Incremental Reinforcement Learning

for Dynamic Environments

Presenter: **Zhi Wang**[1]

Coauthors: Chunlin Chen[2], Han-Xiong Li[1], Daoyi Dong[3], Tzyh-Jong Tarn[4]

[1]City University of Hong Kong

[2] Nanjing University

[3] University of New South Wales, Canberra

[4] Washington University, St. Louis

Table of Contents

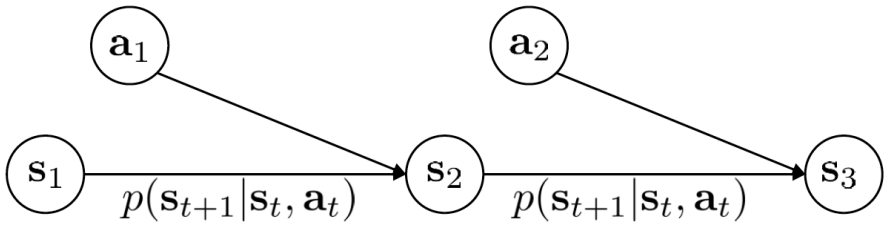
Definitions

Markov Decision Process (MDP) $M = \langle S, A, T, R \rangle$

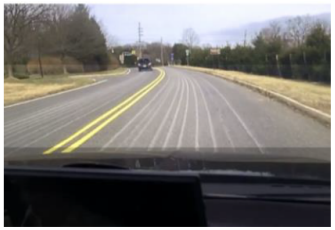
S: State space — states $s \in S$ (discrete or continuous)

A: Action space — actions $a \in A$ (discrete or continuous)

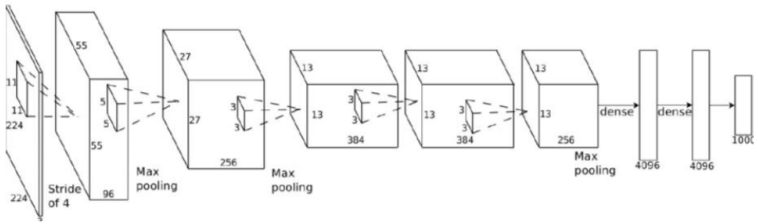
T: Transition operator $T_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$



Reward functions



$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define Markov decision process

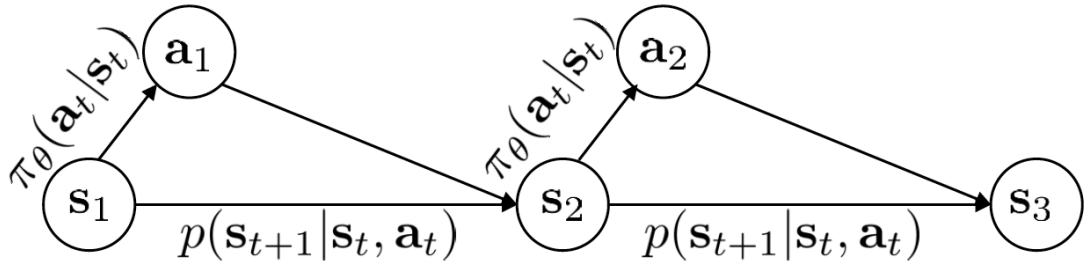
high reward

low reward

The Goal of Reinforcement Learning

$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$



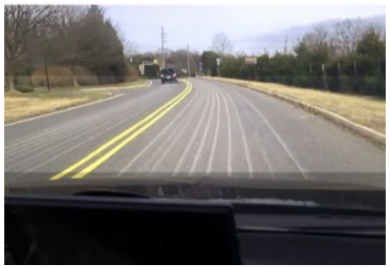
- How agents take actions in an environment to maximize cumulative reward
- In a trial-and-error manner
- A general framework for optimization of sequential decision problems

Table of Contents

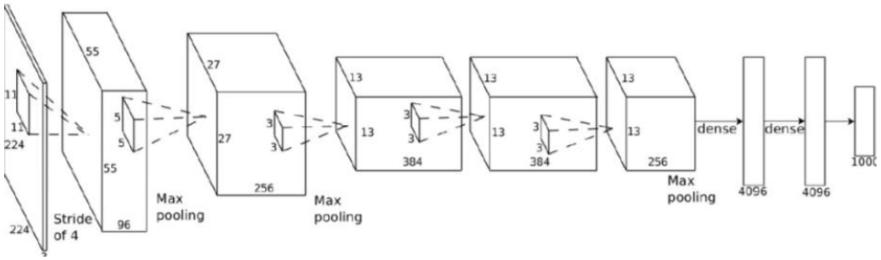
Policy Gradient

Objective Function

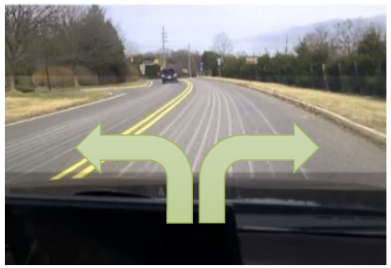
$$\text{maximize } J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)}[r(\tau)] = \int_{\tau} \pi_{\boldsymbol{\theta}}(\tau) r(\tau) \, \mathrm{d}\tau$$



$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



$\mathbf{a}_t$

Direct Policy Differentiation

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

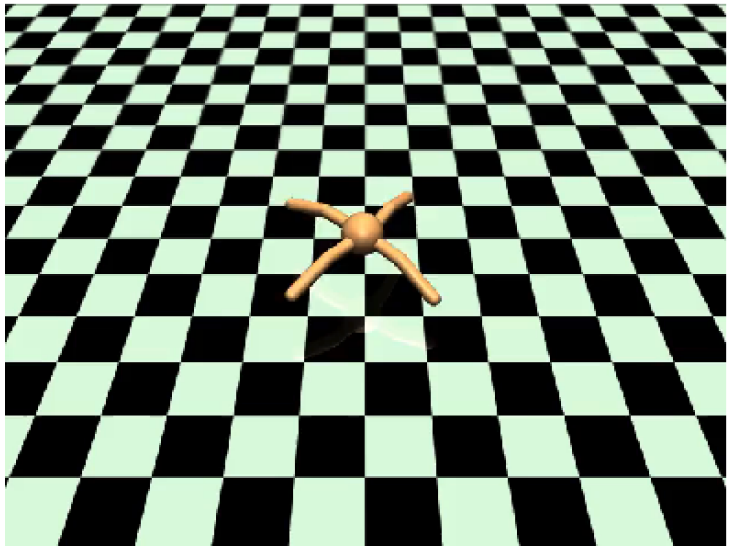
a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)}] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Example: Robot Locomotion



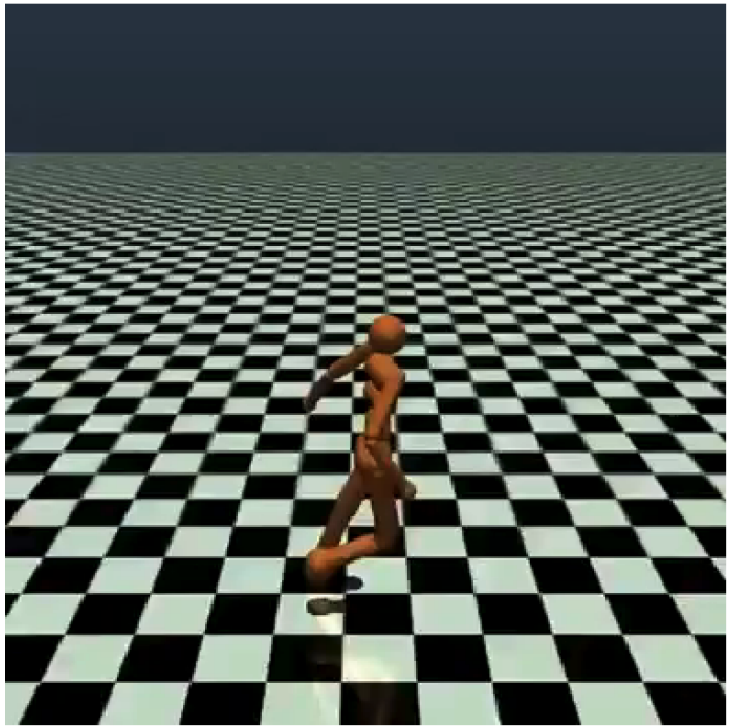


Table of Contents

Q-learning

Q-function (Maximum expected future rewards)

$$Q^*(s,a) = \max_\pi \mathbb{E}_\pi[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} | s_t = s, a_t = a]$$

Bellman Equation (Markov decision process)

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a') | s,a]$$

Value Iteration, Minimize Bellman Residual

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

$$a^* = \arg\max_a Q(s,a)$$

Deep Q-learning

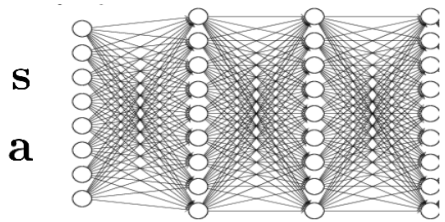
Loss Function, Minimize Bellman Residual

$$\mathcal{L}(\phi) = \mathbb{E}[(r + \gamma \max_{a'} Q_\phi(s', a') - Q_\phi(s, a)^2]$$

full fitted Q-iteration algorithm:

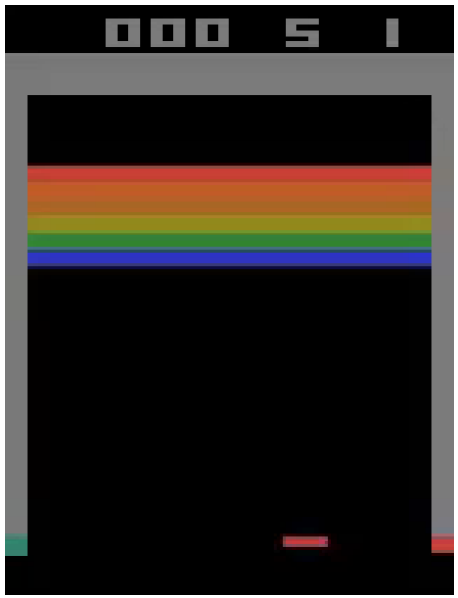
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

($K\times$)



$Q_\phi(\mathbf{s}, \mathbf{a})$
parameters ϕ

Example: Atari Games



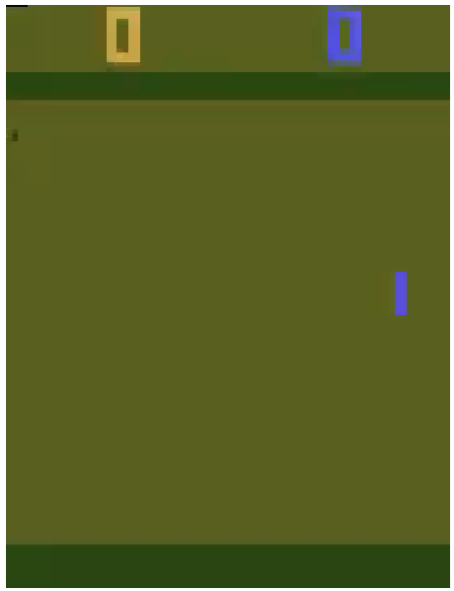


Table of Contents

Challenge: Dynamic Environments

Dynamic Environments

State/action space, reward/state transition function change over time.

Original environment

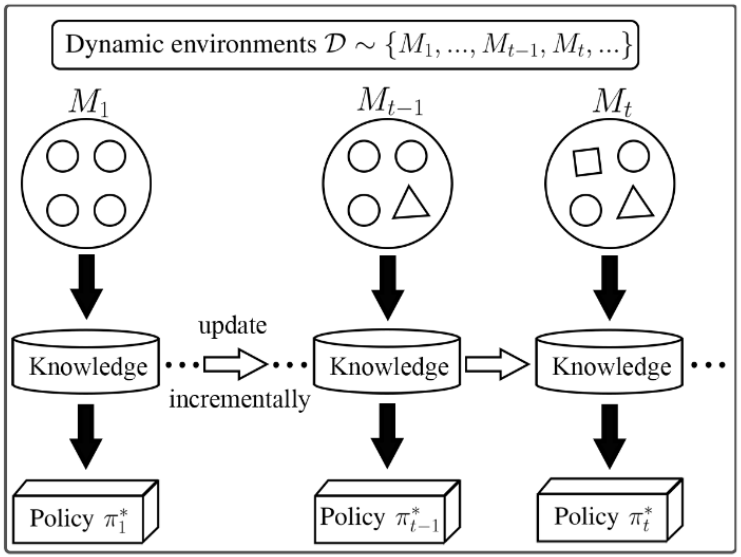
$\Rightarrow$

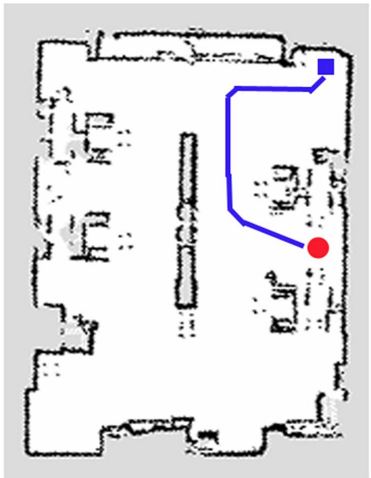
New environment

Incremental Learning

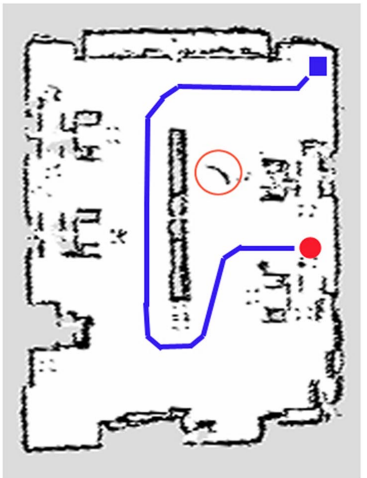
Incremental Learning for Dynamic Environments

- Detect the environment change,
- Adjust existing knowledge rapidly ***in an incremental manner***.





a



b

Why is incremental learning important?

- Changing factors/unexpected perturbations are very common
- Avoid repeatedly training, save large computational resources
- Maintain and update learned knowledge for online applications

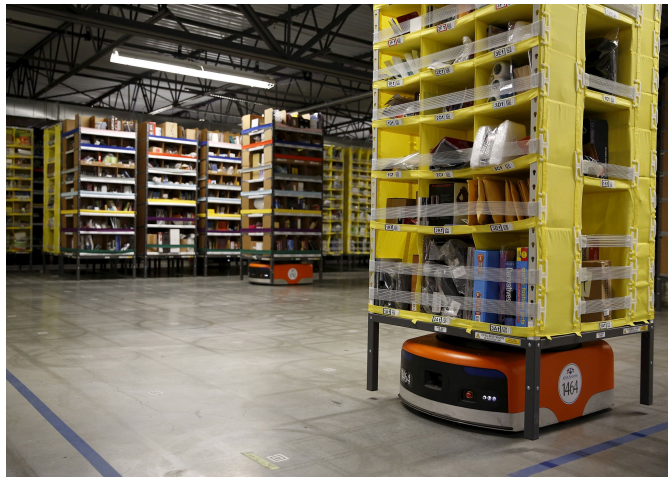
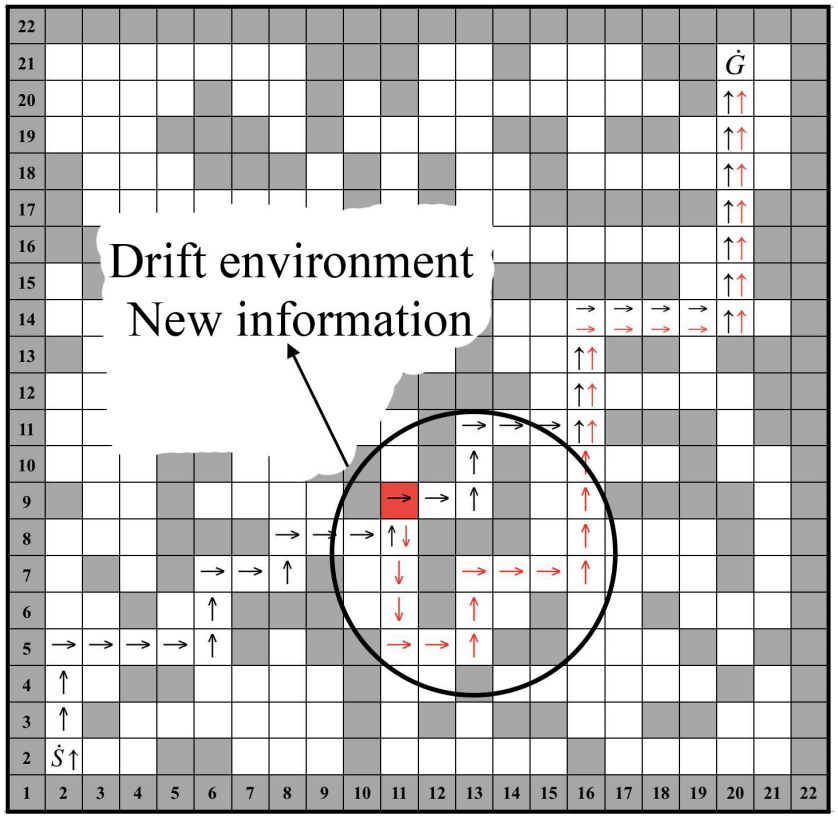


Table of Contents

Algorithm Framework



Detect drift Env.

⇓

m-degree neighbor of drift Env.

⇓

Prioritized sweeping/updating of neighbor Env.

⇓

Re-start a new learning process

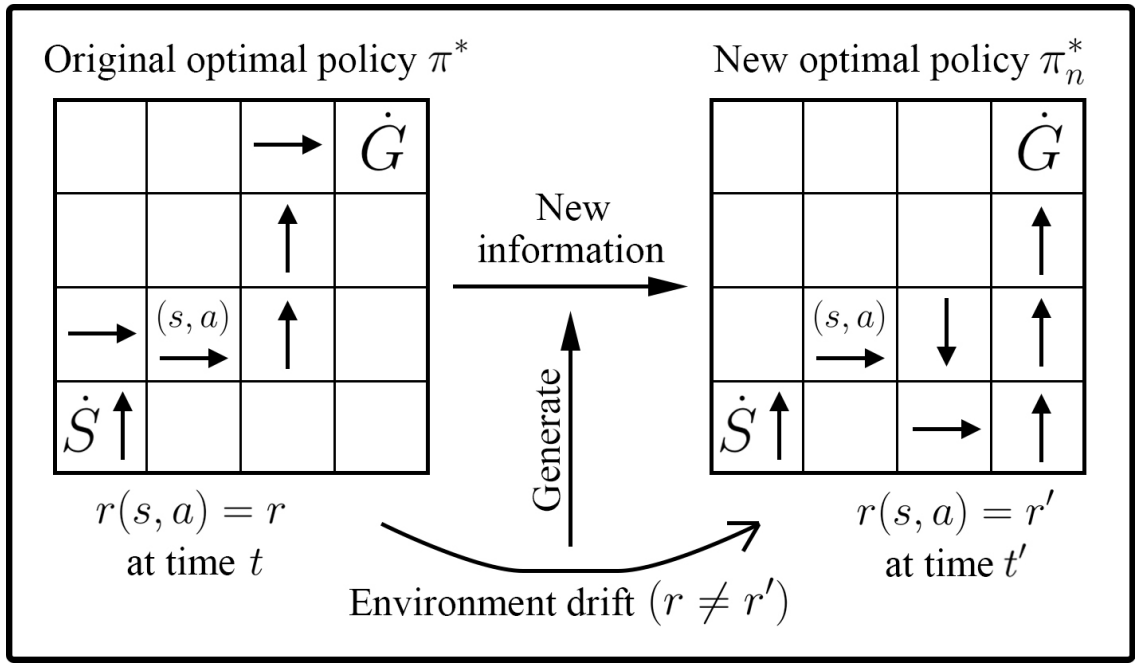
Core Idea – Update New Information First

Fuse new information into the existing knowledge system incrementally

Step 1: Drift Environment

Dynamic Environment – Change of Reward Function

Drift environment: the set of all drifted state-action pairs



Step 2: Detection of Drift Environment

Original Env.

$R(S, A)$

$\Uparrow$

We already have

Observe

$\Longleftrightarrow$

Compare

New Env.

$R_n(S, A)$

$\Uparrow$

Exploration by a detector-agent

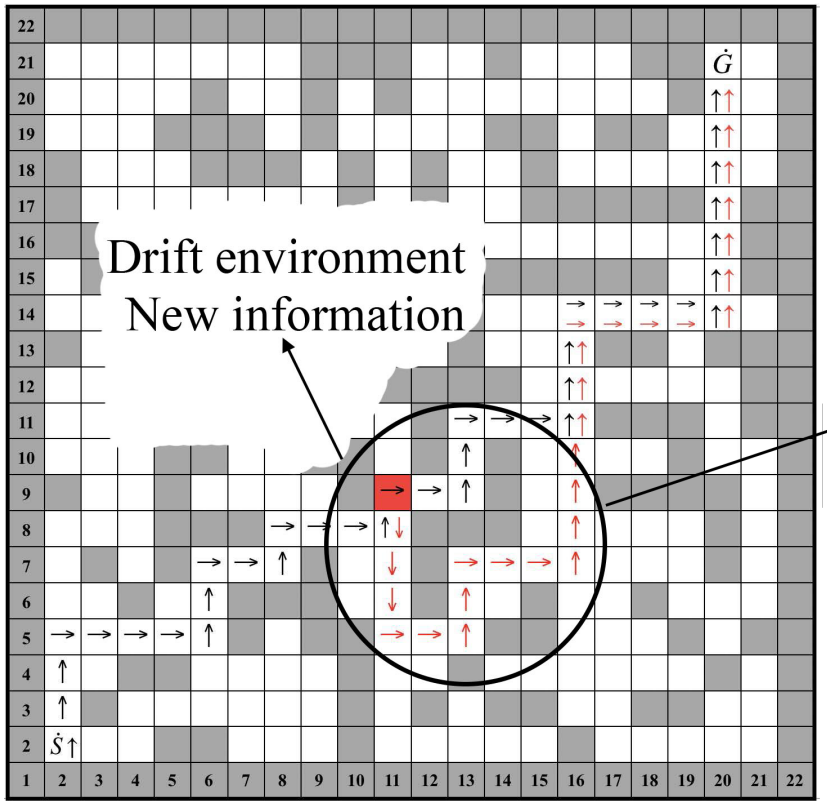
A virtual RL process, pure exploration

- $\pi : p^{\pi}(s, a) = 1/|A(s)|$
- Traverse as many state-action pairs as possible
- No need to update the value functions and policies
- Computational complexity can be neglected

Step 3: Prioritized Sweeping of Drift Environment

Neighborhood Degree, $\mathrm{NEI}[(s,a),(s',a')]$

The step distance (i.e. the least steps of actions) from state s to s' with sequential actions starting with action a.



Prioritized Sweeping

using *Dynamic Programming*

m-degree Neighbor of Drift Env.

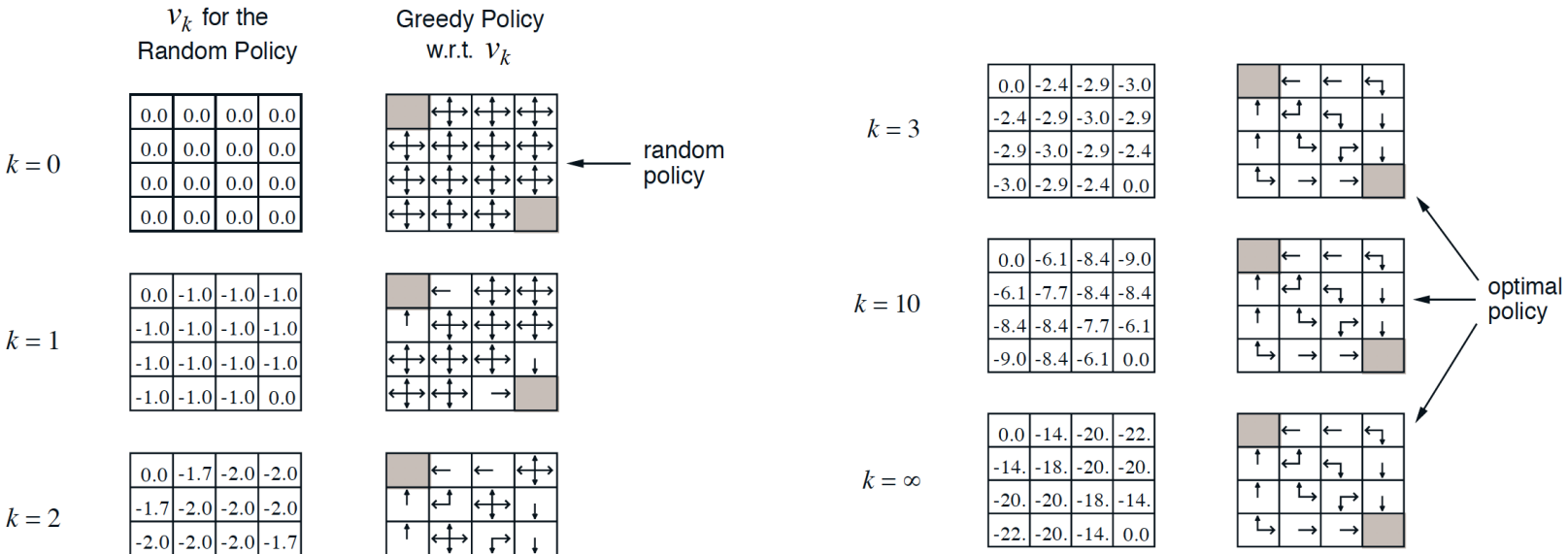
$$E^m_{dn}(S_{dn}, A_{dn}, R_{dn}, P_{dn})$$

$$\mathrm{NEI}[(s_{dn}, a_{dn}), (s_d, a_d)] \leq m$$

Dynamic Programming

A classical model-based algorithm

- $Q(s,a) \leftarrow \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q(s',a')]$
- Repeatedly sweeps through the state-action space till convergence



Step 3: Prioritized Sweeping of Drift Environment

$$Q(s,a) \leftarrow \sum_{s'} p(s'|s,a)[r_n(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

$\Downarrow$ $\Downarrow$

Existing Knowledge

Old value functions

$Q(s,a)$

Fused into $\Longleftarrow$

New Information

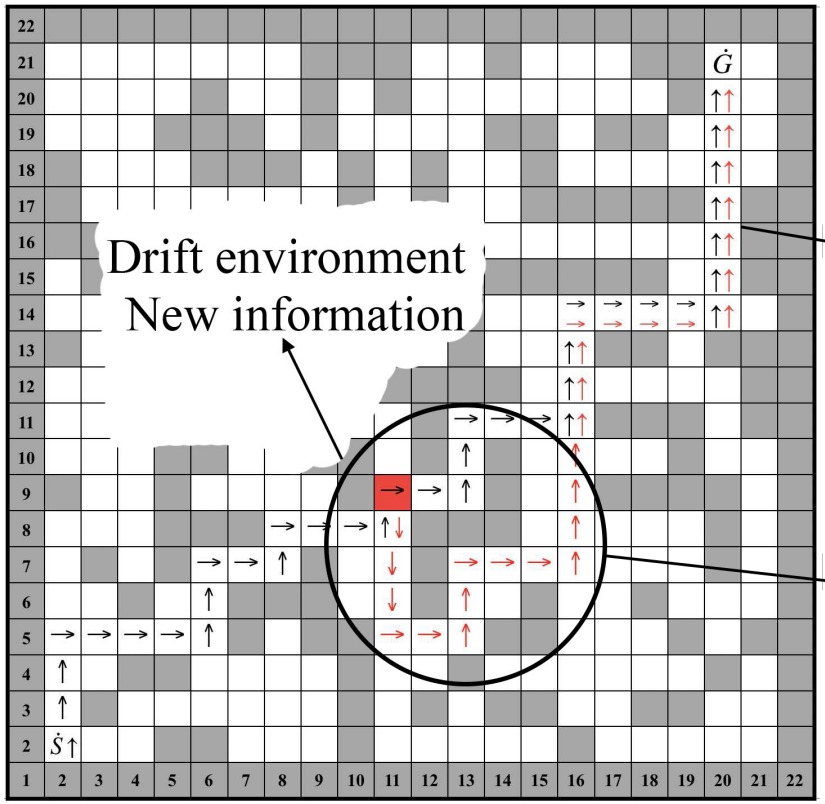
New rewards

$r_n(s,a,s')$

- Do Bellman backups in areas where value functions change most
- Use the implicit old model to concurrently train the new policy
- Alleviate conflict between new information and existing knowledge

Step 4: Re-start a new learning process

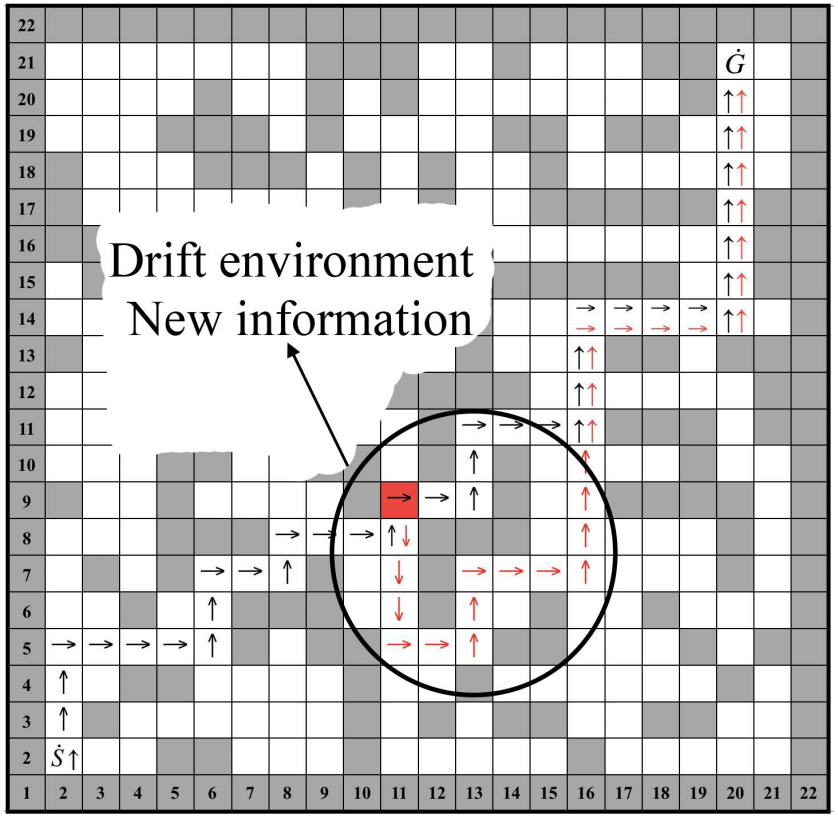
$$Q_n(s_n, a_n) \leftarrow \begin{cases} Q_{dn}(s_n, a_n), & \forall (s_n, a_n) \in (S_{dn}, A_{dn}) \\ Q^*(s_n, a_n), & \forall (s_n, a_n) \notin (S_{dn}, A_{dn}) \end{cases}$$



Q^*: Existing knowledge

Q_{dn}: New information with weakened conflict

Review of Algorithm Framework



Detect drift Env.

⇓

m-degree neighbor of drift Env.

⇓

Prioritized sweeping/updating of neighbor Env.

⇓

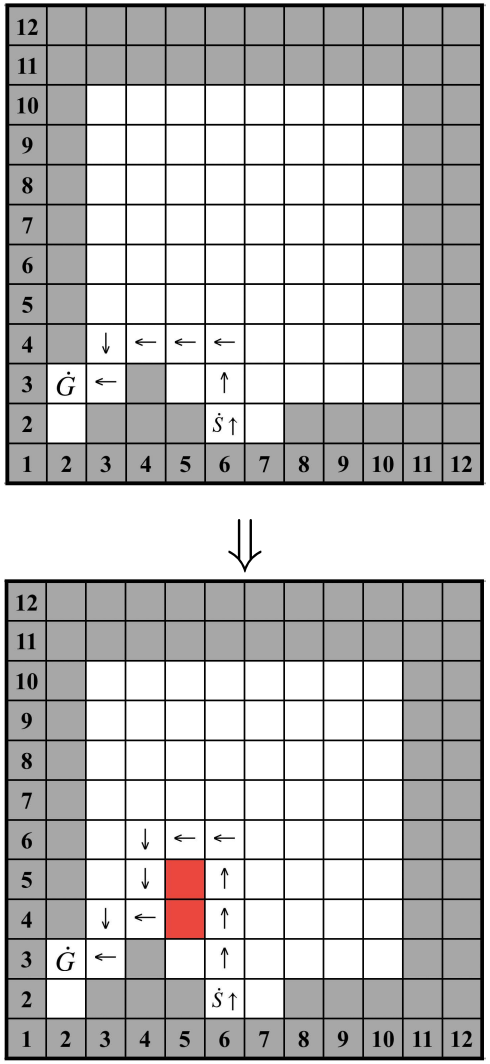
Re-start a new learning process

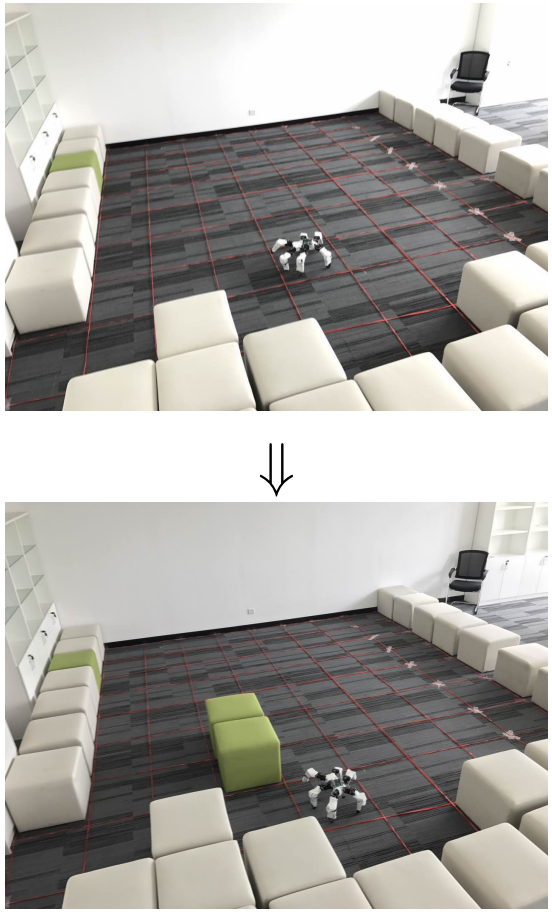
Incremental reinforcement learning with prioritized sweeping for dynamic environments

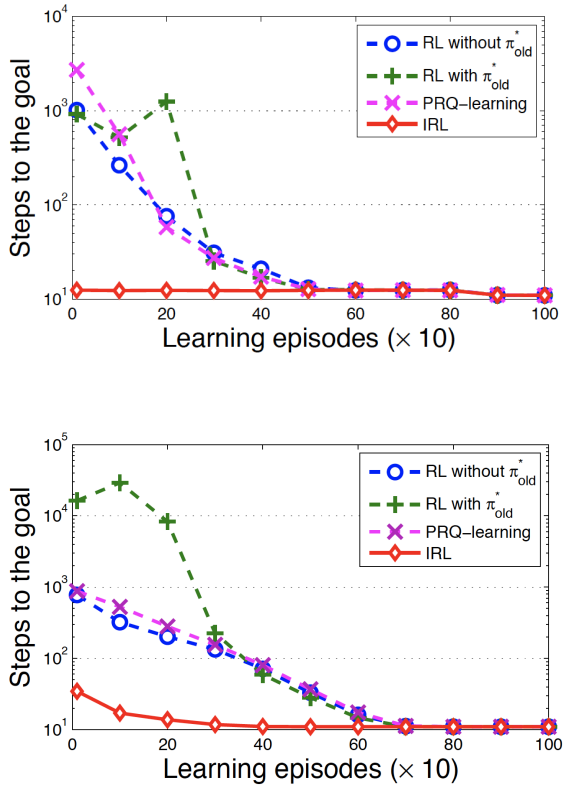
– **Z Wang**, C Chen, HX Li, D Dong, and TJ Tarn
– **IEEE/ASME Transactions on Mechatronics**, 2019

Table of Contents

I. Simple Maze



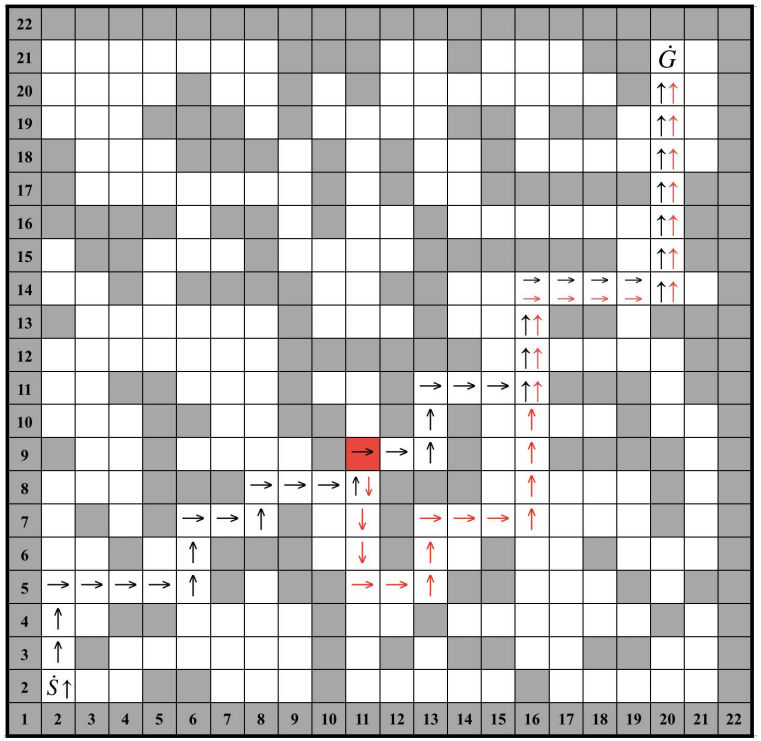


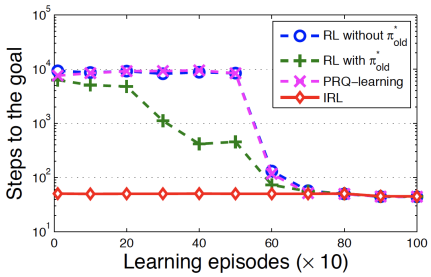


II. Benchmark Maze

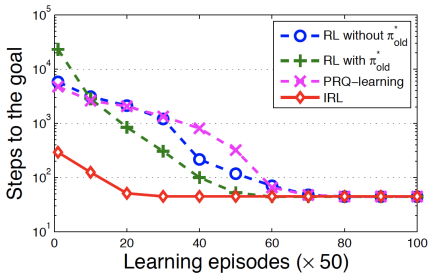
Stochastic Dynamic Environment

- randomly select grids to change
- average over 30 times

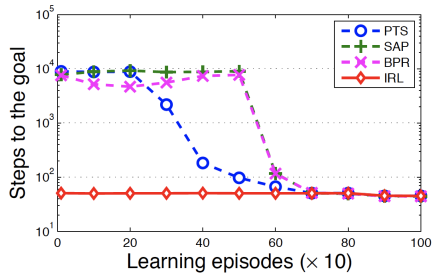




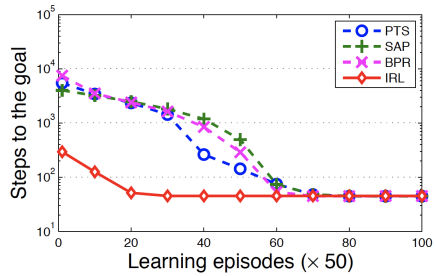
(a) ϵ-greedy



(b) Softmax



(a) ϵ-greedy



(b) Softmax

III. Intelligent Warehouse, Unmanned Logistics

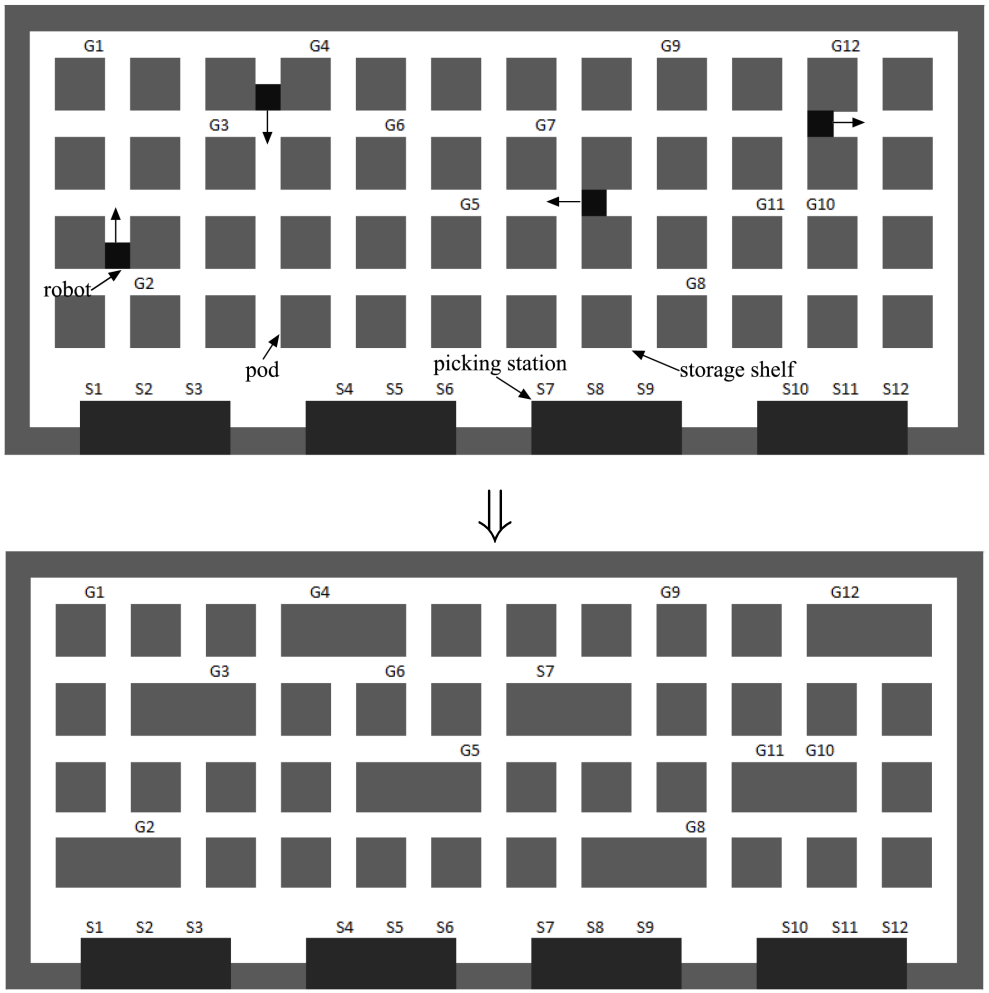
Large-scale Kiva robots system

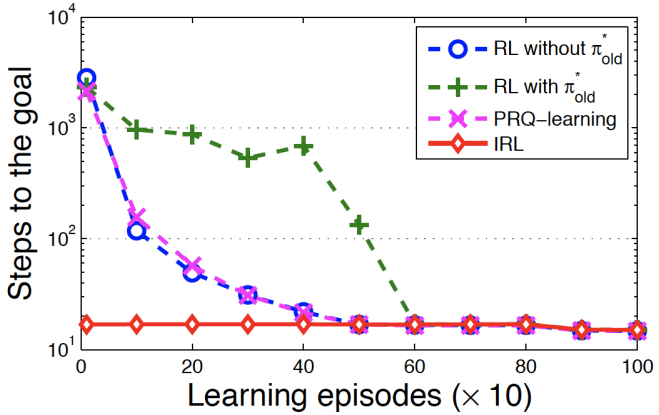
- A promising application for commercial automatic transportation in Amazon warehouses
- A robot lifts a pod, delivers it to a service station, and returns the pod back

A 12-robot simulation platform

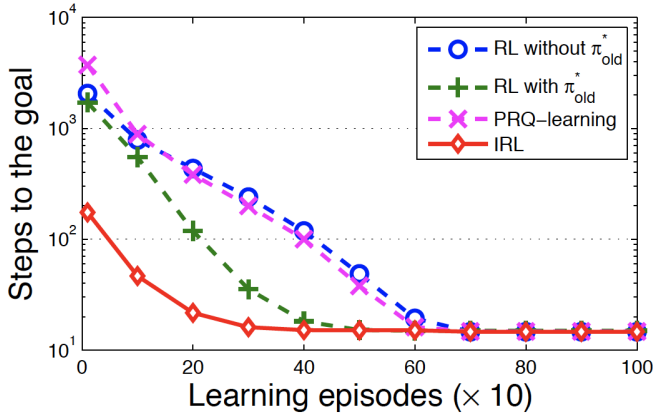
- The storage shelves and the picking stations may change over time
- The environment of a robot may change due to the movements of other robots

III. Intelligent Warehouse, Unmanned Logistics





(a) ϵ-greedy



(b) Softmax

Table of Contents

Conclusions

A systematic Incremental Reinforcement Learning algorithm

- The learning environment changes in the reward function
- Fully utilizes existing knowledge while weakening its conflict with new information
- Enables a fast adaptation, saving large computational resources

Limitations

- Only works in tabular-based RL architecture with discrete state-action spaces
- Hand-designed neighborhood degree m

THANK YOU.

Questions or discussions are welcome.

Zhi Wang

City University of Hong Kong
Email: *njuwangzhi@gmail.com*
Homepage: *heyuanmingong.github.io*