

# Învățare Automată - Tema 1

## Învățare prin recompensă: Cărămizi

Tudor Berariu

Laboratorul AIMAS

Facultatea de Automatică și Calculatoare

24 martie 2015

### 1 Pe scurt...

Se cere să se implementeze algoritmul **SARSA** pentru învățarea unei strategii de joc pentru o variantă a jocului Tetris. Se va testa eficiența algoritmului pentru diferite niveluri de complexitate a jocului.

### 2 Motivația temei

Scopul acestei teme îl reprezintă familiarizarea cu un algoritm de învățare prin recompensă, **SARSA**, precum și implementarea acestuia. Se urmărește înțelegerea diferențelor dintre acesta și algoritmul **Q-learning**, precum și aplicarea **SARSA** pe o problemă în care numărul stărilor este foarte mare.

### 3 Descrierea jocului

Jocul **Cărămizi** este o variantă a jocului Tetris concepută pentru studenții cursului de Învățare Automată cu o listă de modificări descrisă în cele ce urmează.

Se consideră o fântână reprezentată ca un spațiu de înălțime  $H$  și lățime  $W$  și 7 tipuri de cărămizi, enumerate în Tabela 1.

#### 3.1 Obiectivul jucătorului

Jucătorul, denumit de aici înainte **Mester**, este cel care așează cărămizile primite în fântână. Scopul lui este să formeze linii orizontale complete pentru a obține un punctaj cât mai mare.

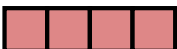
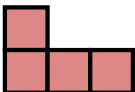
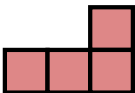
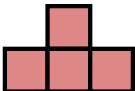
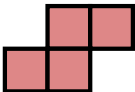
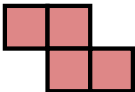

A	
B	
C	
D	
E	
F	
G	

Tabela 1: Tipurile de cărămizi

Meșterul va așeza cărămizile primite în fântână unele peste altele. O cărămidă va putea fi așezată doar deasupra celor fixate deja. Spre deosebire de Tetris clasic, aici cărămida va fi întâi rotită și așezată deasupra poziției dorite și abia apoi lăsată să cadă fără a mai fi deplasată sau rotită până ajunge jos (vezi Figura 1).

Jocul se încheie atunci când au fost așezate **24** cărămizi sau atunci când construcția depășește înălțimea  $H$ .

### 3.2 Punctarea

Atunci când în urma așezării unei cărămizi se completează una sau mai multe linii, acestea vor dispărea (*magie!*) și Meșterul va primi:

- 1 punct pentru o linie completată,
- 3 puncte pentru 2 linii completate,
- 9 puncte pentru 3 linii completate,

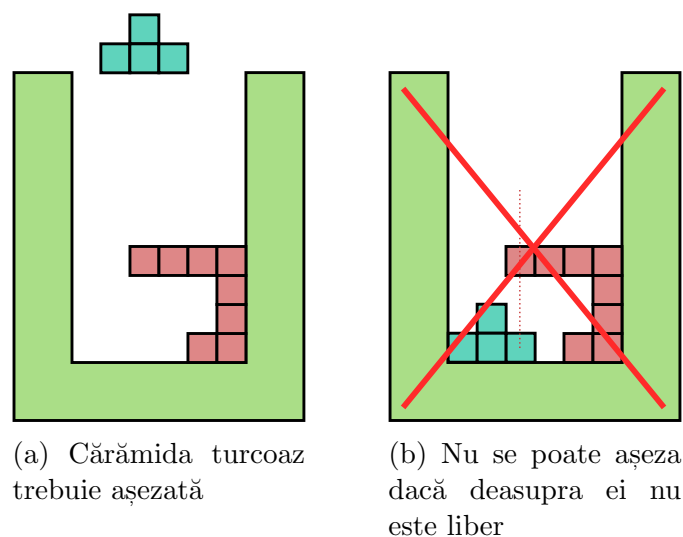


Figura 1: Exemplu de așezare incorectă a unei cărămizi

- 27 puncte pentru 4 linii completate.

Liniile completate nu trebuie să fie consecutive (vezi exemplul din Figura 2).

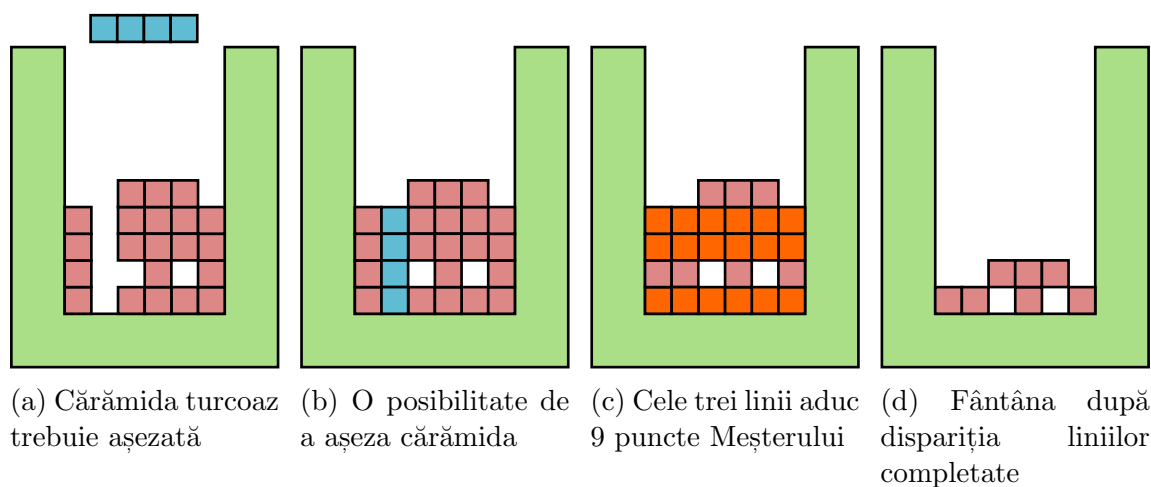


Figura 2: Exemplu de completare a liniilor

Dacă jocul se încheie prin depășirea înălțimii fântânii, atunci Meșterului i se vor retrage 20 de puncte.

Toate regulile de punctare sunt trecute în Tabela 2.

Eveniment	Puncte Meșter
1 linie completată	1
2 linii completate	3
3 linii completate	9
4 linii completate	27
Depășirea înălțimii fântânii	-20

Tabela 2: Punctarea

## 4 Algoritmul SARSA

Algoritmul **SARSA** (State-Action-Reward-State-Action) [RN94, SS96] este similar algoritmului **Q-learning**.

---

### Algoritmul 1 Algoritmul SARSA

---

```

1: pentru toate stările  $s$  execută
2:   pentru toate acțiunile  $a \in A(s)$  execută
3:      $Q(s,a) \leftarrow 0$ 
4:   termină ciclu
5: termină ciclu
6: pentru toate episoadele execută
7:    $s \leftarrow$  stare inițială
8:    $a \leftarrow \epsilon$ -greedy( $Q, s$ )
9:   repetă
10:    execută  $a$ 
11:    observă  $s', r$ 
12:     $a' \leftarrow \epsilon$ -greedy( $Q, s'$ )
13:     $Q(s, a) \leftarrow Q(s, a) + \alpha \left( r + \gamma Q(s', a') - Q(s, a) \right)$ 
14:     $s \leftarrow s'$ 
15:     $a \leftarrow a'$ 
16:   până când  $s$  este stare finală
17: termină ciclu

```

---

Regula de actualizare a funcției  $Q$  este diferită față de cea a algoritmului **Q-learning** (vezi Formula 1) prin faptul că pentru estimarea *recompenselor viitoare* nu se alege întotdeauna valoarea maximă pentru  $Q(s', *)$ , ci se alege o acțiune  $a'$  conform politicii (care include și componenta de explorare) și se utilizează valoarea  $Q(s', a')$ . Din puncte de vedere conceptual, această diferență se traduce prin faptul că **SARSA** este un algoritm *on-policy* spre deosebire de **Q-learning** care este un algoritm *off-policy*.

Formula de actualizare a funcției  $Q$  în algoritmul **SARSA**:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t \left( R(s_t, a_t) + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right) \quad (1)$$

În cazul unui proces de învățare continuu este de așteptat ca algoritmi on-policy să se comporte mai bine decât algoritmi off-policy întrucât politica învățată ține cont și de componenta de explorare. Acest lucru este util mai ales atunci când mediul este unul dinamic și nu se poate renunța la explorare.

Avantajul algoritmului **SARSA** față de **Q-learning** este foarte bine surprinsă în următorul paragraf din [BI99]: [...] *perhaps it would be better to learn the policy that is optimal, given that you will explore  $\epsilon$  of the time. It would be like a person who, when walking, always takes a random step every 100 paces or so. Such a person would avoid walking along the top of a cliff, even when that is the optimal policy for a person who doesn't explore randomly.*

Pseudocodul **SARSA** este prezentat în Algoritmul 1. Pentru alegerea acțiunii dintr-o stare  $s$  folosiți tehnica  $\epsilon$ -greedy, descrisă de Algoritmul 2.

---

#### Algoritmul 2 Algoritmul $\epsilon$ -greedy

---

**Intrări:**  $Q$ , starea  $s, \epsilon, A$

**Ieșire:** acțiunea  $a$

- 1:  $A' \leftarrow \{a \in A \mid a \text{ acțiune validă în } s\}$
  - 2:  $p \leftarrow \text{random}(0, 1)$
  - 3: **dacă**  $p < \epsilon$  **atunci**
  - 4:      $a \leftarrow \text{random}(A')$
  - 5: **altfel**
  - 6:      $a \leftarrow \underset{act \in A'}{\operatorname{argmax}} Q(s, act)$
  - 7: **termină dacă**
- 

## 5 Cerințe

### [6 puncte] Implementarea algoritmului **SARSA**

Să se implementeze algoritmul **SARSA** pentru a determina o politică (o strategie de joc) cât mai bună pentru **Meșter** în jocul **Cărmizi**. Se impune o optimizare a reprezentării stărilor. Implementarea se poate face de la zero sau se pot folosi ca punct de plecare serverul de joc și scripturile din arhivă (vezi Anexa A).

Se vor acorda 4 puncte pentru implementarea corectă a algoritmului și 2 puncte pentru optimizarea reprezentării stărilor. Compararea mai multor reprezentări și studierea implicației fiecăreia asupra învățării (scor obținut în urma învățării și timp necesar pentru învățare) pot aduce punctaj suplimentar.

### [4 puncte] Evaluarea eficienței algoritmului

Se va evalua metoda de învățare propusă prin observarea scorului obținut de **Meșter** de-a lungul unui număr mare de jocuri (**Meșterul** își păstrează utilitățile de la un joc

la altul) pentru patru niveluri de dificultate a jocului.

Se vor realiza două grafice: unul cu evoluția scorului obținut de **Meșter**, eventual netezit, și unul pentru evoluția numărului de stări explorate pe parcursul învățării.

Se va testa algoritmul pentru cele patru cazuri descrise în Tabela 3 și pentru unul la alegere.





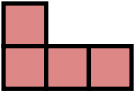
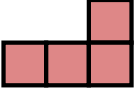

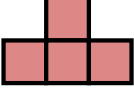
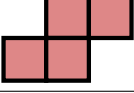
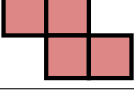
Nr	Fântâna		Căramizile		
	$H$	$W$	Distribuția fixă?	Distribuția de probabilitate	
				Căramida	Probabilitatea de apariție
1	4	4	Da		1
2	8	5	Da		0.5
					0.5
3	8	5	Da		0.2
					0.4
					0.4
4	8	6	Da		0.16667
					0.66665
					0.08334
					0.08334

Tabela 3: Scenariile de test

Trebuie căutate valori cât mai bune pentru rata de învățare ( $\alpha$ ), factorul de atenuare

( $\gamma$ ) și probabilitatea de explorare ( $\epsilon$ ).

## [2 puncte] BONUS

Faceți o comparație între algoritmul **SARSA** și algoritmul **Q-learning** pentru scenariile date. Aduce **SARSA** vreun avantaj față de **Q-learning** pentru învățarea on-line? Interpretați și explicați rezultatele obținute.

## 6 Trimiterea temei

În arhiva temei includeți:

- toate fișierele sursă (cu **Makefile** / script de compilare și rulare)
- fișier **pdf** cu descrierea implementării și a rezultatelor obținute:
  - reprezentarea stărilor (eventual comparație făcută între mai multe reprezentări încercate);
  - grafice cu scorul obținut pe parcursul învățării și cu numărul de stări pentru scenariile propuse;
  - comparația (eventual grafice suprapuse) între **Q-learning** și **SARSA**

## A Arhiva

În arhiva ce însoțește acest document se găsește implementarea unui server de joc pentru **Cărămizi**. Pentru a folosi acest server de joc la testarea temei trebuie să comunicați prin pipe-uri cu nume (engl. *named pipes*) respectând protocolul descris în Figura 3 și *narat* în Secțiunea B.1.

## B Conținutul arhivei

- **src/** - sursele serverului;
- **Makefile** - rețetele pentru compilarea serverului;
- **delete\_pipes.sh** și **create\_pipes.sh** - scripturi pentru crearea și ștergerea pipe-urilor cu nume (denumite **server\_to\_player** și **player\_to\_server**);
- **distributions/** - director cu fișierele cu distribuțiile de probabilități de apariție a cărămizilor;
- **start\_server.sh** - script pentru lansarea serverului pentru scenariile propuse în cerințe.

## B.1 Protocolul de comunicație

Toate mesajele schimbate de cele două entități (serverul de joc și meșterul) se încheie cu `\n`.

Serverul va controla evoluția jocului și îi va cere Meșterului să poziționeze fiecare cărămidă.

Întâi jucătorul trebuie să se conecteze la server (adică să deschidă cele două pipe-uri cu nume) și să trimită un mesaj cu numele lui (`MESHTER\n`). Drept răspuns va primi numărul de jocuri și dimensiunea fântâni (de exemplu un mesaj `100000,10,6\n`).

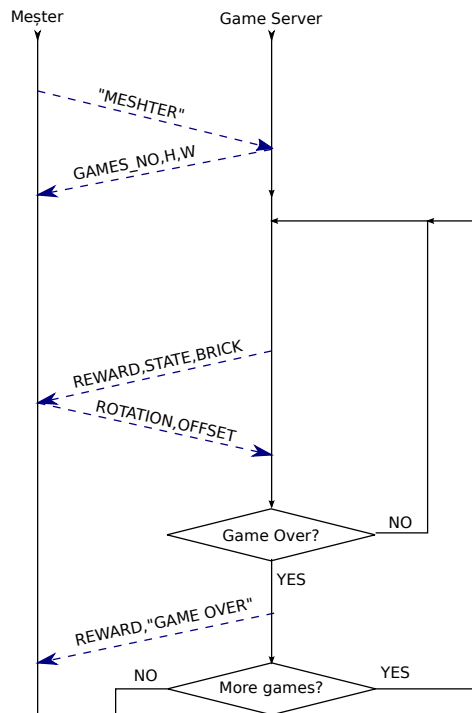


Figura 3: Protocolul de comunicație

Începând din acest moment, pentru fiecare joc:

1. serverul îi trimite Meșterului un mesaj ce conține numărul de puncte obținute pentru mutarea anterioară, starea jocului, dar și piesa ce trebuie poziționată. Un exemplu de mesaj este:

9,| | | #| ##|## #|,C\n

2. Meșterul îi trimite serverului două numere: numărul de rotații în sensul acelor de ceasornic ce trebuie executate și numărul de celule distanță de la peretele din stânga. Un exemplu de mesaj este:

3,2\n



pentru o piesă rotită cu 270 de grade și care va fi plasată la distanță de 2 căsuțe de peretele din stânga (începând cu a treia coloană).

3. Dacă jocul s-a încheiat (s-au pus 24 piese sau s-a depășit înălțimea fântânii), serverul îi trimite Meșterului un mesaj cu ultimele puncte câștigate și textul `GAME OVER`. De exemplu:

```
27,GAME OVER\n
```

## B.2 Utilizarea fișierelor din arhivă

1. Se creează pipe-urile cu nume `server_to_player` și `player_to_server`:

```
$ ./delete_pipes
$ ./create_pipes
```

2. Se lansează serverul de joc:

```
$ ./bricks-game-server --gamesNo 100 --height 8 --width 6 \
    --bricks distributions/dist2 \
    --verbose 1 --log STDOUT \
/
```

Starting server for 1 games on a 8x6 board.

Waiting for player...

- Se pot configura:
  - numărul de jocuri (`--gamesNo`), implicit 100000;
  - dimensiunea fântânii (`--height` și `--width`), implicit  $8 \times 6$ ;
  - fișierul ce conține distribuția folosită pentru generarea cărămizilor (`--bricks`), implicit `distributions/dist1`;
  - pipe-ul cu nume pentru comunicarea de la jucător către server (`--inFIFO`) implicit `player_to_server`;
  - pipe-ul cu nume pentru comunicarea de la server către jucător (`--outFIFO`) implicit `server_to_player`;
  - controlul afișării (`--verbose`), implicit 0 (fals);
  - fișierul către care să se scrie evoluția jocului (`--log`), implicit `STDOUT`, adică afișare la ecran;

3. Se lansează jucătorul (într-un terminal separat):

```
$ ./bricks-game-player
```

Un Meșter care așează piesele aleator este implementat în `bricks-game-player.cc`. Pentru rezolvarea temei se poate porni de la codul din acest fișier.

Rulați `./start_server <n>` pentru a rula scenariul cu numărul `n`.

## Bibliografie

- [BI99] L.C. Baird III. *Reinforcement learning through gradient descent*. PhD thesis, Citeseer, 1999.
- [RN94] G.A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering, 1994.
- [SS96] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, pages 123–158, 1996.