Introducere în Reinforcement Learning

Cursul #2

2025 – 2026 (Sem. I)
Drd. Ștefan Iordache
Conf. Dr. Ciprian Păduraru
Facultatea de Matematică și Informatică
Universitatea din București

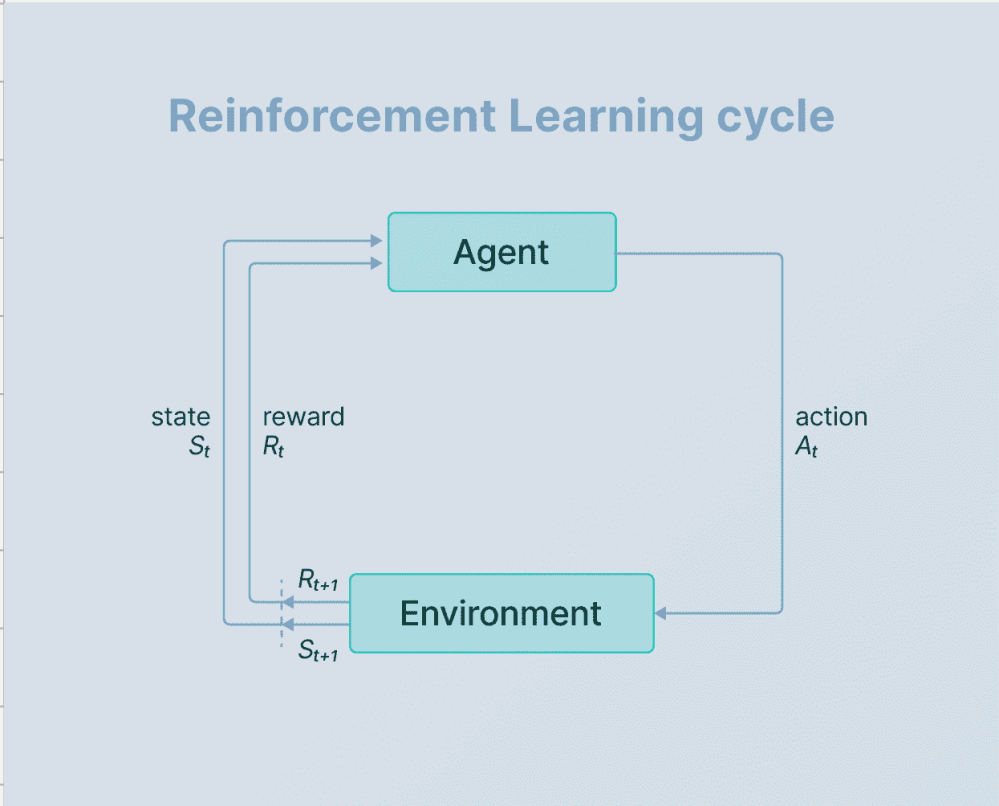
Cuprins

01

Concepte generale

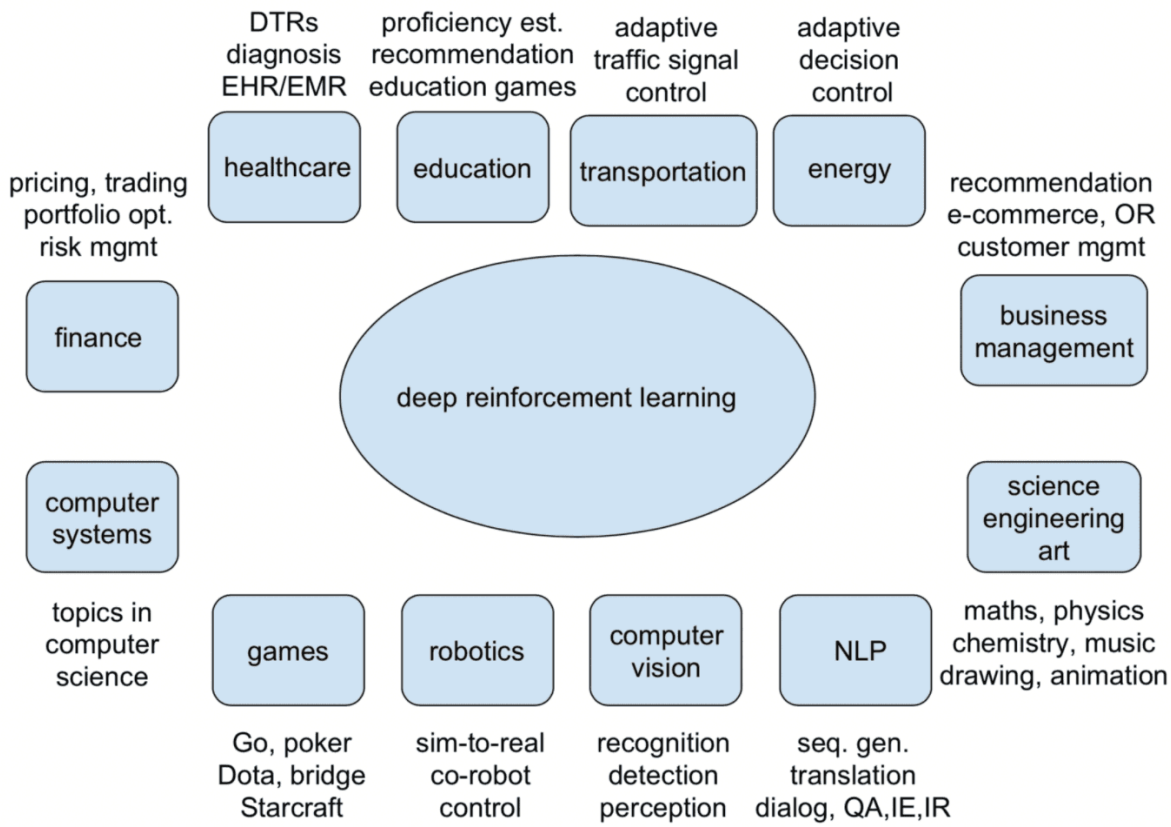
Primii pași în RL

Diagrama generică - RL



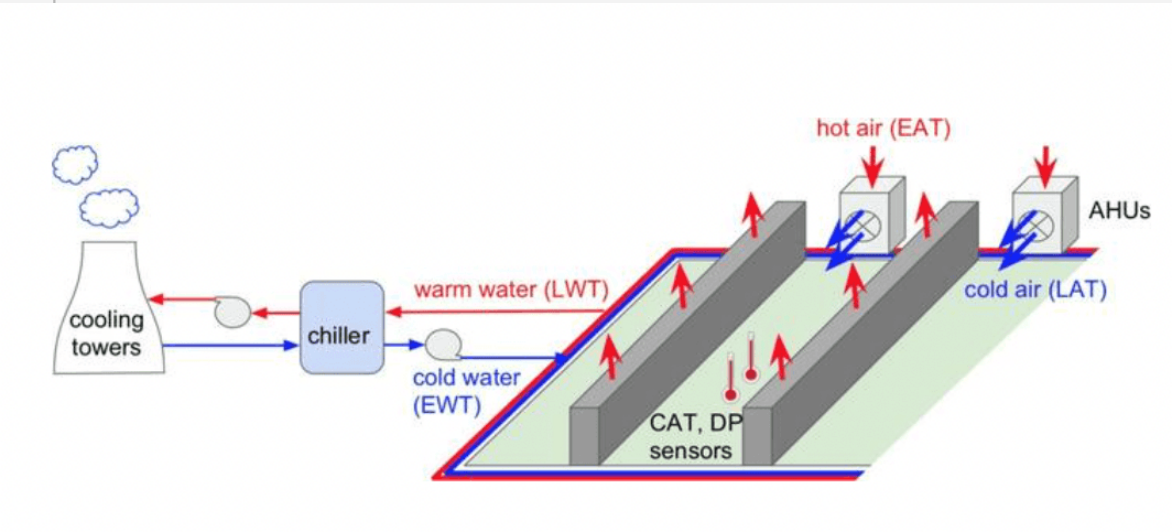
- **Scopul**: dezvoltarea unui agent capabil să aleagă “cele mai bune acțiuni” – strategie optimă pentru maximizare

- Necesită ***balansarea*** recompenselor pe *termen scurt și lung*.



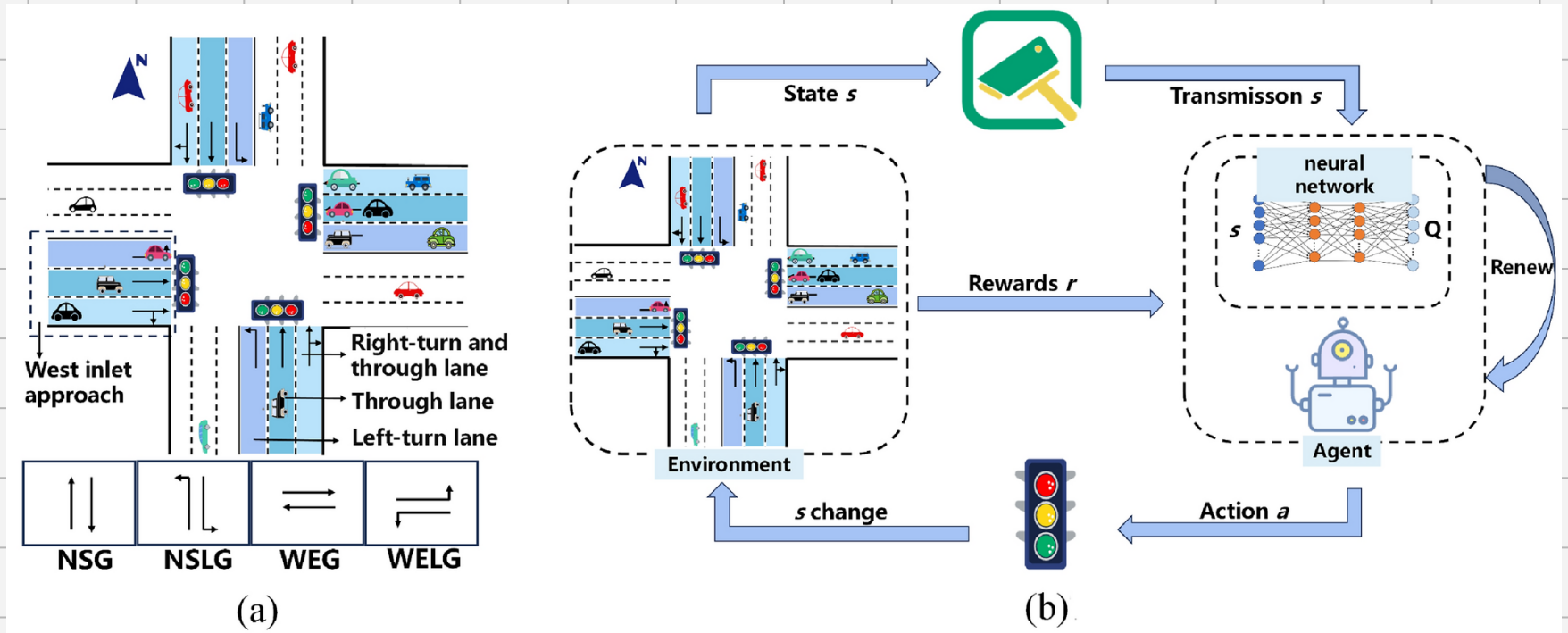
Yuxi Li, Deep Reinforcement Learning, arXiv, 2018

RL – Scenariul #1 – Energie

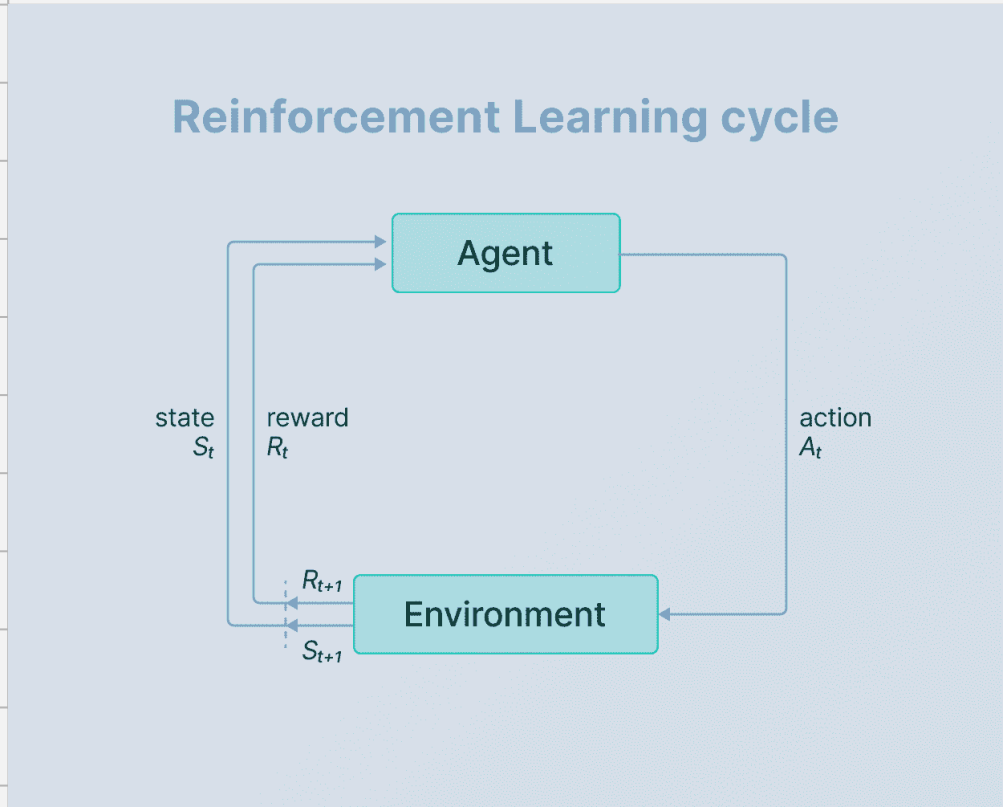


- **Scopul**: răcirea centrelor de date
- **Fun fact:** Google a folosit cu success intelligența artificială pentru a reduce energia necesară răcirii centrelor de date cu 40%!!!

RL – Scenariul #2 – Trafic



Istoricul observațiilor



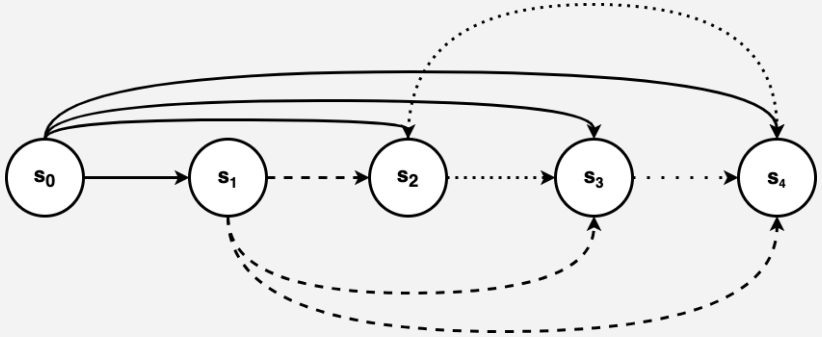
Istoricul la un moment de timp determinat, t:

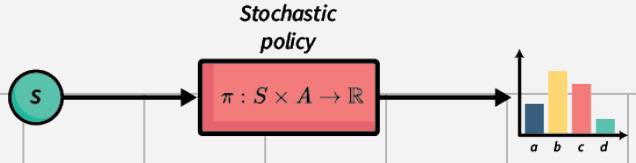
$$h_t = \{a_1, o_1, r_1, \ldots, a_t, o_t, r_t\}$$

- **Acțiunile sunt alese în baza istoricului. Cum?**

În baza unei funcții $\mathbf{s_t = f(h_t)}$

Procese Stochastice





Dinamica unui astfel de proces:

$$Pr(s_t \mid s_{t-1}, s_{t-2}, \ldots, s_0)$$

- **Setul de stări este notat cu S.**
- **Problema: orizont infinit al funcției Pr.**

Care este soluția???

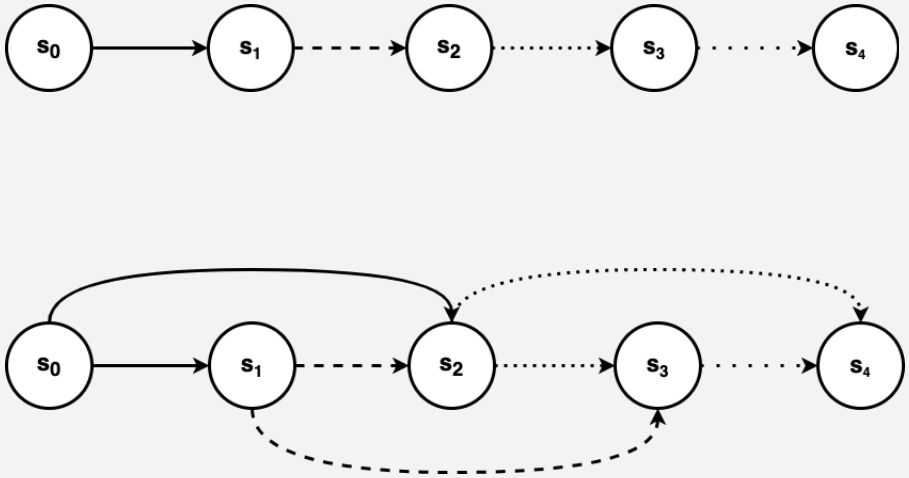
- **Procese Markov**

02

Procese Markov

Baza algoritmilor noștri!

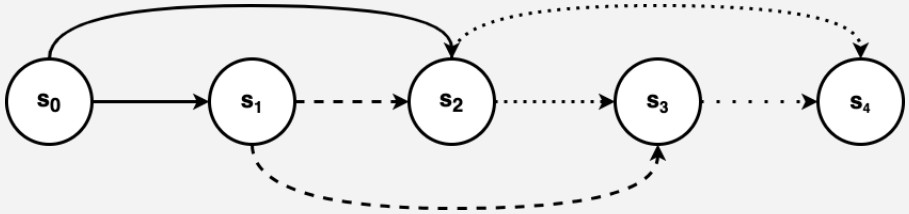
Procese Markov – Definiție



Procesele Markov sunt procese stochastice în care viitorul depinde doar de present.

Mai precis, un proces Markov îndeplinește proprietatea Markov doar dacă probabilitatea de a ajunge într-o stare viitoare depinde exclusive de starea curentă.

Procese Markov - Ordine



Starea curentă depinde doar de un set finit de stări din trecut.

- **Procese Markov de ordinul I**

$$Pr(s_t \mid s_{t-1}, \ldots, s_0) = Pr(s_t \mid s_{t-1})$$

- **Procese Markov de ordinul II**

$$Pr(s_t \mid s_{t-1}, \ldots, s_0) = Pr(s_t \mid s_{t-1}, s_{t-2})$$

Procese Markov – Staționaritate

În mod implicit, un proces Markov se referă la cel de ordinul I.

$$Pr(s_t \mid s_{t-1}, \ldots, s_0) = Pr(s_t \mid s_{t-1}) \ \forall t$$

- Extindem reprezentarea de mai sus către termenul de ***proces staționar:***

$$Pr(s_t \mid s_{t-1}) = Pr(s_{t'} \mid s_{t'-1}) \ \forall t'$$

Avantajul unui proces staționar:

reprezentarea simplă! $Pr(s' \mid s)$

Întrebări naturale…

- **Cum reprezentăm stările? Cât de multe caracteristici adăugăm?**
  - Răspuns: Până când procesul poate fi considerat Markovian, respectiv staționar.
- **Există posibilitatea să adăugăm prea multe componente unei stări?**
  - Răspuns: Da! Adăugarea de componente va crește complexitatea calculelor, implicit necesarul computațional.
  - Soluția: Căutam cel mai mic subset de caracteristici care descrie procesul complet procesul Markovian!
- **Ce utilizăm în practică?**
  - Răspuns: Cel mai frecvent vom utiliza următoarea presupunere -> $s_t = o_t$

Despre decizii

- **Considerăm faptul că simplele predicții sunt inutile. De ce???**
  - Răspuns: Dorim să obținem informații care vor influența alegerile viitoare, nu simple predicții utile unui singur moment.
- **Astfel, sarcina noastră constă în conceperea unor algoritmi capabili să furnizeze decizii!**
- **Dar, cum influențăm deciziile? Cum construim acest algoritm?**
  - Răspuns: ***Procese Decizionale Markov!***

Procese Decizionale Markov (MDP)

Un **Proces Decizional Markov (MDP)** este o formulare matematică a unui mediu în care un agent interacționează în mod secvențial, luând decizii care influențează sterile viitoare și recompensele primite.

$$\text{MDP} = (S, A, P, R, \gamma)$$

- **S** — setul de stări posibile;
- **A** — setul de acțiuni disponibile;
- **P(s'|s, a)** — probabilitatea de a ajunge în starea `s'` după ce agentul alege acțiunea `a` în starea `s`;
- **R(s, a, s')** — recompensa primită pentru tranziția `(s → s')`;
- **γ** — factorul de discount (0 ≤ γ ≤ 1), care controlează importanța viitorului.

03

Algoritmi RL

Începe distracția!

Algoritmi RL

Cuprind una sau mai multe dintre următoarele componente:

- **Model**
- **Politică (Policy)**
- **Value Function**

Modelul

- **Este reprezentarea agentului pentru felul în care mediul se va schimba în urma unei anumite acțiuni.**

- **Tranzițiile (felul în care agentul "prezice" următoarea stare):**

$$p\left(s_{t+1} = s' \middle| s_t = s, a_t = a\right)$$

- **Metodologia de predicție a recompenselor:**

$$r\left(s_t = s, a_t = a\right) = E[r_t \mid s_t = s,\ a_t = a]$$

Politica (Policy) - π

- **O politică determină felul în care agentul alege acțiunile pe care le execută. Este o funcție de forma**

$\pi : S \rightarrow A$

- **Politici deterministe:** $\pi(s) = a$

- **Politici stochastice:** $\pi(a \mid s) = Pr(a_t = a \mid s_t = s)$

Value Function- V^{π} / V(s)

- **Reprezintă suma recompenselor (cu discount), sub o anumită politică aplicată.**

$$V^{\pi}(s_t = s) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots \mid s_t = s]$$

Formal:

$$V^{\pi}(s) = E_{\pi}\left[\sum_i \gamma^i, r(s_i, a_i, s_{i+1}) \mid s_0 = s a_i = \pi(s_i)\right]$$

- **Factorul de discount (γ) va stabili importanța recompensei imediate și a celor viitoare.**
- **Metoda poate fi utilizată pentru a decide calitatea anumitor stări și acțiuni, ulterior stabilind o metodă de comparație între diverse politici.**

Value Function- V^{π} / V(s)

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s) V(s')}_{\text{Discounted sum of future rewards}}$$

Cum calculăm V^{π} / V(s)?

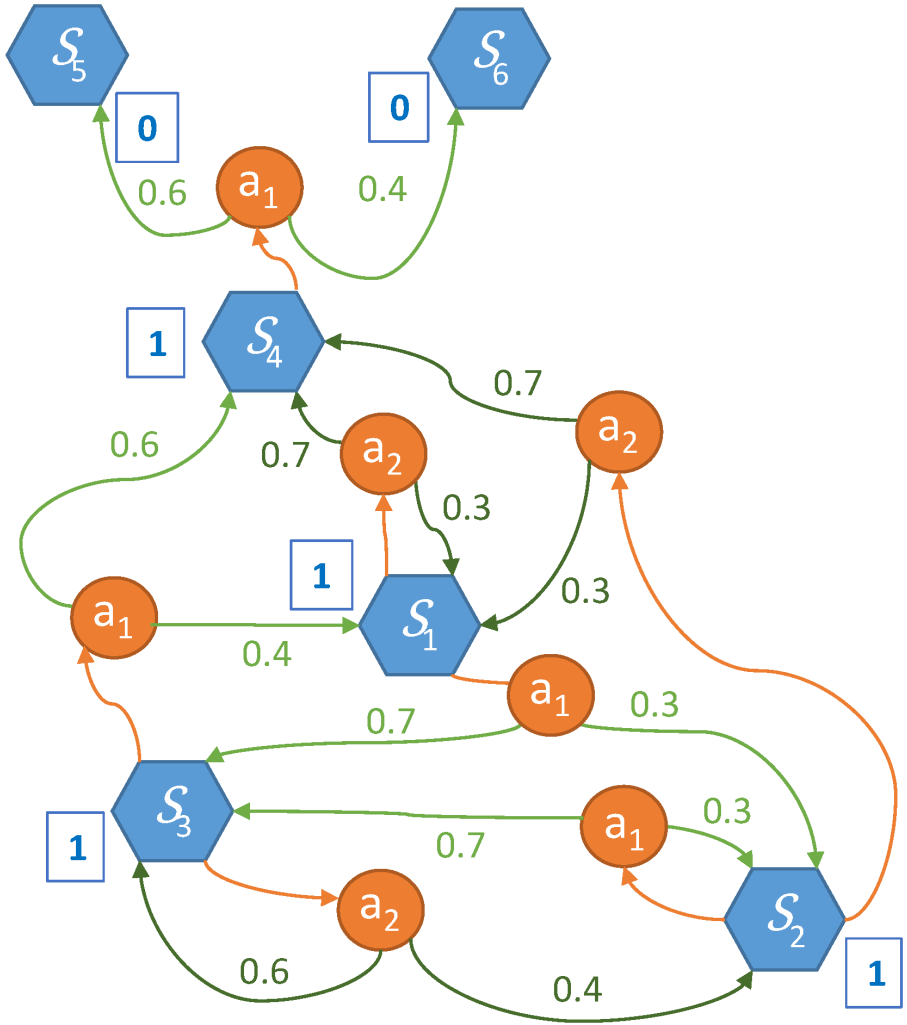
$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

$$\begin{aligned} V &= R + \gamma PV \\ V - \gamma PV &= R \\ (I - \gamma P)V &= R \\ V &= (I - \gamma P)^{-1} R \end{aligned}$$

Complexitate $O(N3)$ –

datorată calculului inversei

EXEMPLIFICARE



$\pi(S_i) = a_1$

	S_1	S_2	S_3	S_4	S_5	S_6
S_1	0	0.3	0.7	0	0	0
S_2	0	0.3	0.7	0	0	0
S_3	0.4	0	0	0.6	0	0
S_4	0	0	0	0	0.6	0.4
S_5	0	0	0	0	0	0
S_6	0	0	0	0	0	0

$\pi(S_i) = a_2$

	0.3	0	0	0.7	0	0
	0.3	0	0	0.7	0	0
	0	0.4	0.6	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

$$\begin{pmatrix} V(S_1) \\ V(S_2) \\ V(S_3) \\ V(S_4) \\ V(S_5) \\ V(S_6) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0.4 & 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V(S_1) \\ V(S_2) \\ V(S_3) \\ V(S_4) \\ V(S_5) \\ V(S_6) \end{pmatrix},$$

$$\begin{pmatrix} -1 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & -0.7 & 0.7 & 0 & 0 & 0 \\ 0.4 & 0 & -1 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V(S_1) \\ V(S_2) \\ V(S_3) \\ V(S_4) \\ V(S_5) \\ V(S_6) \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 0 \\ 0 \end{pmatrix}.$$

$$\begin{pmatrix} V(S_1) \\ V(S_2) \\ V(S_3) \\ V(S_4) \\ V(S_5) \\ V(S_6) \end{pmatrix} = \begin{pmatrix} 5.05 \\ 5.05 \\ 3.62 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Ecuația Bellman – V* (optim)

$$V(s) = \overbrace{\max_{a \in A(s)}}^{\text{best action from } s} \overbrace{\underbrace{\sum_{s' \in S}}_{\text{for every state}} P_a(s' \mid s)\left[\underbrace{r(s,a,s')}_{\text{immediate reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{V(s')}_{\text{value of } s'} \right]}^{\text{expected reward of executing action } a \text{ in state } s}$$

Thanks!

Este timpul pentru întrebări!!!

Acum…sau pe email:

stefan.iordache10@s.unibuc.ro