

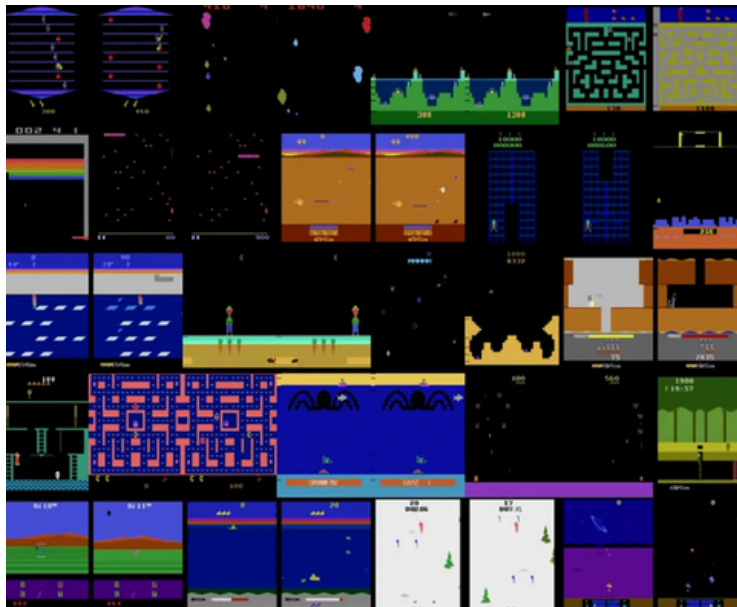
# Control prin învățare

Master ICAF, An 1 Sem 2

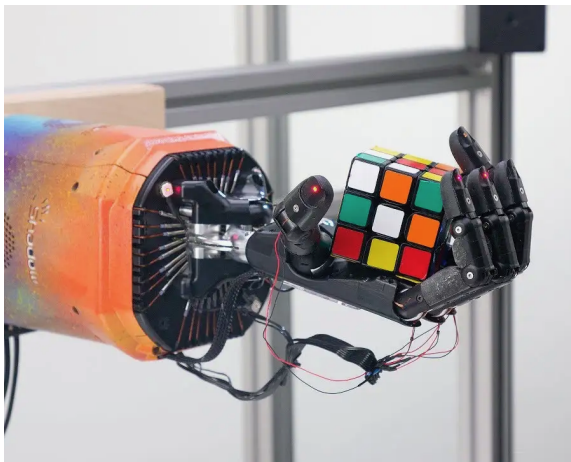
Lucian Bușoniu, Ștefan Pîrje



# Atari cu DQN (DeepMind)

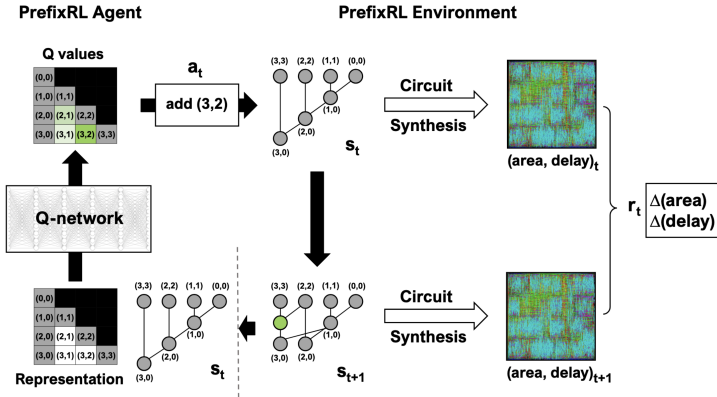


## RL pentru manipularea unui cub Rubik (OpenAI)

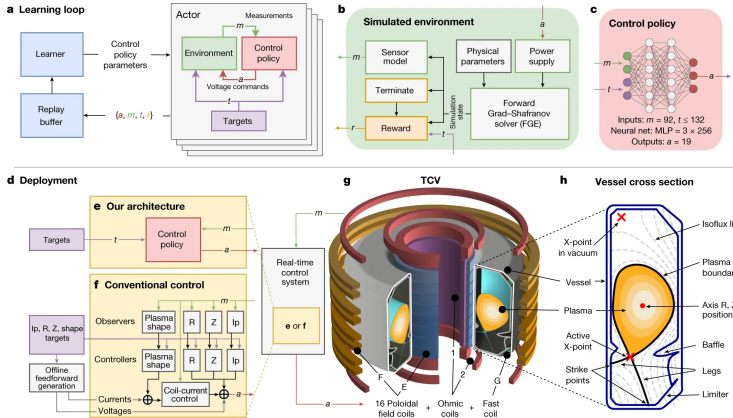


Învăță controlul de finețe al unui număr mare de actuatori, chiar și în prezența perturbațiilor externe.

# RL pentru proiectarea circuitelor digitale (Nvidia / Google)

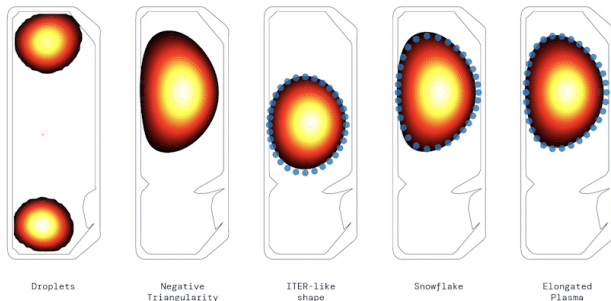


# RL for plasma control in a fusion reactor (DeepMind)



Învățarea plasamentului optim al circuitelor prefix paralele, cum ar fi sumatoarele, optimizând suprafața, întârzierea și puterea.

# RL pentru controlul plasmei într-un reactor de fuziune (DeepMind)



Diferite forme ale plasmei au fost obținute de controlerul învățat, inclusiv o configurație nouă de tip „picătură”.

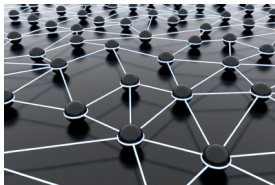
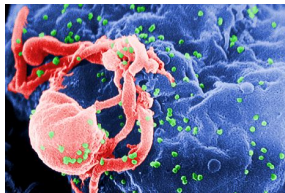
# Planificarea pentru un robot domestic (UTCluj)

Robotul domestic se asigură că întrerupătoarele sunt oprite  
Control la nivel înalt (acțiuni realizate de alte controlere la nivelul fizic)



# Alte aplicații

Inteligența artificială, medicină, sisteme multiagent, economie etc.



- **Curs 1: Problema de învățare prin recompensă**
- Soluția optimală
- Programarea dinamică exactă
- Învățarea prin recompensă exactă
- Tehnici de aproximare
- Programarea dinamică cu aproximare
- Învățarea prin recompensă cu aproximare
- Rețele neurale profunde
- Învățarea prin recompensă cu rețele neurale profunde

## Partea I

### Problema de învățare prin recompensă



# Conținut curs 1

- 1 Introducere
- 2 Cazul determinist
- 3 Cazul stohastic
- 4 Organizarea CI

# De ce învățare?

**Învățarea** identifică soluții care:

- ① nu pot fi proiectate în avans
  - problema prea complexă  
(ex. controlul sistemelor puternic neliniare)
  - problema incomplet cunoscută  
(ex. explorarea robotizată a spațiului cosmic)
- ② se îmbunătățesc permanent
- ③ se adaptează unui mediu variabil în timp

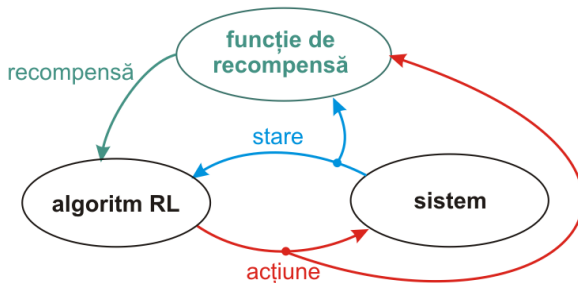
Esențială pentru orice sistem **inteligent**

# Metode bazate pe model

Cursul va pune accent și pe **metodele bazate pe model**:

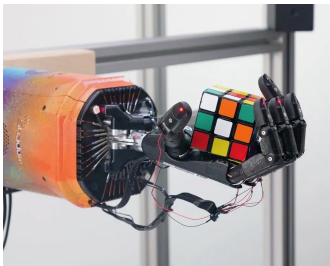
- Stau la baza învățării prin recompensă (ex. programarea dinamică)
- Sunt inspirate din învățarea prin recompensă (ex. planificarea)
- Sunt utile separat de învățare, când avem modelul, fiindcă tratează probleme complexe (ex. neliniare)

# Principiul RL



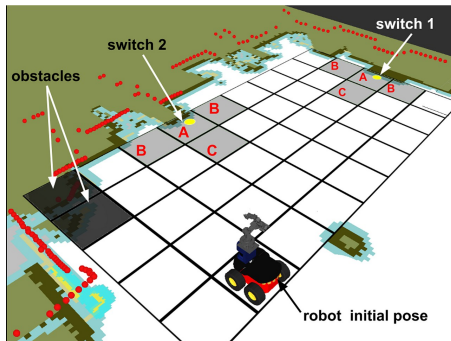
- Interacțiune cu un sistem prin **stări** și **acțiuni**
- Feedback despre performanță în forma **recompensei**
- Inspirată din învățarea umană și animală

# Exemplu: manipularea cubului Rubik



- **Stări:** unghiurile articulațiilor, pozițiile și orientările cubului și ale țintei
- **Acțiuni:** 11 valori pentru fiecare dintre cele 20 de articulații acționate
- **Recompense:**
  - distanța față de starea țintă
  - recompensă pozitivă când o țintă este atinsă
  - recompensă negativă când cubul este scăpat

# Exemplu: robot domestic



- **Stări**: coordonate pe grid, stările întrerupătoarelor
- **Acțiuni**: mișcări NSEV, comutare întrerupător
- **Recompense**: când un întrerupător pornit este oprit (și penalizare când unul oprit este pornit!)

Exemplu de **abstractizare**: problema rezolvată la nivel înalt, acțiunile efectuate de către controlere la nivelul fizic

# Exact vs. aproximativ; Determinist vs. stohastic

- **Părțile 1–3: metode exacte** – stări și acțiuni discrete cu un număr mic de valori
  - pas intermediar, necesar pentru a înțelege problema mai dificilă cu aproximare
  - util și separat, dacă problema poate fi abstractizată într-una discretă la nivel înalt
- Partea 4 și mai departe: metode cu aproximare – stări și acțiuni discrete cu multe valori, sau continue

Sistemul se poate comporta:

- Determinist – răspunde la aceeași acțiune în același fel
- Stohastic



## 1 Introducere

## 2 Cazul determinist

- Proces de decizie Markov
- Legea și obiectivul de control

## 3 Cazul stohastic

## 4 Organizarea CI

# Un exemplu simplu: robot menajer



- Robot menajer într-o lume 1-D
- Colectează gunoi (recompensă +5) sau baterie (recompensă +1)
- După ce un obiect a fost colectat, episodul se termină

# Robot menajer: Stare & acțiune

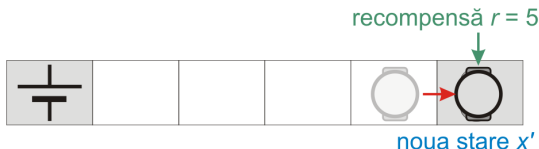


- Robotul se află într-o **stare**  $x$  (căsuță)
- și aplică o **acțiune**  $u$  (ex. înaintează la dreapta)



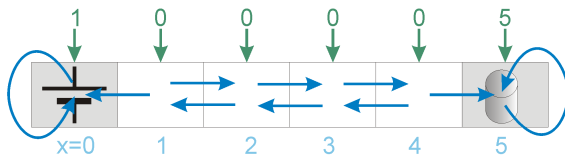
- Spațiul stărilor  $X = \{0, 1, 2, 3, 4, 5\}$
- Spațiul acțiunilor  $U = \{-1, 1\} = \{\text{stânga, dreapta}\}$

# Robot menajer: Tranziție și recompensă



- Robotul atinge o nouă stare  $x'$
- și primește o recompensă  $r$  = calitatea tranziției (aici, +5 pentru colectarea gunoiului)

# Robot menajer: Funcții de tranziție & recompensă



- **Funcția de tranziție** (comportamentul sistemului):

$$x' = f(x, u) = \begin{cases} x & \text{dacă } x \text{ terminal (0 sau 5)} \\ x + u & \text{altfel} \end{cases}$$

- **Funcția de recompensă** (performanța imediată):

$$r = \rho(x, u) = \begin{cases} 1 & \text{dacă } x = 1 \text{ și } u = -1 \text{ (baterie)} \\ 5 & \text{dacă } x = 4 \text{ și } u = 1 \text{ (gunoi)} \\ 0 & \text{altfel} \end{cases}$$

- **De notat:** Stările terminale nu pot fi părăsite și nu sunt recompensate!

# O notă asupra recompensei

- De fapt, recompensa depinde de **tranziție**  $r = \tilde{\rho}(x, u, x')$
- Dar  $x'$  determinat de  $(x, u)$  și poate fi înlocuit în formulă:

$$\tilde{\rho}(x, u, x') = \tilde{\rho}(x, u, f(x, u)) = \rho(x, u)$$

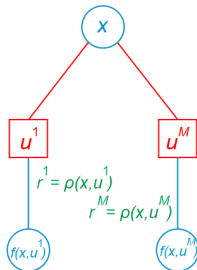
$$r = \rho(x, u) = \begin{cases} 1 & \text{dacă } x = 1 \text{ și } u = -1 \text{ (baterie)} \\ 5 & \text{dacă } x = 4 \text{ și } u = 1 \text{ (gunoi)} \\ 0 & \text{altfel} \end{cases}$$

# Proces de decizie Markov, cazul determinist

## Proces de decizie Markov

Format din:

- 1 Spațiul stărilor  $X$
- 2 Spațiul acțiunilor  $U$
- 3 Funcția de tranziție  $x' = f(x, u)$ ,  $f : X \times U \rightarrow X$
- 4 Funcția de recompensă  $r = \rho(x, u)$ ,  $\rho : X \times U \rightarrow \mathbb{R}$



- 1 Introducere
- 2 **Cazul determinist**
  - Proces de decizie Markov
  - **Legea și obiectivul de control**
- 3 Cazul stohastic
- 4 Organizarea CI

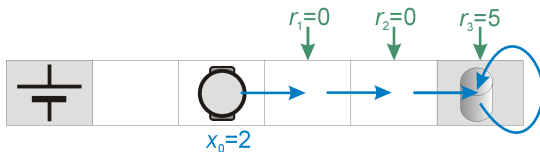
# Lege de control

- **Legea de control**  $h$ : funcție din  $x$  în  $u$  (reacție de la stare)



Exemplu:  $h(0) = *$  (stare terminală, acțiunea este irelevantă),  
 $h(1) = -1$ ,  $h(2) = 1$ ,  $h(3) = 1$ ,  $h(4) = 1$ ,  $h(5) = *$

# Robot menajer: Return



Luăm  $h$  care merge întotdeauna la dreapta

$$\begin{aligned} R^h(2) &= \gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \gamma^3 0 + \gamma^4 0 + \dots \\ &= \gamma^2 \cdot 5 \end{aligned}$$

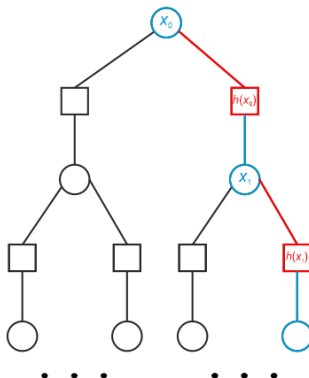
Fiindcă  $x_3$  terminală, toate recompensele ulterioare sunt 0

# Obiectiv

Găsește  $h$  care maximizează **returnul**:

$$R^h(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$

din orice  $x_0$



# Factor de discount

Factor de discount  $\gamma \in [0, 1)$ :

- induce un “pseudo-orizont” pentru optimizare
- mărginește suma infinită
- reprezintă incertitudinea crescândă despre viitor
- ajută convergența algoritmilor

De notat: Există și alte tipuri de return!



# Alegerea factorului de discount

Pentru a alege  $\gamma$ , **compromis** între:

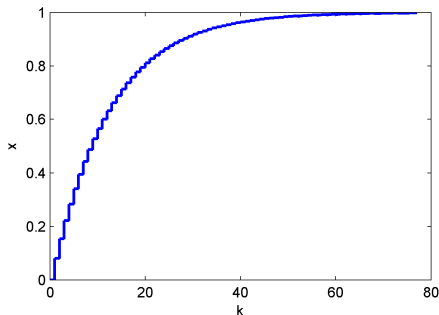
- 1 Calitatea pe termen lung a soluției ( $\gamma$  mare)
- 2 “Simplitatea” problemei ( $\gamma$  mic)

În practică,  $\gamma$  suficient de mare pentru a nu ignora recompense importante de-a lungul traiectoriilor sistemului



# Exemplu: Alegerea $\gamma$ pentru un sistem simplu

Răspunsul unui sistem liniar de ordinul 1:



Valoarea  $\gamma$  pentru ca recompensele la intrarea în regim staționar să fie vizibile din starea inițială?

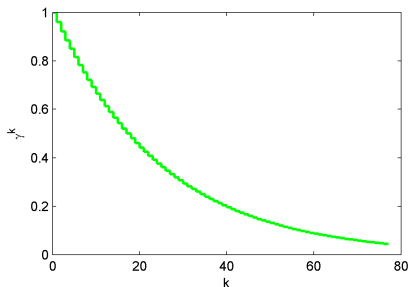
# Soluție: Alegerea $\gamma$ pentru un sistem simplu

Pentru  $k \approx 60$ ,  $\gamma^k$  să nu fie prea mic, de ex.

$$\gamma^{60} \geq 0.05$$

$$\gamma \geq 0.05^{1/60} \approx 0.9513$$

$\gamma^k$  pentru  $\gamma = 0.96$ :



1 Introducere

2 Cazul determinist

3 Cazul stohastic

- Probabilități
- Problema de RL în cazul stohastic

4 Organizarea CI

# Variabile aleatoare discrete

- O variabilă discretă  $x$  poate lua  $n$  valori, în setul  $X = \{x_1, x_2, \dots, x_n\}$ .
- Fiecare valoare este asociată cu o probabilitate  $p(x_1), p(x_2), \dots, p(x_n)$ , unde  $p(x_i) \in [0, 1]$ ,  $\sum_i p(x_i) = 1$ . Funcția  $p : X \rightarrow [0, 1]$  se numește **funcția de masă de probabilitate** (probability mass function, PMF).

Exemplu: Valoarea unui zar este o variabilă aleatoare discretă, cu  $n = 6$  valori posibile,  $x_1 = 1, \dots, x_6 = 6$ . Pentru un zar corect,  $p(x_i) = \frac{1}{6}, \forall i = 1, \dots, 6$

De notat:  $n$  poate să crească la infinit; descrierea matematică rămâne validă

## Valoarea așteptată (expectanța)

- Media valorilor, ponderată de probabilități; valoarea “așteptată” a priori, dată fiind distribuția de probabilitate:

$$E \{x\} = \sum_{x \in X} p(x)x$$

Exemplu: Pentru un zar corect, expectanța este

$$E \{x\} = \frac{1}{6}1 + \frac{1}{6}2 + \dots + \frac{1}{6}6 = 7/2$$

- O **funcție** cu o variabilă aleatoare ca argument,  $g : X \rightarrow \mathbb{R}$  este la rândul ei o variabilă aleatoare, cu expectanța:

$$E \{g(x)\} = \sum_{x \in X} p(x)g(x)$$

Exemplu: Dacă fețele 1-4 câștigă 1\$, iar fețele 5-6, 10\$,

$$E \{x\} = \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}10 + \frac{1}{6}10 = 4\$$$



# Independență

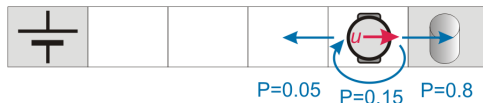
Variabilele aleatoare  $x, y$  sunt independente dacă probabilitatea vectorului  $z = (x, y)$  este  $p_z(z) = p_x(x) \cdot p_y(y)$ , unde  $p_z, p_x, p_y$  sunt PMFurile celor trei variabile (conceptul se extinde la oricâte variabile)

Exemple:

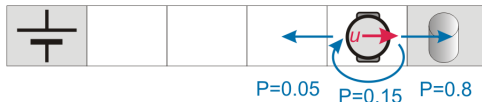
- Valorile unui zar aruncat la momente diferite de timp sunt independente. Printre altele, probabilitatea de a obține 6 este independentă de câte valori 6 au fost obținute la pașii anteriori
- Valorile temperaturii în două zile consecutive nu sunt independente! Sistemul este dinamic (are inerție), valorile curente depind de cele anterioare

# Cazul stochastic

- Starea nu mai evoluează deterministic, ci **stochastic**
- Ex. robotul menajer “alunecă” și:
  - se deplasează în direcția intenționată cu proba. 0.8
  - rămâne pe loc cu proba. 0.15
  - se deplasează în direcția opusă cu proba. 0.05



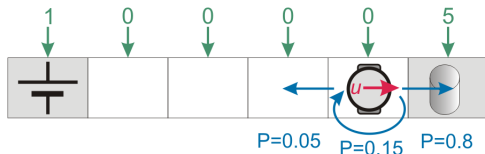
# Robot menajer stochastic: Funcția de tranziție



$\tilde{f}(x, u, x') = \text{probabilitatea}$  de a ajunge în  $x'$   
după ce  $u$  a fost aplicată în  $x$

$$\tilde{f}(x, u, x') = \begin{cases} 1 & \text{dacă } x \text{ terminal, } x' = x \\ 0.8 & \text{dacă } x \text{ neterminal, } x' = x + u \\ 0.15 & \text{dacă } x \text{ neterminal, } x' = x \\ 0.05 & \text{dacă } x \text{ neterminal, } x' = x - u \\ 0 & \text{altfel} \end{cases}$$

# Robot menajer stohastic: Funcția de recompensă



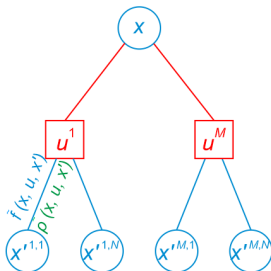
- Tranziția nu mai este complet determinată de  $(x, u)$   
 $\Rightarrow$  starea următoare  $x'$  trebuie inclusă explicit
- $\tilde{p}(x, u, x')$  = recompensa asociată atingerii  $x'$  ca urmare a acțiunii  $u$  în  $x$
- Pentru robotul menajer:

$$\tilde{p}(x, u, x') = \begin{cases} 5 & \text{dacă } x \neq 5 \text{ și } x' = 5 \\ 1 & \text{dacă } x \neq 0 \text{ și } x' = 0 \\ 0 & \text{altfel} \end{cases}$$

# Proces de decizie Markov: cazul stohastic

## Proces de decizie Markov

- 1 Spațiul stărilor  $X$
- 2 Spațiul acțiunilor  $U$
- 3 Funcția de tranziție  $\tilde{f}(x, u, x')$ ,  $\tilde{f} : X \times U \times X \rightarrow [0, 1]$
- 4 Funcția de recompensă  $\tilde{p}(x, u, x')$ ,  $\tilde{p} : X \times U \times X \rightarrow \mathbb{R}$

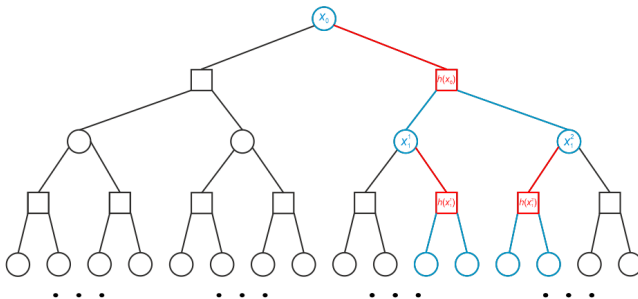


# Obiectiv în cazul stochastic

Găsește  $h$  care maximizează **returnul așteptat**:

$$R^h(x_0) = E_{x_1, x_2, \dots} \left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{\rho}(x_k, h(x_k), x_{k+1}) \right\}$$

din orice  $x_0$

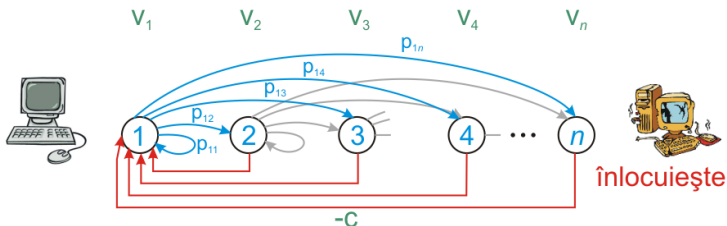


# Lege de control, factor de discount în cazul stohastic

De notat:

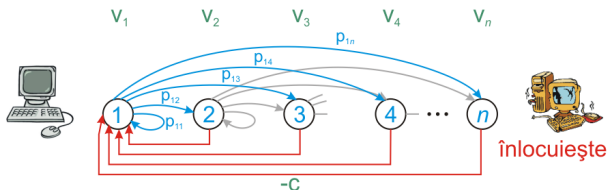
- legea de control  $h(x)$  are aceeași structură
- factorul de discount  $\gamma$  are aceeași semnificație

# Exemplu: Înlocuirea unei mașini



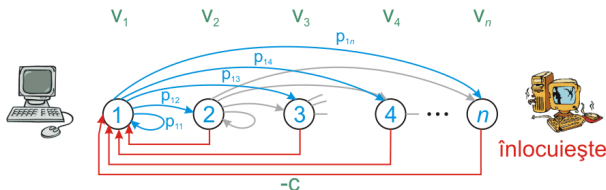
- Mașină de producție cu  $n$  stări diferite = grad de uzură  
1=stare perfectă,  $n$ =complet degradată
- Produce valoarea  $v_i$  operând în starea  $i$
- Uzură stohastică: starea  $i$  trece în  $j > i$  cu proba.  $p_{ij}$ ,  
rămâne în  $i$  cu  $p_{ii} = 1 - p_{i,i+1} - \dots - p_{i,n}$
- Mașina poate fi oricând înlocuită (presupunem  
instantaneu), plătind costul  $c$

# Exemplu: Procesul de decizie Markov



- Spațiul stărilor  $X = \{1, 2, \dots, n\}$
- Spațiul acțiunilor  $U = \left\{ \text{Așteaptă}, \text{Înlocuiește} \right\}$   
(en. **W**ait, **R**eplace)

# Exemplu: Procesul de decizie Markov (continuare)



- Funcția de tranziție:

$$\tilde{f}(x = i, u, x' = j) = \begin{cases} p_{ij} & \text{dacă } u = A \text{ și } i \leq j \\ 1 & \text{dacă } u = I \text{ și } j = 1 \\ 0 & \text{în orice altă situație} \end{cases}$$

- Funcția de recompensă:

$$\tilde{\rho}(x = i, u, x' = j) = \begin{cases} v_i & \text{dacă } u = A \\ -c + v_1 & \text{dacă } u = I \end{cases}$$

# Înlocuirea unei mașini: motivare

Cadrul RL oferă o

**lege de decizie** optimală care

**maximizează valoarea pe termen lung** a mașinii

$$R^h(x_0) = \mathbb{E}_{x_1, x_2, \dots} \left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{p}(x_k, h(x_k), x_{k+1}) \right\}$$

# Terminologie engleză

învățarea prin recompensă  
stare

acțiune

recompensă

funcție de tranziție

funcție de recompensă

proces de decizie Markov

lege de control

return

factor de discount

variabilă aleatoare

funcție de masă de probabilitate

valoarea așteptată

= reinforcement learning, RL

= state

= action

= reward

= transition function

= reward function

= Markov decision process

= policy

= return

= discount factor

= random variable

= probability mass function

= expected value

# Bibliografie

Material obligatoriu: slide-urile pentru cursuri

Cărți opționale:

- D. Bertsekas, Dynamic Programming and Optimal Control, vol. 2, Athena Scientific, 2012.
- R. Sutton, A. Barto, Reinforcement Learning: An Introduction, ed. 2, 2018.
- D. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, 2024.
- L. Buşoniu, Reinforcement learning and dynamic programming for control, 2012 (lecture notes).



# Logistică

## Notare:

- 50% examen
- 50% laboratoare
- 10% teste curs

## Regulament laboratoare:

- laboratoarele **obligatorii pentru participarea la examen**
- soluție = raport PDF + cod: max 10p predată la timp, max 5p dacă întârzie
- soluțiile trebuie validate prin discuții la colocviu
- orice laborator copiat sau generat de LLM  $\Rightarrow$  recontractare



# Website, contact

<http://busoniu.net/teaching/ci2026>

Email: [lucian@busoniu.net](mailto:lucian@busoniu.net), [stefan.pirje@aut.utcluj.ro](mailto:stefan.pirje@aut.utcluj.ro)

## Info

- Material de curs (prezentări)
- Laboratoare
- Program
- etc.

# Test

# Test