Control prin învăţare

Master ICAF, An 1 Sem 2

Lucian Buşoniu, Ştefan Pîrje

Partea III

Învăţarea prin recompensă

Partea III în plan

- Problema de învăţare prin recompensă
- Soluţia optimală
- Programarea dinamică exactă
- **Învăţarea prin recompensă exactă**
- Tehnici de aproximare
- Programarea dinamică cu aproximare
- Învăţarea prin recompensă cu aproximare
- Reţele neurale profunde
- Învăţarea prin recompensă cu reţele neurale profunde

Gama de algoritmi

După utilizarea unui model:

- Bazat pe model: f, ρ cunoscute
- Fără model: doar date (**învăţarea prin recompensă**)

După nivelul de interacţiune:

- Offline: algoritmul rulează în avans
- Online: algoritmul controlează direct sistemul

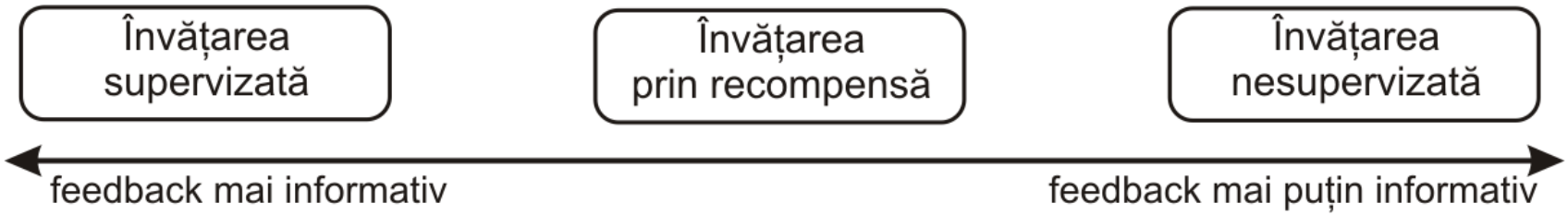
Exact vs. cu aproximare:

- Exact: x, u număr mic de valori discrete
- Cu aproximare: x, u continue (sau multe valori discrete)

Conţinut partea III

1. Monte Carlo, MC
2. Nevoia de explorare
3. Diferenţe temporale, TD
4. Accelerarea metodelor TD
5. Încheiere

Învăţarea prin recompensă în gama învăţării automate



- Supervizată: pentru fiecare eşantion de antrenare, **ieşirea corectă** este cunoscută
- Nesupervizată: doar intrări, **fără ieşiri**; găseşte structuri în date
- Prin recompensă: acţiunile corecte nu sunt disponibile, **doar recompense**

Important: învăţarea prin recompensă găseşte un **control optimal pentru un sistem dinamic**!

1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD

4 Accelerarea metodelor TD

5 Încheiere

Reamintim: Iteraţia pe legea de control

Iteraţia pe legea de control cu funcţii Q

iniţializează legea de control h_0 arbitrar

repeat la fiecare iteraţie ℓ

1: evaluarea legii de control: găseşte Q^{h_ℓ}

2: îmbunătăţirea legii de control:

$$h_{\ell+1}(x) \leftarrow \arg\max_u Q^{h_\ell}(x, u)$$

until convergenţă la h^*

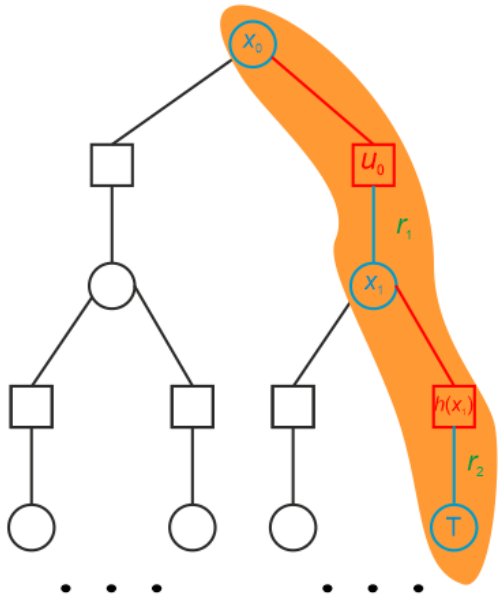
Evaluarea legii de control

Pentru a găsi Q^h:

- Până acum: metode bazate pe model
- Învăţare prin recompensă: **modelul nu este disponibil**
- Învaţă Q^h din date sau prin **interacţiune online cu sistemul**

Evaluarea Monte-Carlo a legii de control

Reamintim: $Q^h(x_0, u_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1}$

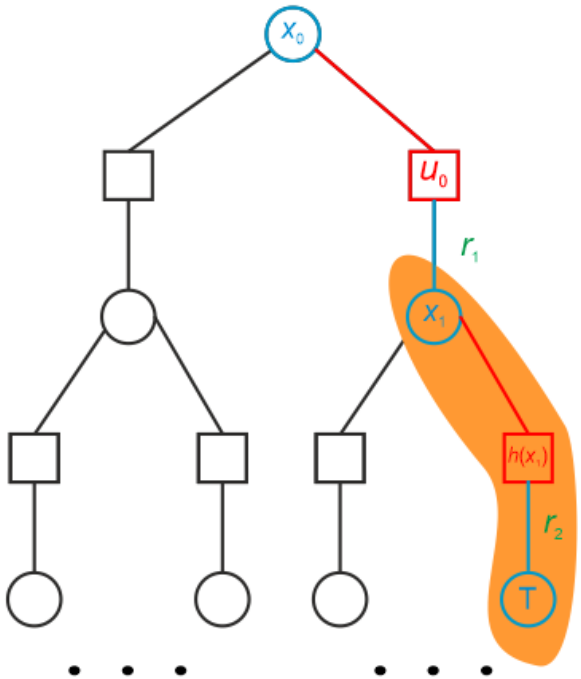


- Traiectorie de la (x_0, u_0) la x_K (**terminal**) utilizând $u_1 = h(x_1)$, $u_2 = h(x_2)$ etc.

$\Rightarrow$ $Q^h(x_0, u_0) =$ returnul de-a lungul traiectoriei:

$$Q^h(x_0, u_0) = \sum_{j=0}^{K-1} \gamma^j r_{j+1}$$

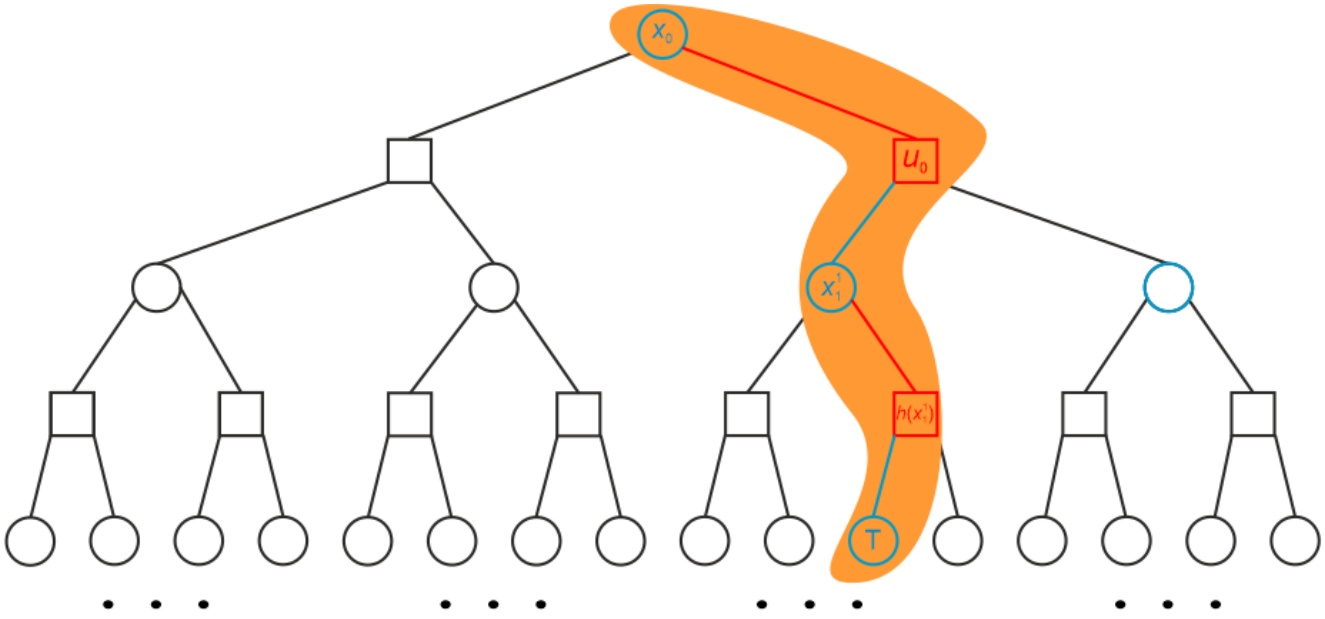
Evaluarea Monte-Carlo a legii de control (cont.)



- În plus, la fiecare pas k:

$$Q^h(x_k, u_k) = \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$$

Evaluarea Monte-Carlo a legii de control: caz stohastic



- N traiectorii (diferite datorita tranziţiilor stohastice)
- Valoarea estimată a funcţiei Q = **media** returnurilor, de exemplu:

$$Q^h(x_0, u_0) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=0}^{K_i - 1} \gamma^j r_{i,j+1}$$

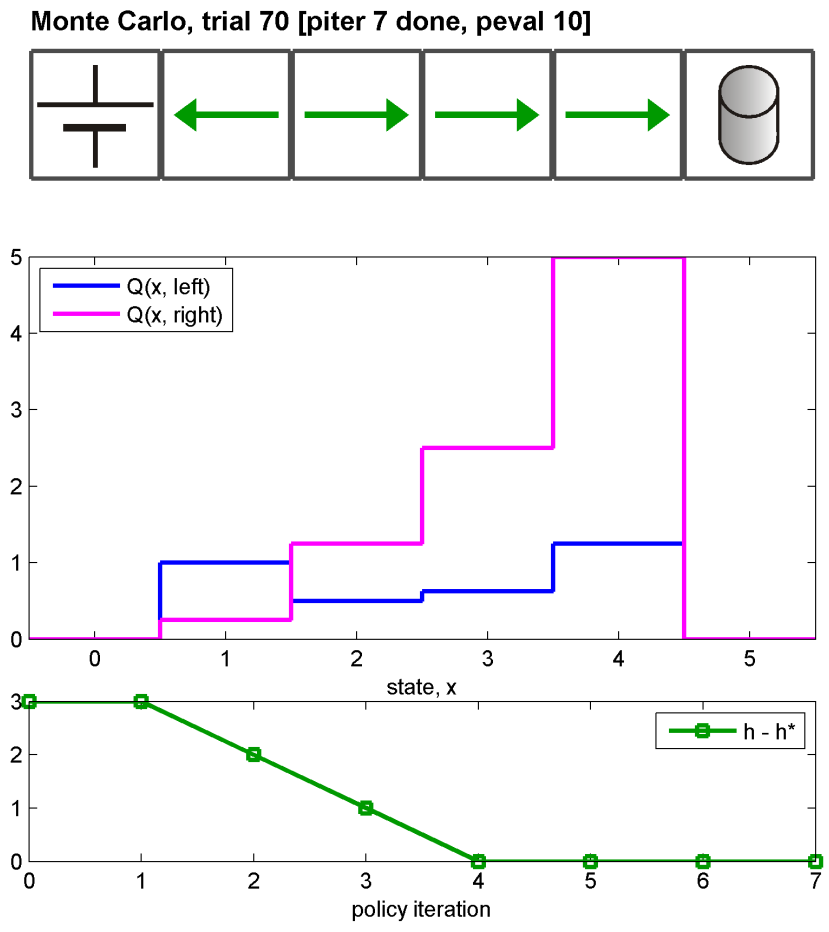
Iteraţia Monte Carlo pe legea de control

Iteraţia Monte Carlo pe legea de control

for fiecare iteraţie ℓ **do**
  efectuează N traiectorii aplicând h_ℓ
  resetează la 0 acumulator $A(x,u)$, counter $C(x,u)$
  for fiecare pas k din fiecare traiectorie i **do**
    $A(x_k,u_k) \leftarrow A(x_k,u_k) + \sum_{j=k}^{K_i-1} \gamma^{j-k} r_{i,j+1}$ (return)
    $C(x_k,u_k) \leftarrow C(x_k,u_k) + 1$
  end for
  $Q^{h_\ell}(x,u) \leftarrow A(x,u)/C(x,u)$
  $h_{\ell+1}(x) \leftarrow \arg\max_u Q^{h_\ell}(x,u)$
end for

De notat: **atingerea stării terminale** trebuie garantată!

Robot menajer: Monte Carlo, demo



1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD

4 Accelerarea metodelor TD

5 Încheiere

Nevoia de explorare

$Q^h(x,u) \leftarrow A(x,u)/\mathbf{C(x,\ u)}$

Cum asigurăm $C(x,u) > 0$ – **informaţie** despre fiecare (x,u)?

1. Stări iniţiale x_0 selectate reprezentativ
2. Acţiuni:
   u_0 reprezentative, câteodată diferite de $h(x_0)$
   şi în plus, posibil:
   u_k reprezentative, câteodată diferite de $h(x_k)$

Dilema explorare-exploatare

- **Explorarea** necesară:
  acţiuni diferite de legea curentă de control
- **Exploatarea** cunoştinţelor curente necesară:
  legea de control trebuie aplicată

Dilema explorare-exploatare
– esenţială în toţi algoritmii de RL

(nu doar în MC)

Explorare-exploatare: strategia ε-greedy

- O soluţie simplă: **ε-greedy**

$$u_k = \begin{cases} h(x_k) = \arg\max_u Q(x_k, u) & \text{cu probabilitatea } (1 - \varepsilon_k) \\ \text{o acţiune aleatoare} & \text{cu probabilitatea } \varepsilon_k \end{cases}$$

- Probabilitatea de explorare $\varepsilon_k \in (0, 1)$ scade de obicei în timp

Strategia softmax

- Selecţia acţiunii:

$$u_k = u \text{ cu proba. } \frac{e^{Q(x_k,u)/\tau_k}}{\sum_{u'} e^{Q(x_k,u')/\tau_k}}$$

  unde $\tau_k > 0$ este **temperatura de explorare**
- Pentru $\tau \to 0$, recuperăm selecţia greedy; $\tau \to \infty$ duce la distribuţie aleatoare uniformă
- Comparativ cu ε-greedy, acţiunile mai bune au o probabilitate mai mare de a fi aplicate chiar şi în explorare

Explorare bazată pe bandiţi

Într-o stare unică, explorarea este modelată ca **bandit cu mai multe braţe**:

- Acţiunea j = braţ cu distribuţie a recompensei ρ_j, valoare aşteptată μ_j
- Cel mai bun braţ (acţiunea optimală) are valoarea aşteptată μ^*
- La pasul k, tragem braţul (încercăm acţiunea) j_k, obţinând $r_k \sim \rho_{j_k}$
- **Obiectiv:** După n încercări, regret minim: $\sum_{k=1}^{n} \mu^* - \mu_{j_k}$

Algoritmul UCB

Un algoritm des utilizat: după n paşi, alegem braţul cu cea mai mare **margine superioară de încredere** (en. upper confidence bound):

$$b(j) = \widehat{\mu}_j + \sqrt{c\frac{\log n}{n_j}}$$

unde:

- $\widehat{\mu}_j$ = media recompenselor observate pentru braţul j până în prezent
- n_j = de câte ori a fost tras braţul j
- c constantă acordabilă, de exemplu $3/2$

Acestea sunt doar câteva metode simple, există multe altele, de exemplu explorare bayesiană, recompense intrinseci, iniţializare optimistă etc.

1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD
- Introducere
- SARSA
- Învăţarea Q

4 Accelerarea metodelor TD

5 Încheiere

Monte-Carlo cu actualizări incrementale

Luăm în considerare eşantionul de return de la pasul k încolo:

$$R_k = \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$$

În loc să facem media acestor eşantioane pentru a obţine Q, efectuăm **actualizări incrementale**:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k [R_k - Q(x_k, u_k)]$$

unde α_k este o mărime a pasului sau **rata de învăţare**

Discuţie

- MC incremental motivat de probleme variabile în timp (eşantioanele recente au ponderi mai mari), dar funcţionează şi în MDPuri invariante în timp
- Nu mai este nevoie să stocăm acumulatoare A şi contoare C, doar direct valorile Q
- Dacă α_k satisface condiţiile de aproximare stohastică:
  1. scade la 0, $\sum_{k=0}^{\infty} \alpha_k^2 = \text{finit}$
  2. dar nu prea repede, $\sum_{k=0}^{\infty} \alpha_k \to \infty$

  metoda converge: lim când # de eşantioane $\to \infty$ = valoarea Q

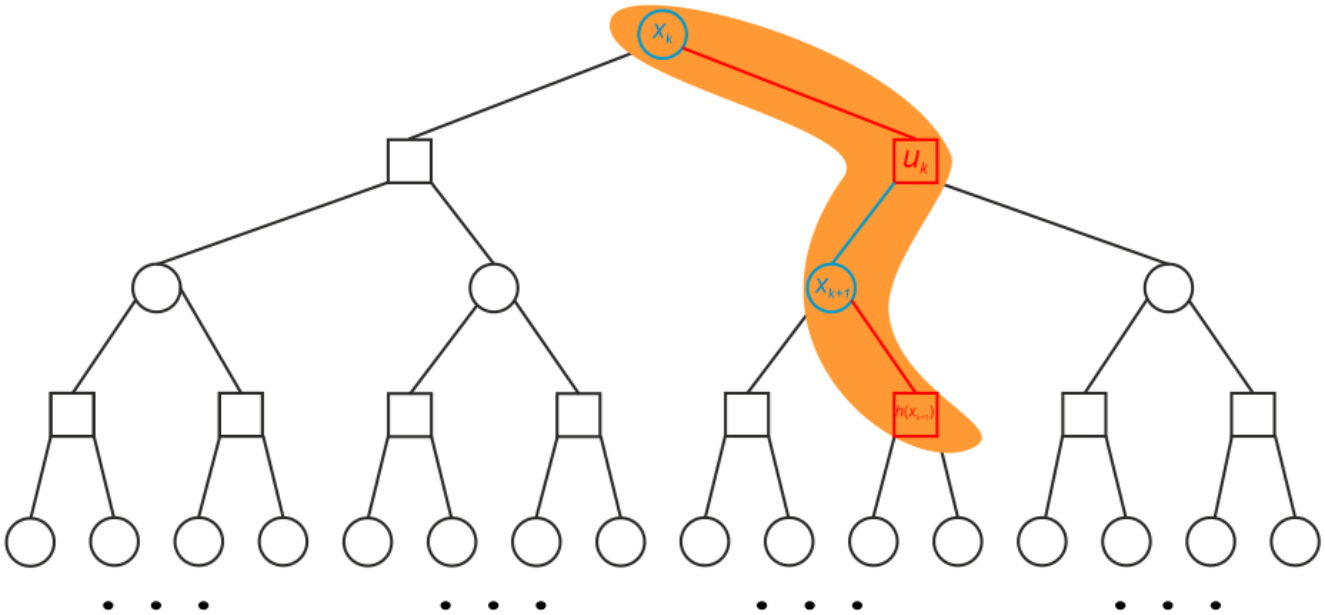
De la Monte-Carlo la diferenţe temporale

Pentru a nu aştepta sfârşitul traiectoriei, folosim ecuaţia Bellman:

$$Q^h(x,u) = \mathrm{E}_{x'}\left\{\tilde{\rho}(x,u,x') + \gamma Q^h(x', h(x'))\right\}$$

pentru a estima returnul:

$$\hat{R}_k = r_{k+1} + \gamma Q(x_{k+1}, h(x_{k+1}))$$



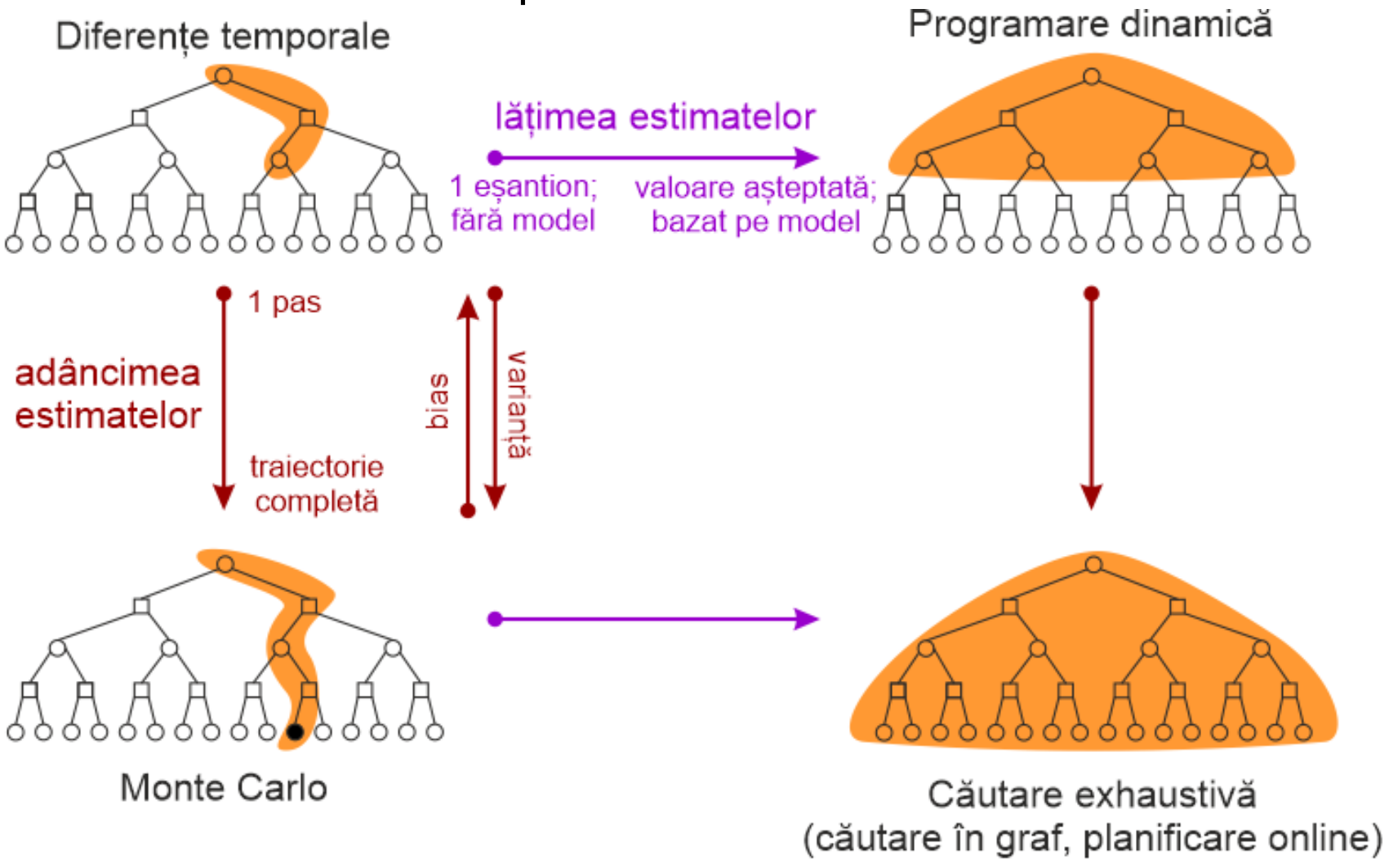
Diferenţe temporale (TD)

În rest, actualizarea rămâne aceeaşi, dar explicităm returnul estimat:

$$\begin{aligned} Q(x_k, u_k) &\leftarrow Q(x_k, u_k) + \alpha_k[\hat{R}_k - Q(x_k, u_k)] \\ &= Q(x_k, u_k) + \alpha_k[r_{k+1} + \gamma Q(x_{k+1}, h(x_{k+1})) - Q(x_k, u_k)] \end{aligned}$$

- $[\ldots]$ este **diferenţa temporală** între două estimări ale $Q(x_k, u_k)$, utilizând informaţii la paşi de timp consecutivi
- Actualizări bazate pe date, fără model (ca în MC): r_{k+1}, x_{k+1} observate de ex. în timpul interacţiunii online
- Actualizările estimează $Q(x_k, u_k)$ utilizând o altă estimare, $Q(x_{k+1}, h(x_{k+1}))$: **bootstrapping**
- Programarea dinamică face, de asemenea, bootstrapping, dar utilizând un model

TD vs. MC vs. DP: Perspectivă unificată



Diferenţe temporale pentru evaluarea h

Diferenţe temporale pentru evaluarea h

for fiecare traiectorie **do**
  iniţializează x_0, alege acţiunea iniţială u_0
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    alege acţiunea **următoare** $u_{k+1} \sim h(x_{k+1})$
    $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$
    $[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$
  until traiectoria terminată
end for

Observaţie: am înlocuit $h(x_k + 1)$ cu u_{k+1}, aleasă cu h

Explorare-exploatare

alege acţiunea următoare $u_{k+1} \sim h(x_{k+1})$

- Informaţii despre $(x, u) \neq (x, h(x))$ necesare
  $\Rightarrow$ **explorare**
- h trebuie urmărită
  $\Rightarrow$ **exploatare**

- Ex. ε-greedy:

$$u_{k+1} = \begin{cases} h(x_{k+1}) & \text{cu prob. } (1 - \varepsilon_{k+1}) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_{k+1} \end{cases}$$

1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD
- Introducere
- SARSA
- Învăţarea Q

4 Accelerarea metodelor TD

5 Încheiere

Îmbunătăţirea optimistă a legii de control

- Legea de control neschimbată pentru N traiectorii

$\Rightarrow$ Algoritmul învaţă încet

- Îmbunătăţirea legii de control după fiecare traiectorie
  = **optimist**
- Vom folosi de asemenea şi explorare ε-greedy

Metoda Monte Carlo optimistă

Metoda Monte Carlo optimistă

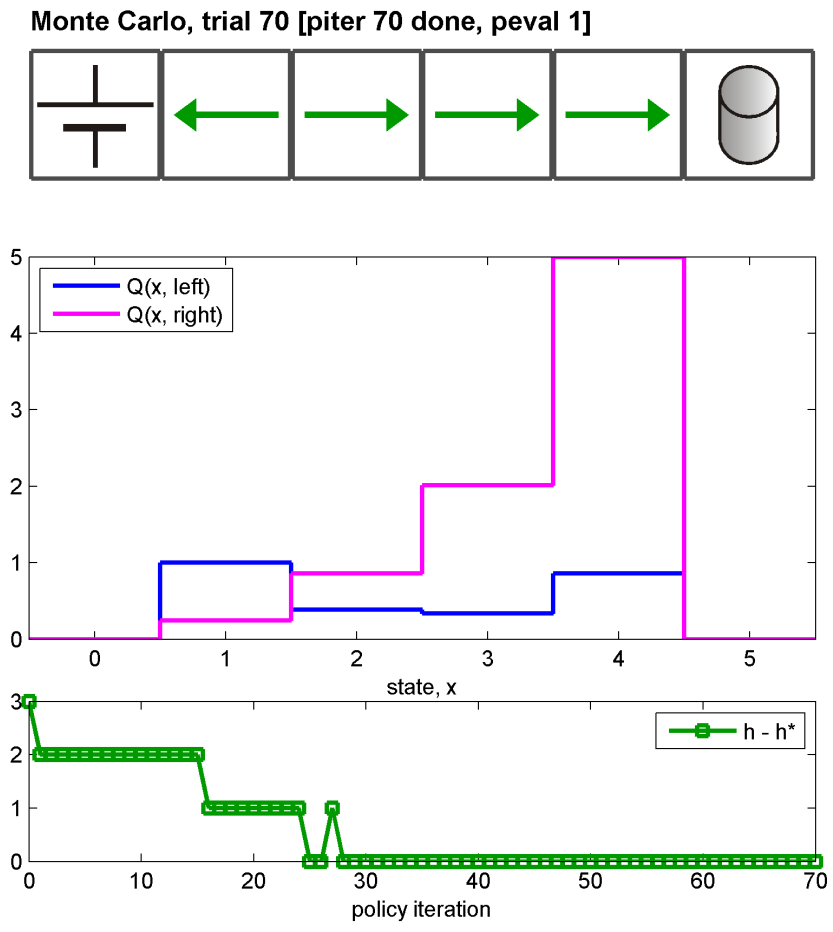
iniţializează la 0 acumulator $A(x,u)$, counter $C(x,u)$
for fiecare traiectorie **do**
  efectuează traiectoria, ex. aplicând ε-greedy:

$$u_k = \begin{cases} \arg\max_u Q(x_k,u) & \text{cu prob. } (1-\varepsilon_k) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_k \end{cases}$$

  for fiecare pas k al fiecărei traiectorii **do**
    $A(x_k,u_k) \leftarrow A(x_k,u_k) + \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$
    $C(x_k,u_k) \leftarrow C(x_k,u_k) + 1$
  end for
  $Q(x,u) \leftarrow A(x,u)/C(x,u)$
end for

- h implicit, greedy în Q
- actualizarea $Q \Rightarrow$ implicit îmbunătăţirea h

Robot menajer: Monte Carlo optimist, demo



Optimism în TD

- Algoritm precedent: h fixată
- Îmbunătăţirea h: cel mai simplu, după fiecare tranziţie

$\Rightarrow$ interpretare: iteraţie pe legea de control **optimistă** la nivel de tranziţie

- h implicit, greedy în Q
  (actualizarea $Q \Rightarrow$ implicit îmbunătăţirea h)

SARSA

SARSA cu ε-greedy

for fiecare traiectorie **do**

iniţializează x_0

$$u_0 = \begin{cases} \arg\max_u Q(x_0, u) & \text{cu prob. } (1 - \varepsilon_0) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_0 \end{cases}$$

repeat la fiecare pas k

aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}

$$u_{k+1} = \begin{cases} \arg\max_u Q(x_{k+1}, u) & \text{cu prob. } (1 - \varepsilon_{k+1}) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_{k+1} \end{cases}$$

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]$$

until traiectoria terminată

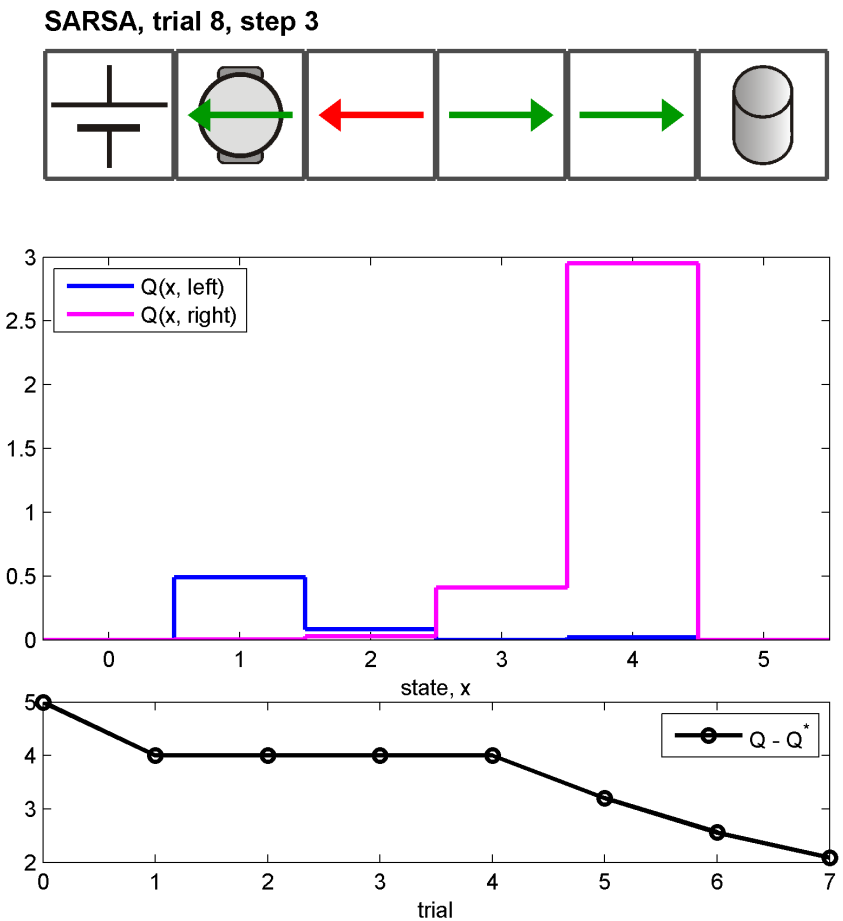
end for

Originile numelui SARSA

$(x_k, u_k, r_{k+1}, x_{k+1}, u_{k+1}) =$
(**S**tare, **A**cţiune, **R**ecompensă, **S**tare, **A**cţiune) = **SARSA**

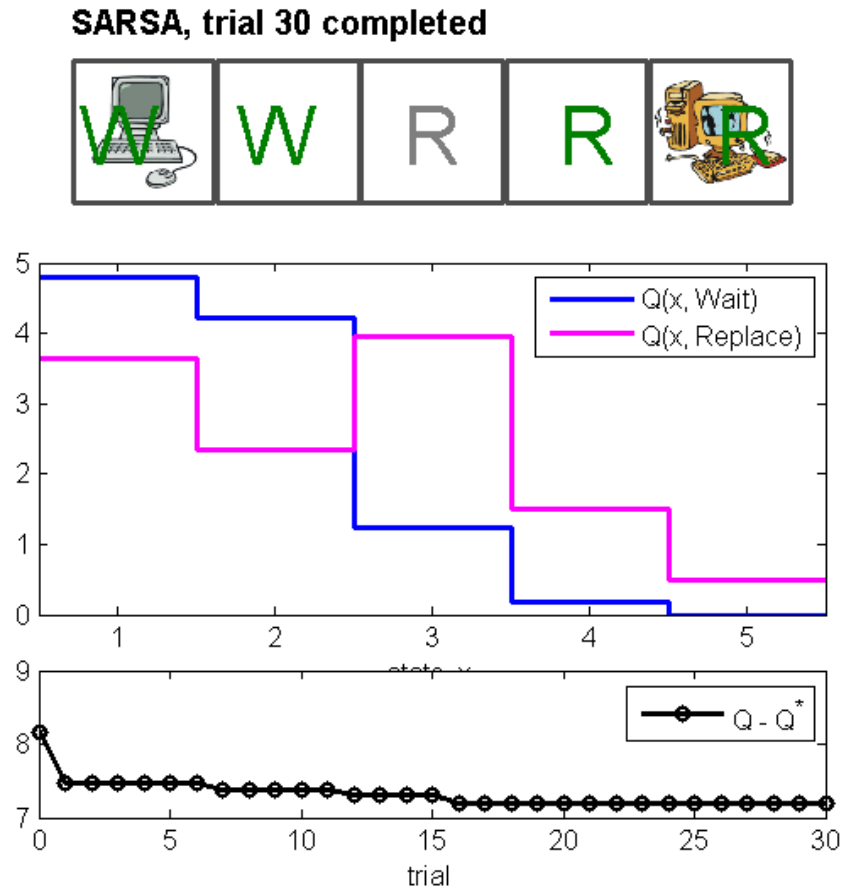
Robot menajer: SARSA, demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$ (constanţi)
$x_0 = 2$ sau 3 (aleator)



Înlocuirea unei maşini: SARSA, demo

Parametri: $\alpha = 0.1$, $\varepsilon = 0.3$ (constanţi), 20 paşi pe traiectorie
$x_0 = 1$



1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD
- Introducere
- SARSA
- Învăţarea Q

4 Accelerarea metodelor TD

5 Încheiere

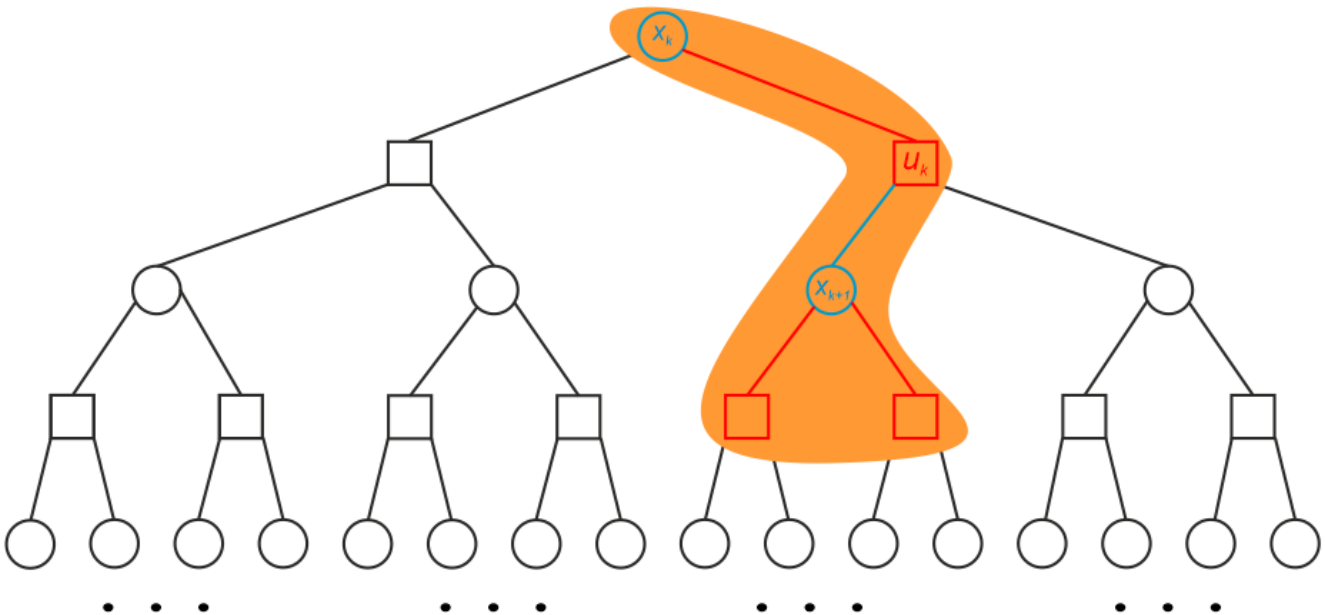
Estimarea Q^* prin bootstrapping

Ecuaţia de optimalitate Bellman:

$$Q^*(x,u) = \mathrm{E}_{x'}\left\{\tilde{\rho}(x,u,x') + \gamma \max_{u'} Q^*(x',u')\right\}$$

conduce la estimatul:

$$\hat{Q}_k = r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$$



Actualizare TD pentru Q^*

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k [\hat{Q}_k - Q(x_k, u_k)]$$
$$[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

Învăţarea Q

Învăţarea Q cu ε-greedy

for fiecare traiectorie **do**

iniţializează x_0

repeat la fiecare pas k

$$u_k = \begin{cases} \arg\max_u Q(x_k, u) & \text{cu prob. } (1 - \varepsilon_k) \\ \text{aleatoare} & \text{cu prob. } \varepsilon_k \end{cases}$$

aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}

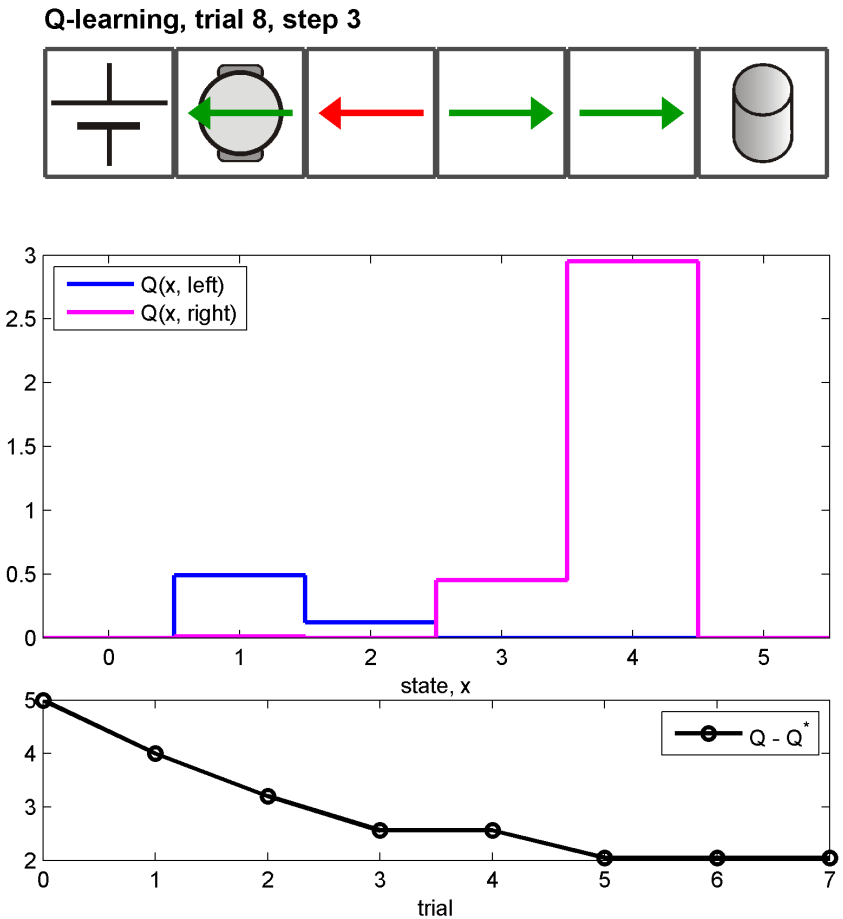
$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot [r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$$

until traiectoria terminată

end for

Robot menajer: învăţarea Q, demo

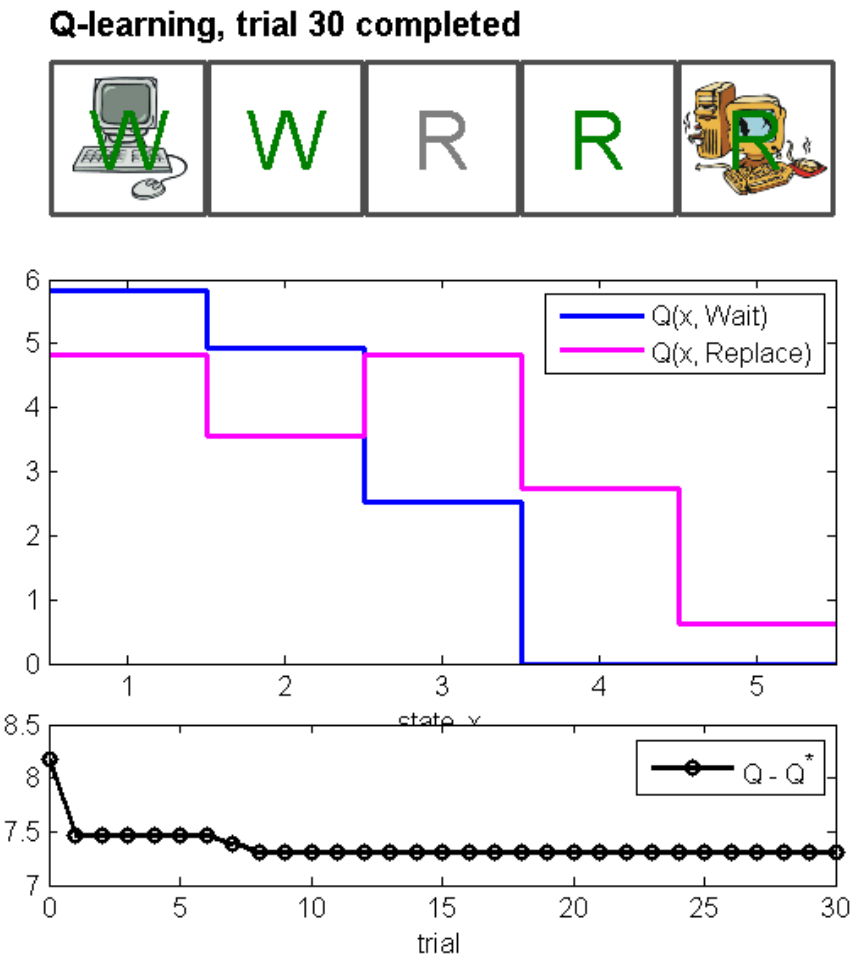
Parameteri – ca şi SARSA: $\alpha = 0.2$, $\varepsilon = 0.3$ (constanţi)
$x_0 = 2$ sau 3 (aleator)



Înlocuirea unei maşini: învăţarea Q, demo

Parametri: $\alpha = 0.1$, $\varepsilon = 0.3$ (constanţi), 20 paşi pe traiectorie

$x_0 = 1$



Convergenţă

Condiţii de convergenţă la Q^*:

1. Toate perechile (x, u) continuă să fie actualizate: asigurat de **explorare**, ex. ε-greedy
2. Condiţiile de aproximare stohastică pentru α_k (scade spre 0, $\sum_{k=0}^{\infty} \alpha_k^2 =$ finit, dar nu prea repede, $\sum_{k=0}^{\infty} \alpha_k \to \infty$)

În plus, pentru SARSA:

3. Legea de control trebuie să devină greedy la infinit ex. $\lim_{k\to\infty} \varepsilon_k = 0$

On-policy / off-policy

SARSA: **on-policy**

- Estimează permanent funcţia Q a legii de control curente

Învăţarea Q: **off-policy**

- Indiferent de legea de control curentă, estimează funcţia Q optimală

Diferenţe temporale: Discuţie

Avantaje

- Simplu de înţeles, implementat
- Complexitate scăzută $\Rightarrow$ execuţie rapidă

SARSA vs. învăţarea Q

- SARSA mai puţin complex decât învăţarea Q (fără max în actualizarea funcţiei Q)

Secvenţele α_k, ε_k **influenţează semnificativ** performanţa

Dezavantaj principal

- Necesită multe date

1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD

4 Accelerarea metodelor TD
- Motivare
- Reluarea experienţei
- Return pe *n* paşi

5 Încheiere

Nevoia de a accelera metodele TD

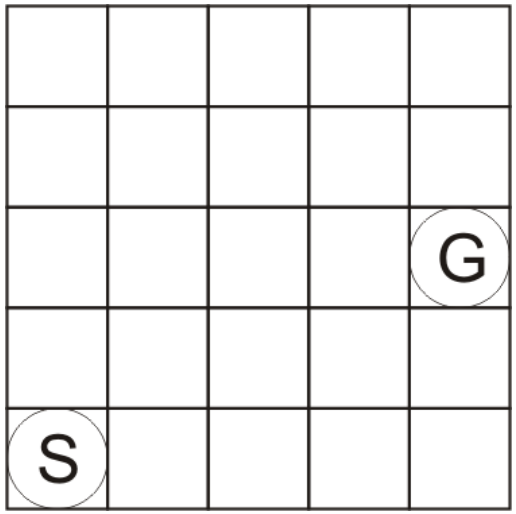
Dezavantaj principal: învaţă încet – **necesită multe date**

În practică, datele costă:

- timp
- profit (performanţă scăzută datorită explorării)
- uzură a sistemului

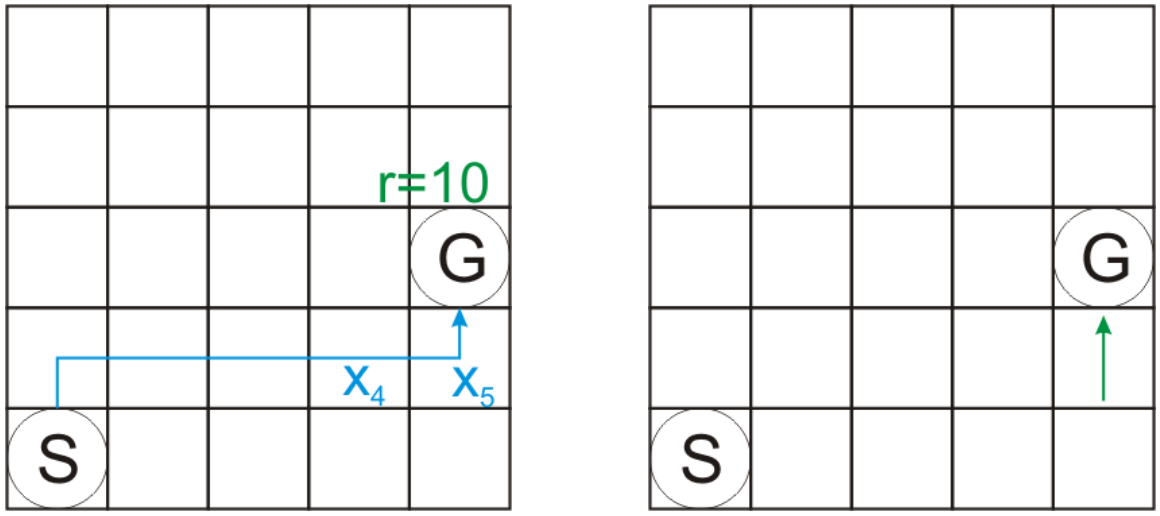
Accelerarea RL = **eficientizarea folosirii datelor**

Exemplu: Navigare 2D



- Navigare într-o lume 2D discretă
  de la **S**tart la **G**oal
- Singura recompensă = 10 la atingerea G (stare terminală)

Exemplu: TD



- Alegem SARSA, $\alpha = 1$; iniţializăm $Q = 0$
- Actualizări de-a lungul traiectoriei din stânga:

$$\dots$$
$$Q(x_4, u_4) = 0 + \gamma \cdot Q(x_5, u_5) = 0$$
$$Q(x_5, u_5) = 10 + \gamma \cdot 0 = 10$$

- O nouă tranziţie de la x_4 la x_5 necesară pentru a propaga informaţia la x_4!

Accelerarea TD: 2 idei

1. Stochează şi **reia experienţa**
2. Return pe *n* paşi

1 Monte Carlo, MC

2 Nevoia de explorare

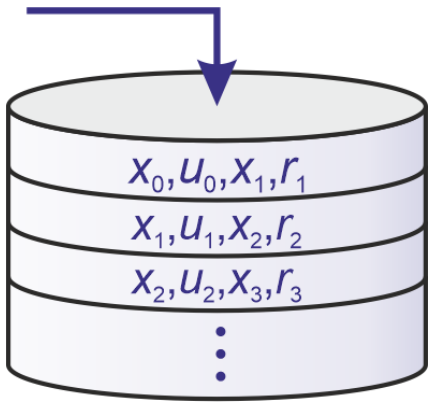
3 Diferenţe temporale, TD

4 Accelerarea metodelor TD
- Motivare
- Reluarea experienţei
- Return pe *n* paşi

5 Încheiere

Reluarea experienţei

- Stochează fiecare tranziţie $(x_k, u_k, x_{k+1}, r_{k+1})$ (pentru SARSA, şi u_{k+1}) într-o bază de date



- La fiecare pas, **reia** n tranziţii din baza de date (pe lângă actualizările normale)

Învăţarea Q cu reluarea experienţei

Învăţarea Q cu reluarea experienţei

for fiecare traiectorie **do**
  iniţializează x_0
  repeat la fiecare pas k
    aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
    $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha_k \cdot$
      $[r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)]$
    adaugă $(x_k, u_k, x_{k+1}, r_{k+1})$ la baza de date ReiaExperienţa
  until traiectoria terminată
end for

Procedura ReiaExperienţa

ReiaExperienţa

loop de n ori

preia o tranziţie (x, u, x', r) din baza de date

$$Q(x,u) \leftarrow Q(x,u) + \alpha \cdot [r + \gamma \max_{u'} Q(x',u') - Q(x,u)]$$

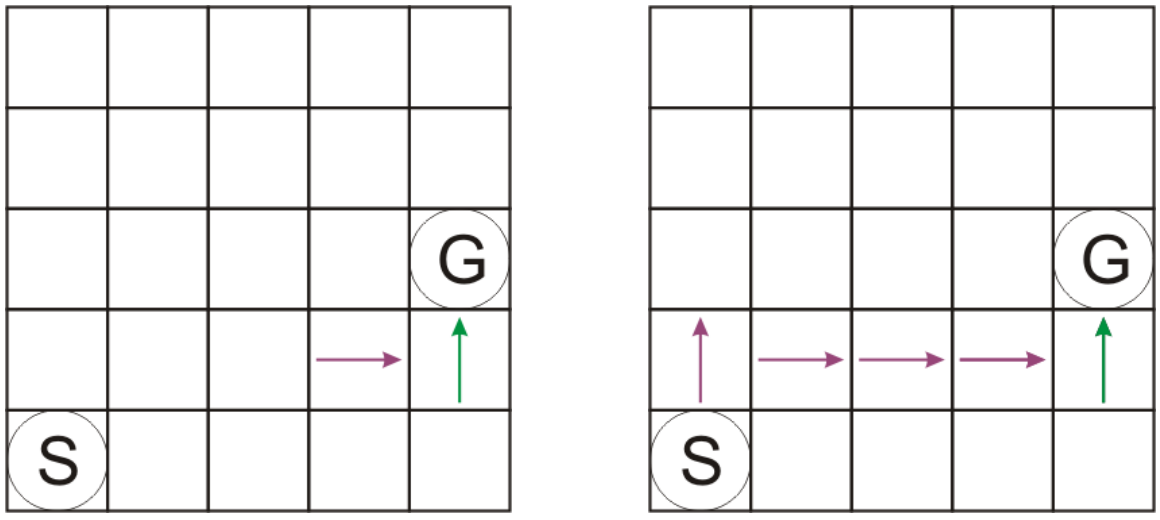
end loop

Direcţia de reluare

Ordinea de reluare a tranziţiilor:

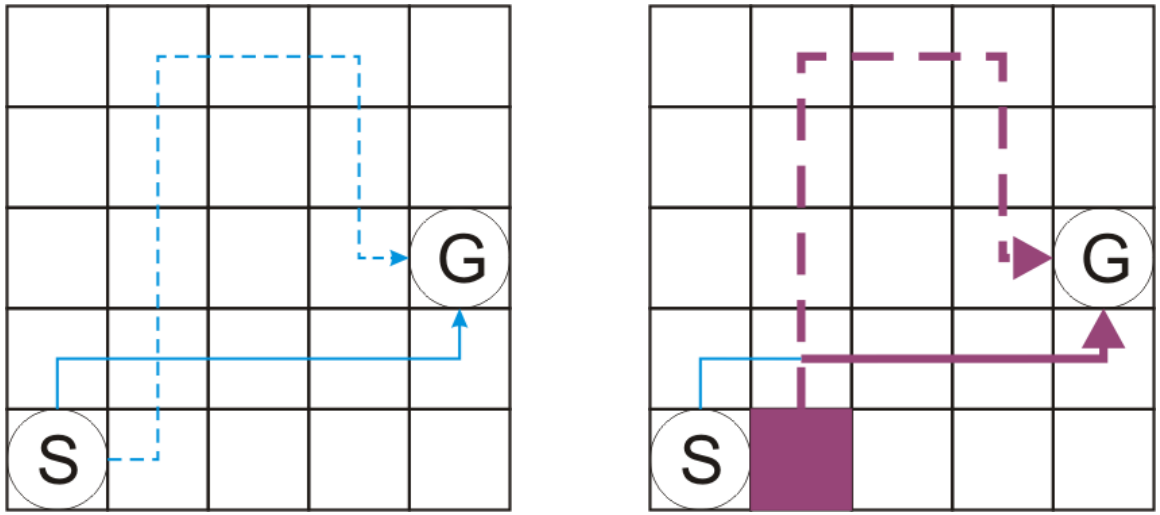
1. Înainte
2. Înapoi
3. Arbitrar

Exemplu: Influenţa direcţiei de reluare



- Verde: actualizările normale, mov: reluarea experienţei
- Stânga: reluare înainte; dreapta: reluare înapoi
- **Reluarea înapoi** în general preferabilă

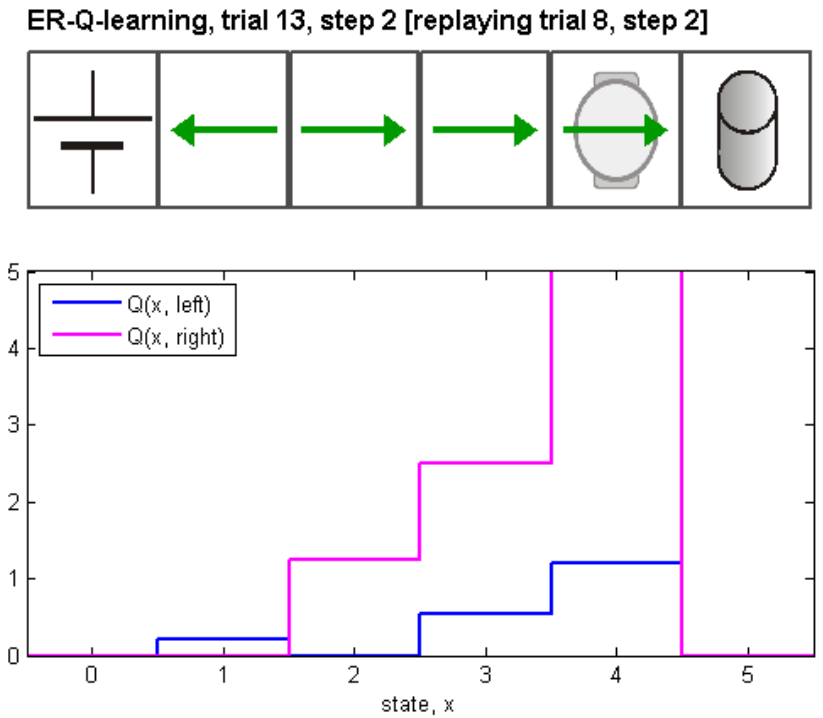
Exemplu: Agregarea informaţiei



- Reluarea experienţei **agregă informaţie** din mai multe traiectorii
- Căsuţa indicată profită de informaţii de-a lungul ambelor traiectorii

Robot menajer: Q cu reluarea experienţei, demo

Parametri: $\alpha = 0.2$, $\varepsilon = 0.3$, $n = 5$, direcţia înapoi
$x_0 = 2$ or 3 (aleator)



1 Monte Carlo, MC

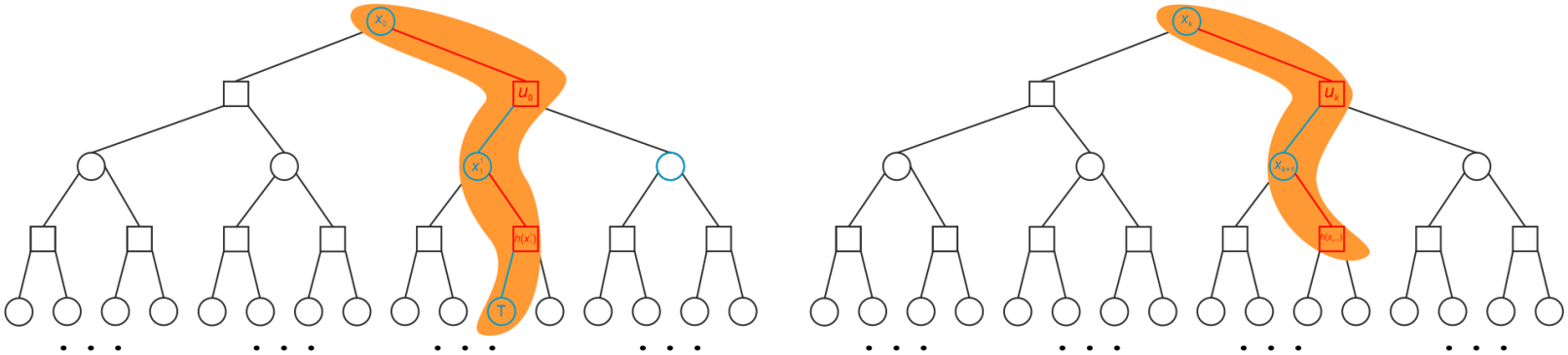
2 Nevoia de explorare

3 Diferenţe temporale, TD

4 Accelerarea metodelor TD
- Motivare
- Reluarea experienţei
- Return pe *n* paşi

5 Încheiere

Reamintire: estimările returnului în MC şi TD

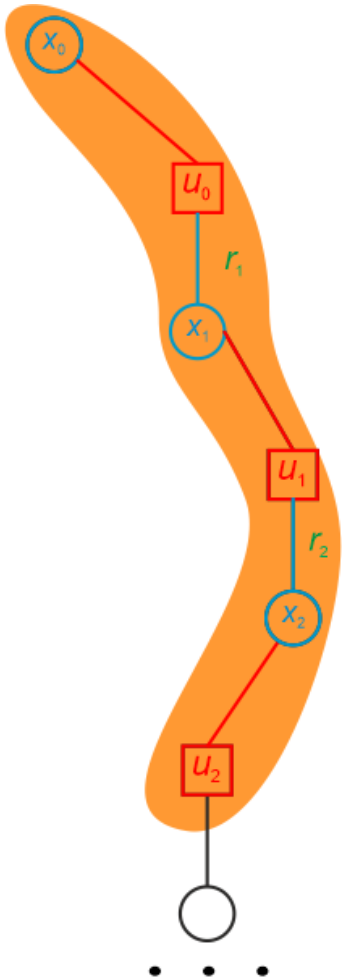


$$R_k = \sum_{j=k}^{K-1} \gamma^{j-k} r_{j+1}$$

$$\hat{R}_k = r_{k+1} + \gamma Q(x_{k+1}, h(x_{k+1}))$$

Există o variantă intermediară?

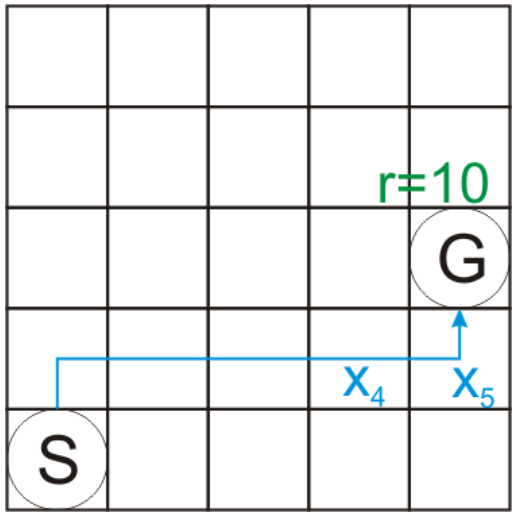
Compromis: return pe n paşi

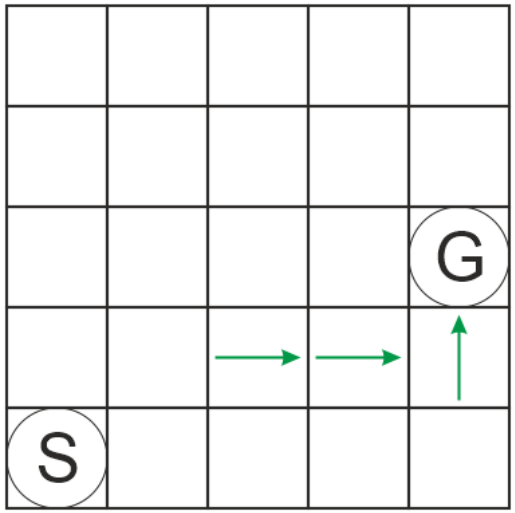


SARSA (on-policy):

$$\hat{R}_k = r_{k+1} + \gamma r_{k+2} + \ldots + \gamma^{n-1} r_{k+n} + \gamma^n Q(x_{k+n}, u_{k+n})$$

Exemplu: Efectul returnului pe n paşi



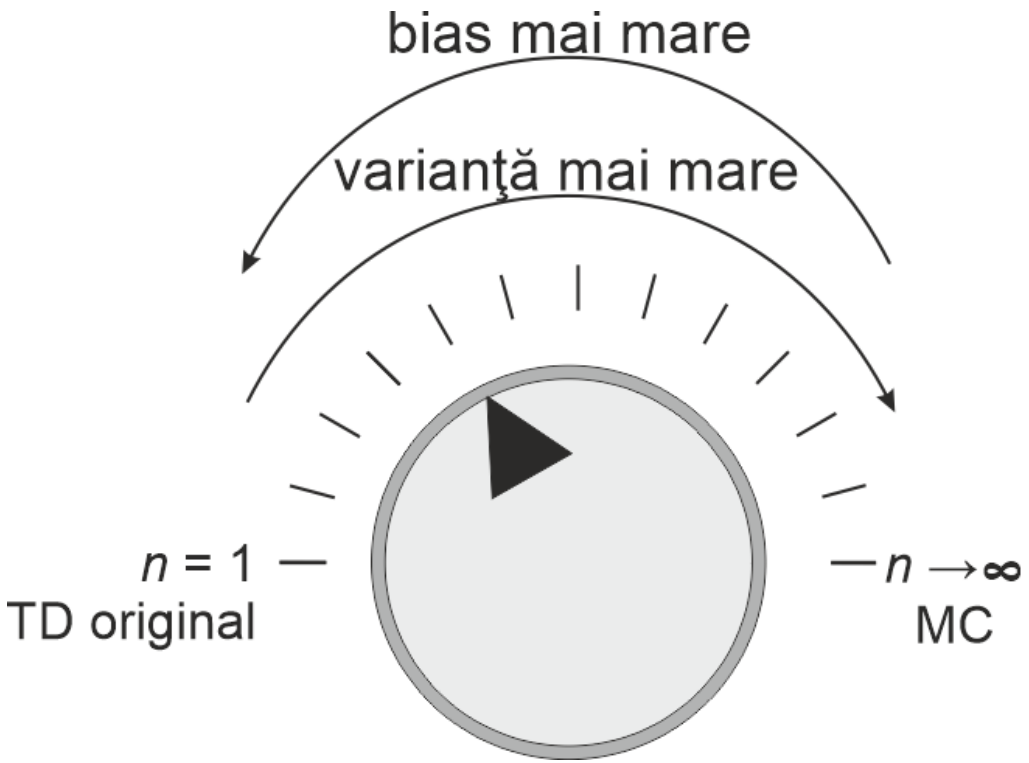


Pentru $n = 3$:

$$Q(x_5, u_5) = 10 + 0 \quad \text{(terminal)}$$
$$Q(x_4, u_4) = 0 + \gamma 10 + 0 \quad \text{(terminal)}$$
$$Q(x_3, u_3) = 0 + \gamma 0 + \gamma^2 10 + 0 \quad \text{(terminal)}$$
$$Q(x_2, u_2) = 0 + \gamma 0 + \gamma^2 0 + \gamma^3 0 \quad \text{(bootstrap)}$$
$$\dots$$

TD versus MC

- $n = 1$ recuperează TD, $n \to \infty$ recuperează MC
- Valorile intermediare combină TD şi MC, conducând la un compromis reglabil între bias şi varianţă



1 Monte Carlo, MC

2 Nevoia de explorare

3 Diferenţe temporale, TD

4 Accelerarea metodelor TD

5 Încheiere

Recapitulare: Metode în Partea III

Metode Monte Carlo, MC:

- Iteraţia Monte Carlo pe legea de control
- MC cu actualizări incrementale

Dilema explorare-exploatare:

- ε-greedy, folosit uzual
- Există multe alte soluţii, de ex. UCB

Diferenţe temporale, TD:

- TD pentru evaluarea legii de control
- Îmbunătăţirea optimistă a legii de control
- SARSA
- Învăţarea Q

Accelerarea TD:

- Reluarea experienţei
- Return pe *n* paşi

Terminologie engleză

metode Monte Carlo	=	Monte Carlo methods
dilema explorare-exploatare	=	exploration-exploitation dilemma
diferenţe temporale	=	temporal differences, TD
învăţarea Q	=	Q-learning
SARSA	=	SARSA
rata de învăţare	=	learning rate
reluarea experienţei	=	experience replay
return pe n paşi	=	n-step return

Exerciţii

1. Descreşterea exponentială a ratei de învăţare, $\alpha_k = \alpha^k$, cu $\alpha \in (0, 1)$ o constantă, satisface condiţiile de aproximare stohastică?
2. Este garantată convergenţa învăţării Q când $\varepsilon_k = \varepsilon$, o constantă în $(0, 1)$? Dar pentru SARSA? Ce se întâmplă dacă folosim o descreştere exponentială $\varepsilon_k = \varepsilon^k$?
3. Ar avea sens un algoritm Monte-Carlo care îmbunătăţeşte legea de control după fiecare tranziţie (ca în TD)?
4. Ar propaga învăţarea Q (fără return pe *n* paşi, deoarece este mai complicat în cazul off-policy) informaţia mai rapid decât SARSA în exemplul traiectoriei pe grid?

Exerciţii (cont.)

5. Presupunând că avem acces la un model **doar pentru îmbunătăţirea legii de control**, furnizaţi alternative bazate pe funcţia V pentru toţi algoritmii din această parte a cursului. Urmaţi aceeaşi ordine ca pentru funcţiile Q:
   - Estimări Monte-Carlo, bazate pe mediere şi incrementale
   - Estimări şi actualizări cu bootstrap
   - Evaluarea legii de control, SARSA şi învăţarea Q

   Nu uitaţi să desenaţi arbori şi să încercuiţi nodurile relevante, acest lucru vă va ajuta să vizualizaţi metodele.