compilarea surselor folositi make.Aceasta va genera 1 executabil: Sarsa.

Pentru a rula Sarsa folositi comanda:

```
java Sarsa tip_scenariu epsilon alfa delta episoade
```

Valoare epsilon trebuie sa fie un double intre 0.0 si 1.0 .

Scenariu ia valori din multimea {1,2,3,4}

Alfa si Delta trebuie sa fie tot de tip double.

Episoade reprezinta numarul de episoade pentru care se va executa algoritmul.

VIII.Arhiva

❖ Sursele:

- Sarsa.java
- Brick.java
- Board.java

❖ Readme

❖ Makefile