Învățarea în sisteme multi-agent

Andrei Olaru

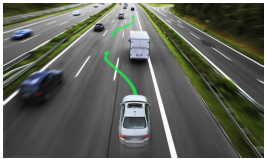
iulie 2025

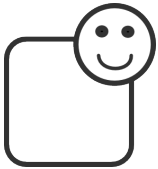
Învățarea în sisteme multi-agent

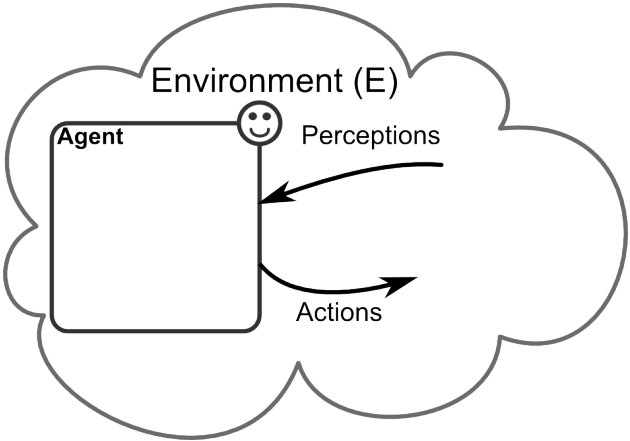
Cuprins

Ce este un agent?

Ce este un agent software?





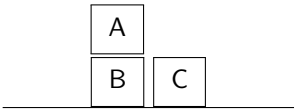


agenții există întotdeauna într-un mediu

Agentul $\equiv$ cel care realizează o acțiune

Agentul $\equiv$ cel care realizează o acțiune

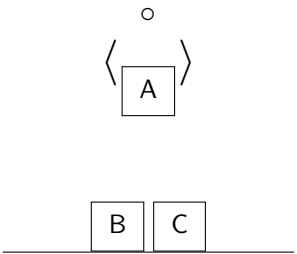
Cum decide agentul *ce* acțiune să realizeze?



vreau blocul B peste A peste C

Abordare reactivă:

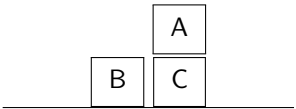
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul B peste A peste C

Abordare reactivă:

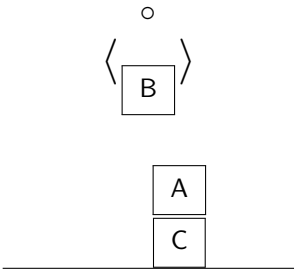
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul B peste A peste C

Abordare reactivă:

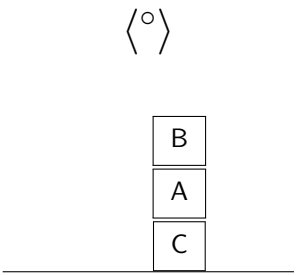
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul B peste A peste C

Abordare reactivă:

- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul B peste A peste C

Abordare reactivă:

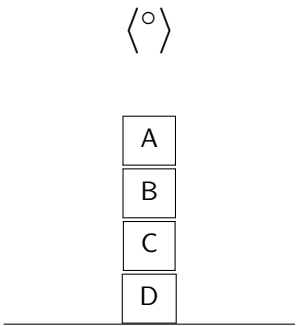
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă

Scenariul 2

Abordare reactivă:

- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă

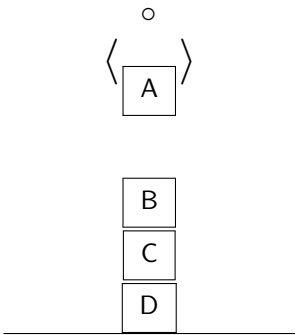
vreau blocul A peste B peste D peste C



vreau blocul A peste B peste D peste C

Abordare reactivă:

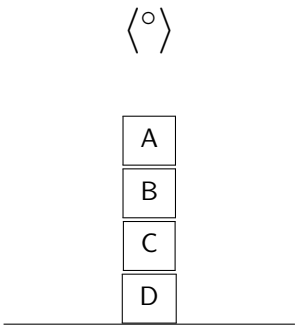
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul A peste B peste D peste C

Abordare reactivă:

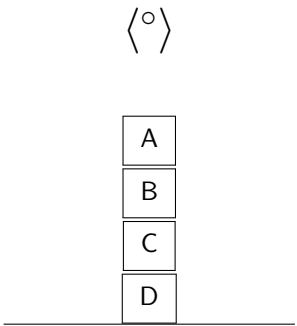
- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul A peste B peste D peste C

Abordare reactivă:

- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă



vreau blocul A peste B peste D peste C

Abordare reactivă:

- dacă blocul X este liber și nu este plasat corect, ridică blocul X
- dacă blocul X este în brat, și blocul Y peste care trebuie să fie este liber, pune X peste Y
- altfel, pune blocul din brat pe masă

???

Abordare cognitivă:

Model al mediului

Abordare cognitivă:

Model al mediului

↓

Simulare a acțiunilor asupra mediului

Abordare cognitivă:

Model al mediului

↓

Simulare a acțiunilor asupra mediului

↓

Explorare a posibilelor stări ale mediului până la starea scop

Abordare cognitivă:

Model al mediului

↓

Simulare a acțiunilor asupra mediului

↓

Explorare a posibilelor stări ale mediului până la starea scop

↓

construcție plan

Abordare cognitivă:

Model al mediului

↓

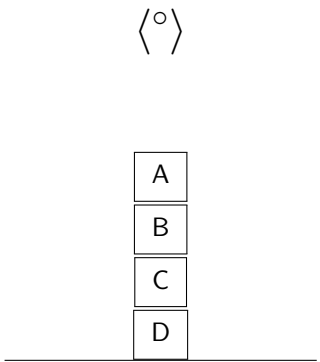
Simulare a acțiunilor asupra mediului

↓

Explorare a posibilelor stări ale mediului până la starea scop

↓

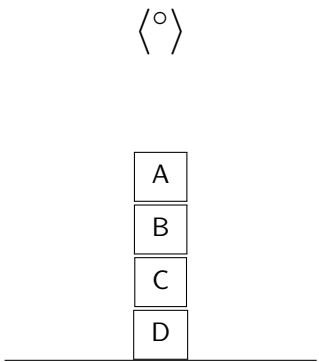
construcție plan ← planul este o secvență de acțiuni



vreau blocul A peste B peste D peste C

Plan:

- iau blocul A, îl pun jos
- iau blocul B, îl pun jos
- iau blocul C, îl pun jos
- iau blocul D, îl pun peste C
- iau blocul B, îl pun peste D
- iau blocul A, îl pun peste B



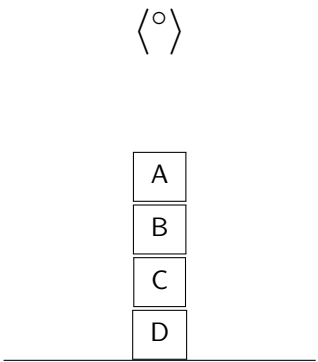
vreau blocul A peste B peste D peste C

Plan:

- iau blocul A, îl pun jos
- iau blocul B, îl pun jos
- iau blocul C, îl pun jos
- iau blocul D, îl pun peste C
- iau blocul B, îl pun peste D
- iau blocul A, îl pun peste B

Am nevoie de

- modelul mediului +
- un algoritm de planificare



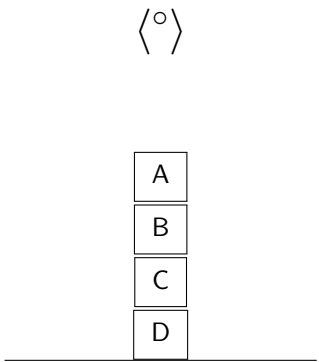
vreau blocul A peste B peste D peste C

Plan:

- iau blocul A, îl pun jos
- iau blocul B, îl pun jos
- iau blocul C, îl pun jos
- iau blocul D, îl pun peste C
- iau blocul B, îl pun peste D
- iau blocul A, îl pun peste B

Am nevoie de

- modelul mediului + ← specific
- un algoritm de planificare



vreau blocul A peste B peste D peste C

Plan:

- iau blocul A, îl pun jos
- iau blocul B, îl pun jos
- iau blocul C, îl pun jos
- iau blocul D, îl pun peste C
- iau blocul B, îl pun peste D
- iau blocul A, îl pun peste B

Am nevoie de

- modelul mediului + ← specific
- un algoritm de planificare ← general

Abordare cu învățare:

Antrenare

Abordare cu învățare:

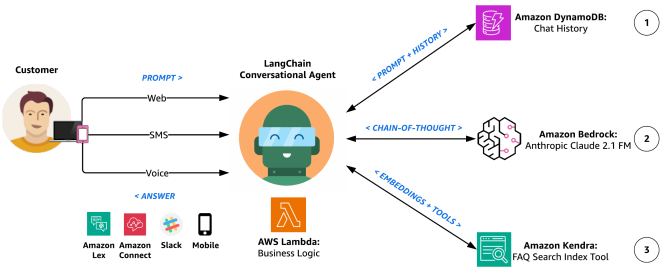
Antrenare → mecanism reactiv în care *inteligența este îmbarcată*

Abordare cu învățare:

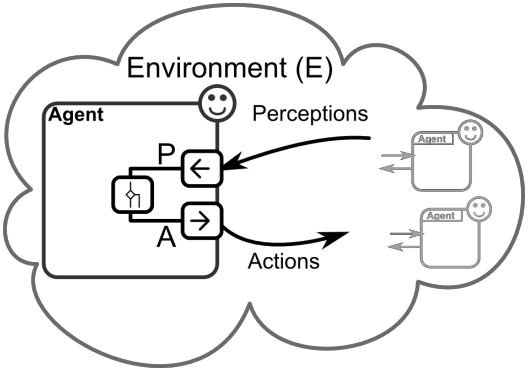
Antrenare → mecanism reactiv în care *inteligența este îmbarcată*

RL · GA · NN

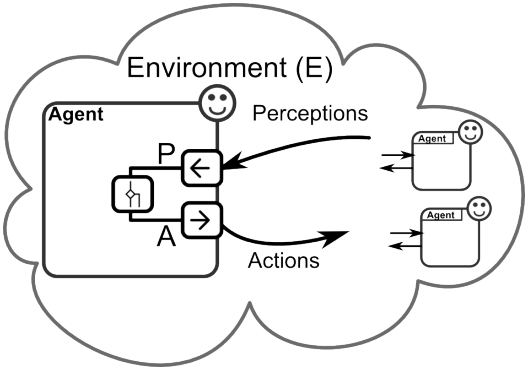
Agentic AI → Agenți care foloses Large Language Models (LLMs) pentru a efecuta raționament / planificare și pentru a hotărî acțiuni de realizat în mediu.



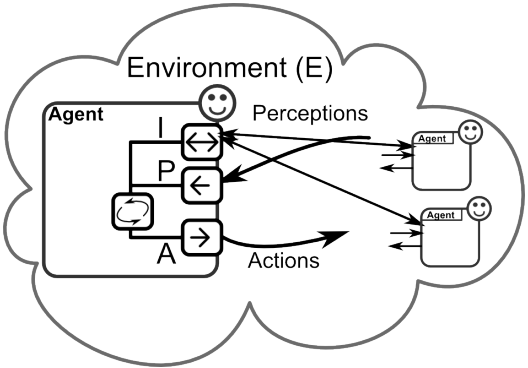
În același mediu putem avea mai mulți agenți care acționează asupra mediului.



agenții percep efectul acțiunilor celorlalți agenți



agenții percep ceilalți agenți ca atare [și pot avea un model pentru ei]



agenții percep ceilalți agenți și pot interacționa cu ei (e.g. prin mesaje)

Cum iau agenții decizii în condițiile existenței mai multor agenți?

(OpenAI Hide and seek)

Problema deciziei

Teoria jocurilor

Știind efectele pe care le au acțiunile agenților, ce acțiuni ar fi *cel mai bine* să realizeze agenții?

Dilema prisonierului (*prisonner's dilemma*)

Dilema prisonierului (*prisonner's dilemma*)

Raționalitate limitată

Dilema prisonierului (*prisonner's dilemma*)

Raționalitate limitată

Cooperare (cu celălalt prizonier) ≡ tăcere la interogatoriu

Trădare (*defection*) (a celuilalt prizonier) ≡ dă vina pe celălalt la interogatoriu

Dilema prisonierului (*prisonner's dilemma*)

Raționalitate limitată

Cooperare (cu celălalt prizonier) ≡ tăcere la interogatoriu

Trădare (*defection*) (a celuilalt prizonier) ≡ dă vina pe celălalt la interogatoriu

		P_2	
	P_1, P_2	C	D
P_1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

		P_2	
	P_1, P_2	C	D
P_1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

Ce înseamnă *cel mai bine* ?

		P_2	
	P_1, P_2	C	D
P_1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

Ce înseamnă *cel mai bine* ?

bunăstare socială ← interesul societății

		P_2	
	P_1, P_2	C	D
P_1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

Ce înseamnă *cel mai bine* ?

bunăstare socială

optimalitate Pareto ← nu se poate mai bine

		P_2	
	P_1, P_2	C	D
P_1	C	-1, -1	-3, 0
	D	0, -3	-2, -2

Ce înseamnă *cel mai bine* ?

bunăstare socială

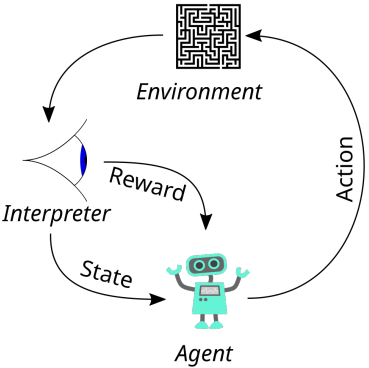
optimalitate Pareto

echilibru Nash ← raționalitate individuală

Battle of sexes:

		Bob	
	Alice, *Bob*	*Football*	*Ballet*
Alice	*Football*	2, 3	0, 0
	Ballet	1, 1	3, 2

Agenții pot învăța



[Wikipedia]

Într-un număr de episoade, un agent încearcă să descopere o soluție și primește un feedback asupra calității soluției.

În *învățarea supervizată*, feedbackul este unul pozitiv sau negativ pentru fiecare episod / exemplu.

În *învățarea prin recompensă*, feedbackul este o recompensă pentru fiecare soluție / pentru fiecare pas în evoluția către o soluție.

[Albert walks]

[Albert escapes]

[AI Warehouse]

Value | Policy

cât de bună este o stare → Value | Policy

$$V(s) := \sum_{s'} P_{\pi(s)}(s, s') \left(R_{\pi(s)}(s, s') + \gamma V(s') \right)$$

· se propagă valorile din posibilele stări următoare, cumulat cu recompensa mutării în stările următoare

Value | Policy ← cea mai bună acțiune într-o stare

$$V(s) := \sum_{s'} P_{\pi(s)}(s, s') \left(R_{\pi(s)}(s, s') + \gamma V(s') \right)$$

· se propagă valorile din posibilele stări următoare, cumulat cu recompensa mutării în stările următoare

$$\pi(s) := \text{argmax}_a \left\{ \sum_{s'} P(s' \mid s, a) \left(R(s' \mid s, a) + \gamma V(s') \right) \right\}$$

· politica în fiecare stare este de a alege acțiunea care duce spre o recompensă cumulată maximă

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{est. optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Big)}^{\text{temporal difference}}$$

$$s_t \xrightarrow{\text{acțiunea a durează între t și t+1}} s_{t+1} \text{ \& recompensa } r_t$$

Știu în ce stare am fost și în ce stare am ajuns $\Rightarrow$ pot calcula valoarea Q fără să am nevoie de modelul mediului.

Probleme:

- Dacă spațiul de stări/acțiuni este mare, tabelul devine foarte rar și / sau foarte voluminos de reținut.
- Mediul poate fi dinamic și rezultatul pentru o pereche acțiune / stare să nu fie mereu același
- Distribuția datelor de intrare nu este staționară pentru că agentul explorează diverse părți ale mediului

Deep Q-Learning folosește o rețea neurală pentru a aproxima valorile Q, și o putem folosi chiar și pentru perechi stare-acțiune pentru care nu avem o valoare înregistrată deja în tabel.

Deep Q-Learning folosește o rețea neurală pentru a aproxima valorile Q, și o putem folosi chiar și pentru perechi stare-acțiune pentru care nu avem o valoare înregistrată deja în tabel.

DQN folosește *experience replay* pentru a relua o parte dintre experiența obținută într-o manieră nesecvențială, pentru a deconecta stările succesive și pentru a stabiliza rețeaua neurală.

Se poate folosi un proces de *selecție* a celor mai relevante experiențe și se pot elimina experiențele similare din buffer.

DQN pentru obiecte de învățare vizuale (e.g. jocuri, recunoaștere din clipuri video) și dinamice consideră ca stare ultimele 4 cadre din materialul vizual.

- Dacă sunt mai mulți agenți în sistem, recompensa este primită global, după ce toți agenții și-au executat, simultan, acțiunea $\Rightarrow$ nu știm cărei acțiuni să asociem recompensa (*credit assignment problem*).
- Mediul nu mai este staționar, adică nu se modifică numai în urma acțiunii unui agent care învață, ci și în urma acțiunii altor agenți
- În problemele reale, mediul este dinamic (chiar și în lipsa altor agenți)
- Multe probleme (e.g. jocuri) nu respectă proprietatea Markov că starea viitoare depinde doar de starea curentă, pentru că trebuie realizate secvențe de acțiuni.

Abordarea descentralizată a învățării independente (*independent learners*):

- folosește RL pentru toți agenții cu toate că proprietatea de staționaritate a mediului nu este respectată
- converge pentru jocuri cu sumă zero, dar nu și pentru cazuri cooperative
- converge în cazuri cooperative doar pentru situații specifice

Partajarea informațiilor:

- învățare centralizată și execuție descentralizată – învățarea se face agregând acțiunile și stările tuturor agenților în vectori de acțiuni / stări pentru care se calculează valorile de calitate. $\leftarrow \oplus$ staționaritate $\ominus$ dimensiune tabel

Partajarea informațiilor:

- învățare centralizată și execuție descentralizată – învățarea se face agregând acțiunile și stările tuturor agenților în vectori de acțiuni / stări pentru care se calculează valorile de calitate. ← ⊕ staționaritate ⊖ dimensiune tabel

- utilizarea unui supervizor care construiește recompense pentru fiecare agent în parte.
← ⊕ staționaritate ⊖ implementare supervizor

Partajarea informațiilor:

- învățare centralizată și execuție descentralizată – învățarea se face agregând acțiunile și stările tuturor agenților în vectori de acțiuni / stări pentru care se calculează valorile de calitate. ← ⊕ staționaritate ⊖ dimensiune tabel
- utilizarea unui supervizor care construiește recompense pentru fiecare agent în parte. ← ⊕ staționaritate ⊖ implementare supervizor
- partajarea parametrilor (*parameter sharing*) – agenții partajează, practic, aceeași matrice de valori pentru perechile stare-acțiune, și o actualizează împreună. ← ⊕ descentralizare ⊖ comunicare intensă

Partajarea informațiilor:

- învățare centralizată și execuție descentralizată – învățarea se face agregând acțiunile și stările tuturor agenților în vectori de acțiuni / stări pentru care se calculează valorile de calitate. ← ⊕ staționaritate ⊖ dimensiune tabel
- utilizarea unui supervizor care construiește recompense pentru fiecare agent în parte. ← ⊕ staționaritate ⊖ implementare supervizor
- partajarea parametrilor (*parameter sharing*) – agenții partajează, practic, aceeași matrice de valori pentru perechile stare-acțiune, și o actualizează împreună. ← ⊕ descentralizare ⊖ comunicare intensă
- partajare asimetrică a parametrilor – unul dintre agenți învață din experiența tuturor agenților. ← ⊕ comunicare mai puțin intensă ⊖ performanța celorlați agenți

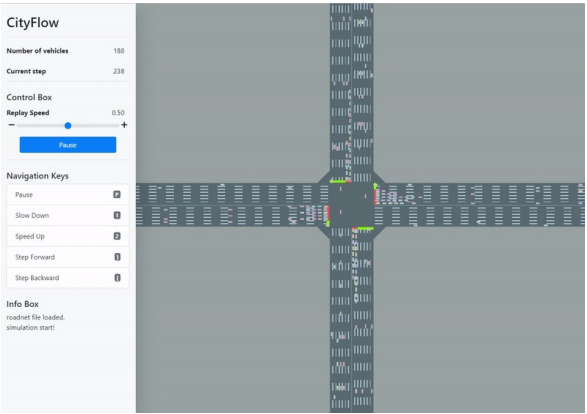
Aplicații

Folosim învățare multi-agent atunci când agenții trebuie să învețe să coopereze.

- Analiza comportamentului emergent:
  - studiul sistemelor de agenți care învață independent
  - observarea proprietăților care apar după ce agenții interacționează
  - favorizarea explorării spațiului de stări prin oferirea de recompense în întreg spațiul

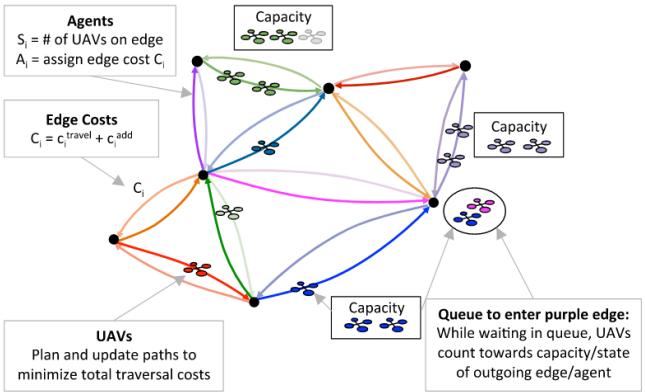
- Învățarea comunicării:
  - în medii *parțial observabile*, este de interes pentru agenți să își comunice informații despre experiența lor
  - rețelele neurale sunt folosite atât pentru a evalua valoarea stărilor, cât și pentru a determina informația de transmis altor agenți

- Învățarea cooperării:



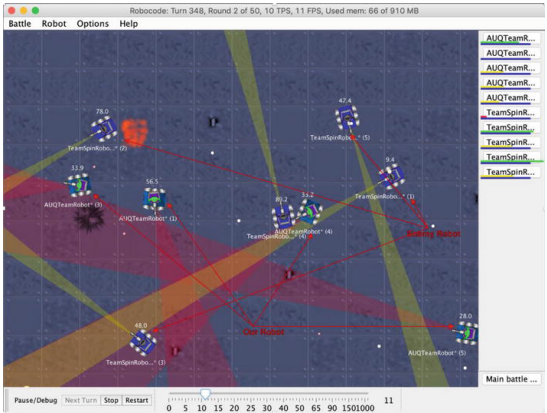
[Tang et al., 2019]

- Învățarea cooperării:



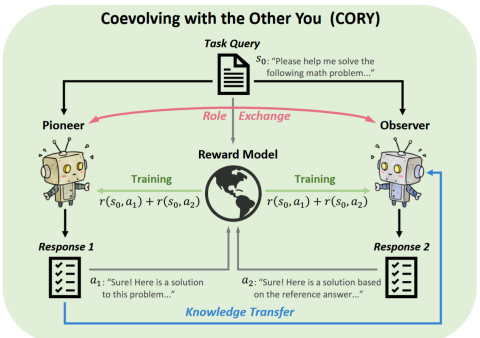
[Chung et al., 2019]

- Învățarea cooperării:



[Ulusoy et al., 2020]

- LLMs + agregarea recompenselor + schimbarea periodică a rolurilor → învățare mai eficientă pentru ambii agenți.



[Ma et al., 2024]

Concluzie

- Învățarea multi-agent este utilă pentru învățarea cooperării
- Spațiul de stări / acțiuni este mare și dimensionalitatea problemei se înmulțește cu numărul de agenți
- Învățarea prin recompensă necesită modificări importante pentru a putea funcționa în cazul mai multor agenți

Chung, J. J., Rebhuhn, C., Yates, C., Hollinger, G. A., and Tumer, K. (2019).

A multiagent framework for learning dynamic traffic management strategies.
Autonomous Robots, 43:1375–1391.

Ma, H., Hu, T., Pu, Z., Boyin, L., Ai, X., Liang, Y., and Chen, M. (2024).

Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning.
Advances in Neural Information Processing Systems, 37:15497–15525.

Tang, Z., Naphade, M., Liu, M.-Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., and Hwang, J.-N. (2019).

Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8797–8806.

Ulusoy, Ü., Güzel, M. S., and Bostanci, E. (2020).

A Q-learning-based approach for simple and multi-agent systems.
In *Multi Agent Systems-Strategies and Applications*. IntechOpen.

Vă mulțumesc!

Întrebări?