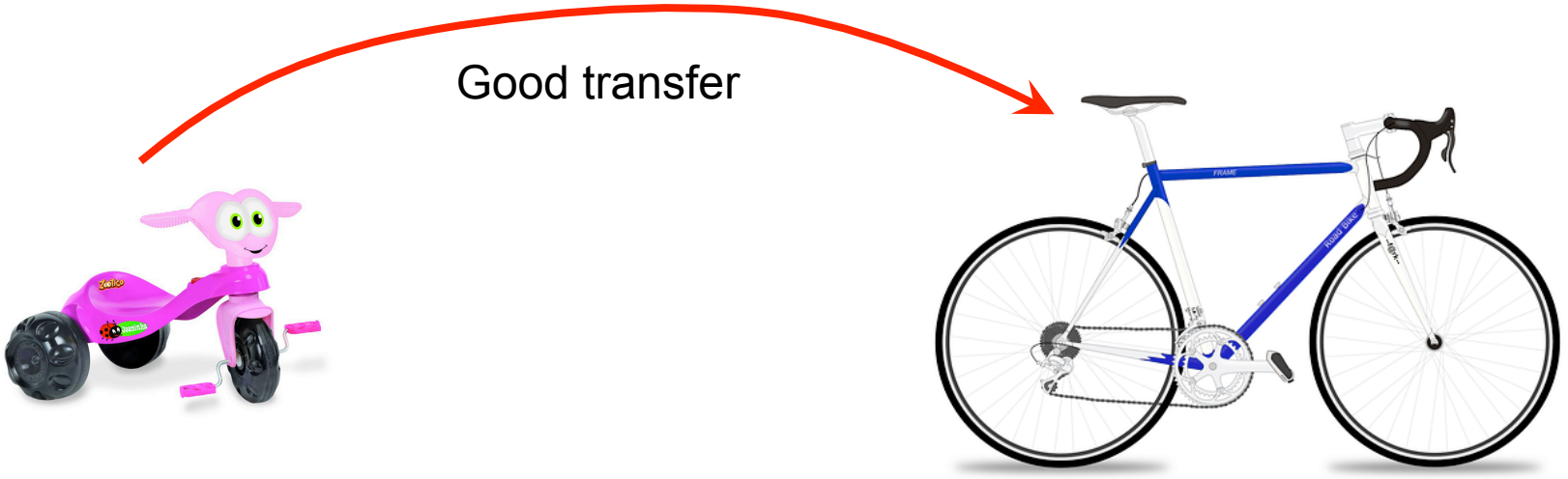
Anna Helena Reali Costa
Escola Politécnica, Universidade de São Paulo
Computer Engineering Department

TRANSFER OF KNOWLEDGE IN REINFORCEMENT LEARNING

Suppose we <u>know</u> how to ride a tricycle

Suppose we want to learn to ride a bicycle

How would knowing to ride a tricycle help in learning to ride a bike?



How would knowing to ride a tricycle help in learning to ride a bike?

Good transfer

Good transfer

Good transfer

Good transfer

Good transfer

Bad transfer

Can we design agents able to learn from experience and transfer knowledge across different problems to improve their learning performance?

Outline

- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

Outline

- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

Learning

- Learning modifies the agent's decision mechanisms to improve performance

Learning Agent

- Design of a learning agent depends on what feedback is available
- Type of feedback:

  Supervised learning: correct answers/output for each example/input (classification, regression)

  Unsupervised learning: no feedback is given (clustering)

  Reinforcement learning: occasional feedback

Learning Agent

- Design of a learning agent depends on what feedback is available
- Type of feedback:

  Supervised learning: correct answers/output for each example/input (classification, regression)

  Unsupervised learning: no feedback is given (clustering)

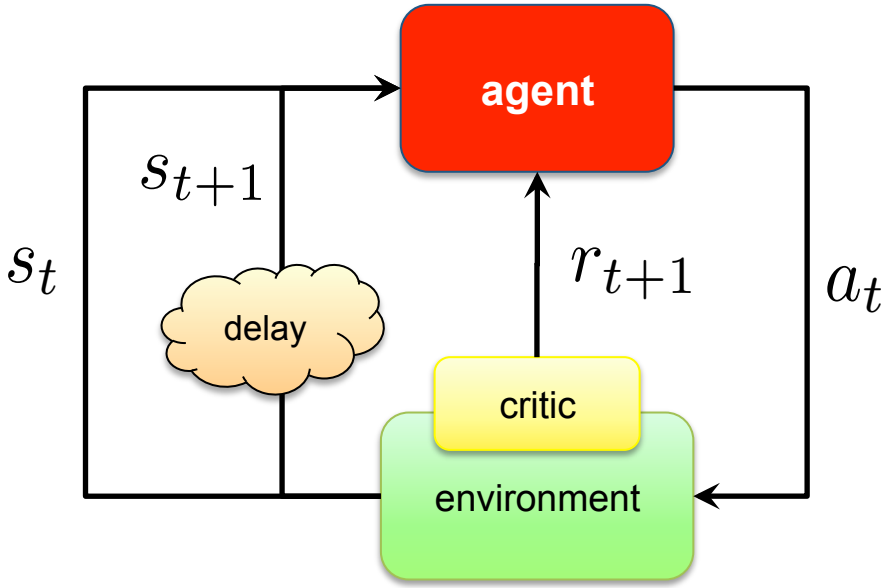
  Reinforcement learning: occasional feedback

Reinforcement Learning (RL)

- Feedback is delayed, occasional
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives
- Trial and error learning (via experiences)

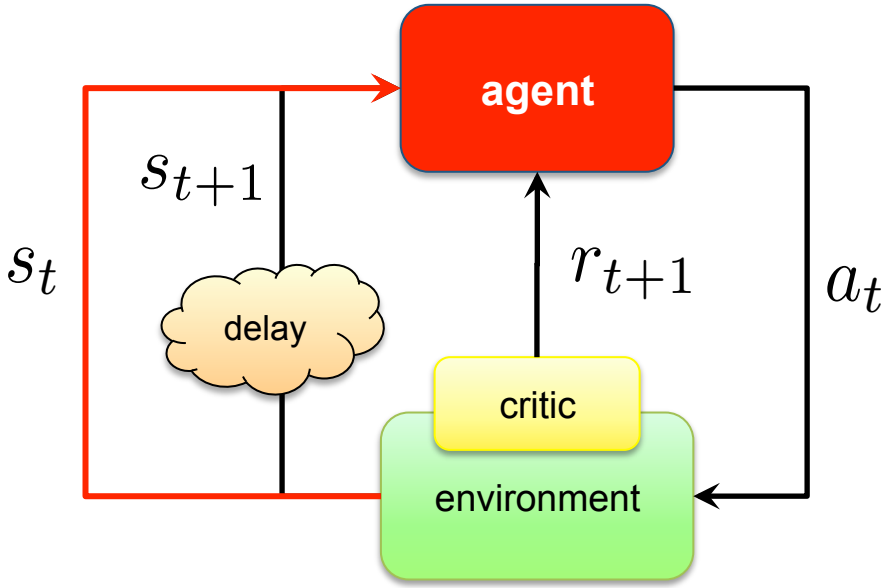
Task: Learn from this delayed reward to choose *sequences of actions* that produce the greatest cumulative reward

Reinforcement Learning



Sequential decision-making problems

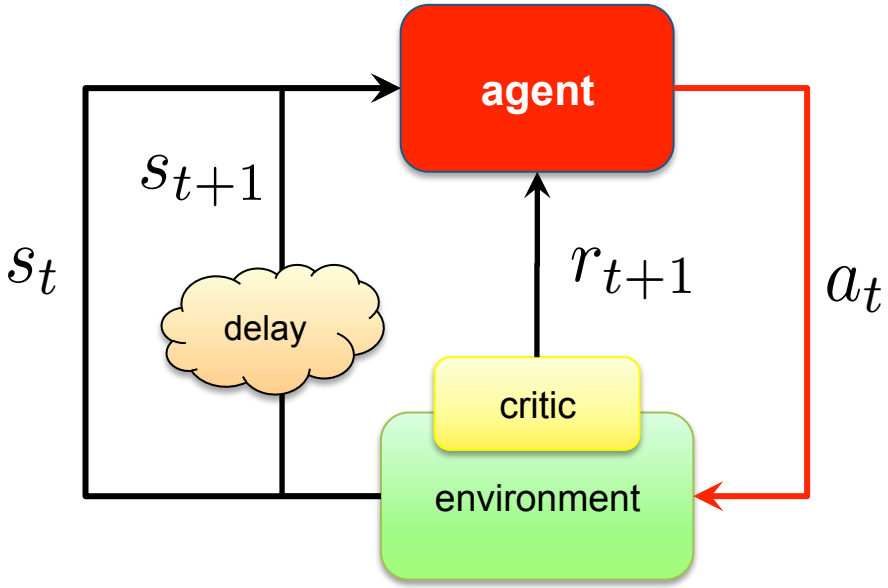
Reinforcement Learning



<position, speed>

S_t

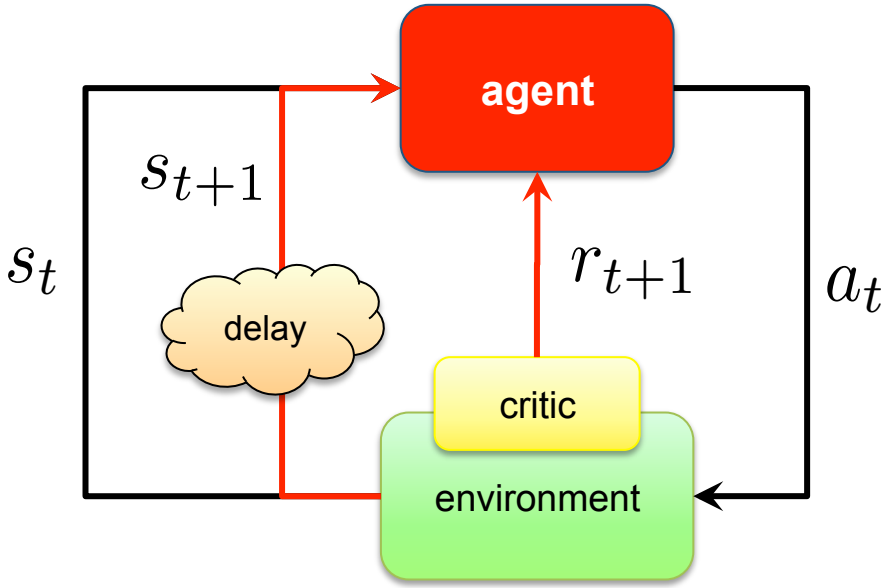
Reinforcement Learning



<position, speed> *<handlebar, pedals>*

s_t a_t

Reinforcement Learning



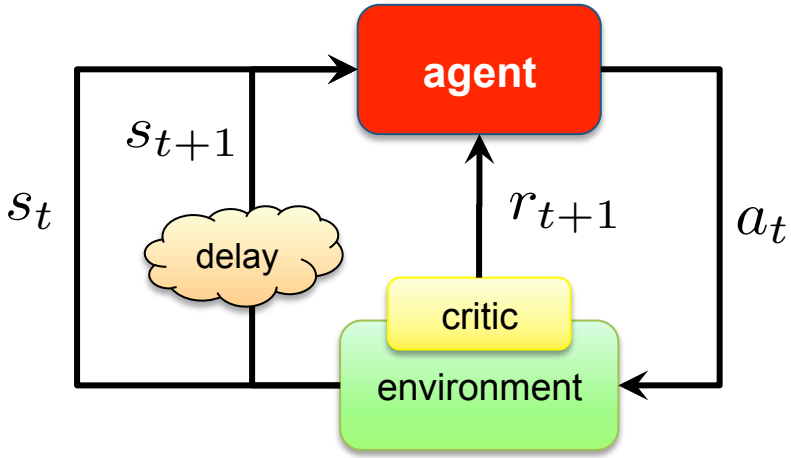
Experience:

$\langle s, a, s', r \rangle_1$

<position, speed> *<handlebar, pedals>* *<new position, new speed>, advancement*

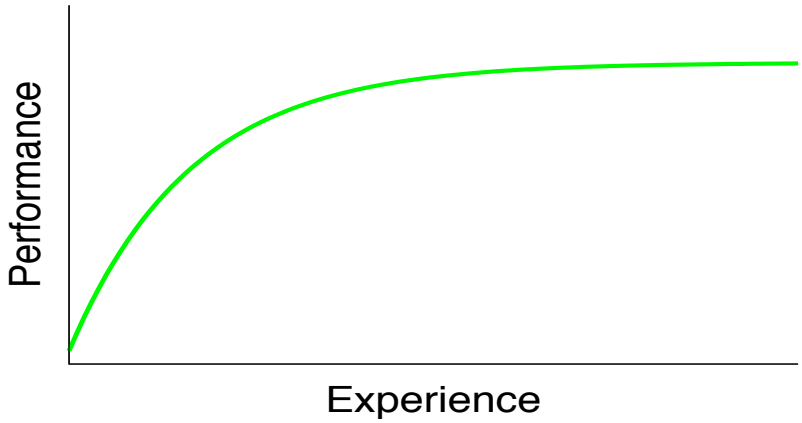
s_t a_t s_{t+1} r_{t+1}

Reinforcement Learning



$$\langle s, a, s', r \rangle_1$$
$$\langle s, a, s', r \rangle_2$$
$$\cdots$$
$$\cdots$$
$$\cdots$$
$$\langle s, a, s', r \rangle_n$$

Learning Curve



Agent learns to choose sequences of actions that produce the highest cumulative reward

Markov Decision Problem

- A Markov Decision Process (MDP) is:
  - -Set of states S
  - -Set of actions A
  - -Dynamics P(s'|s,a) – *probability of transition*
  - -Reward r(s,a)
- **Solution**: a policy, π: S → A, that maximizes

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t) \mid \pi\right]$$

RL method

- Key idea: updating the utility value Q(s,a) using the experience sequences

  - A trade off when choosing action between:
    - its immediately good (Exploitation)

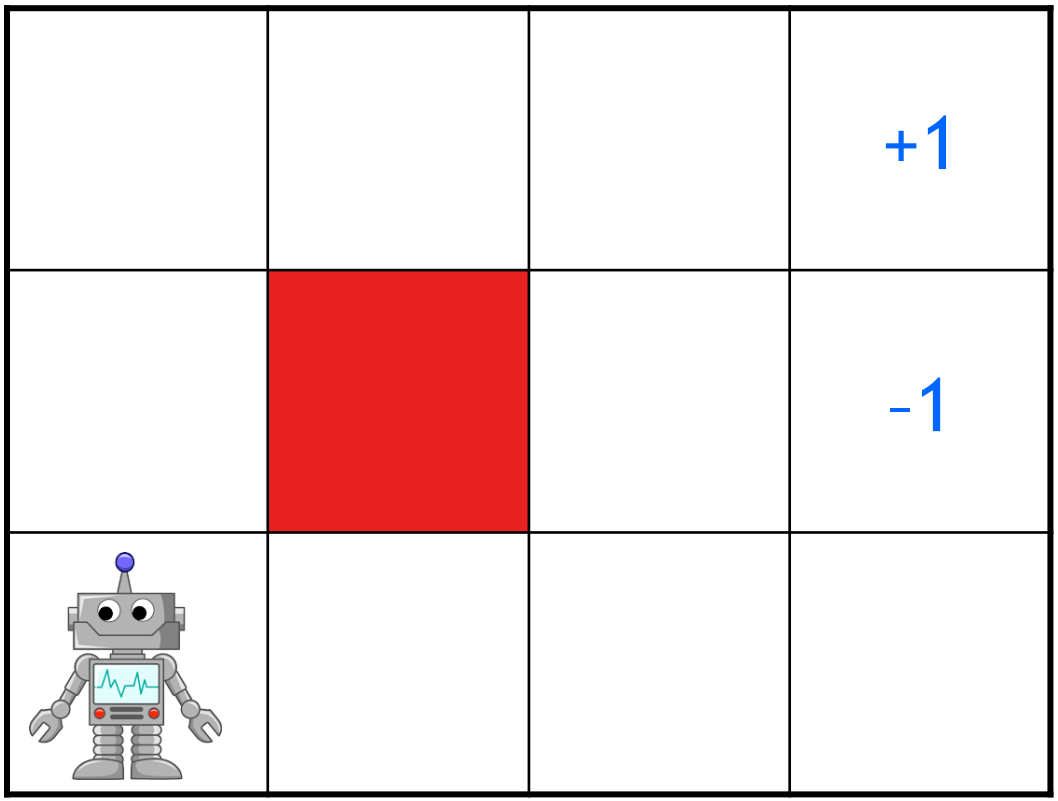
      ×

    - its long term good (Exploration)

RL methods

- Initialize Q(s,a) arbitrarily
- Observe the current state ***s***
- Do until a stop condition is reached:
  - select an action ***a*** and execute it in ***s***
  - receive immediate reward ***r***
  - observe the new state ***s'***
  - update value function Q(s,a) and policy π
  - s $\leftarrow$ s'

E.g.: Robot in a room



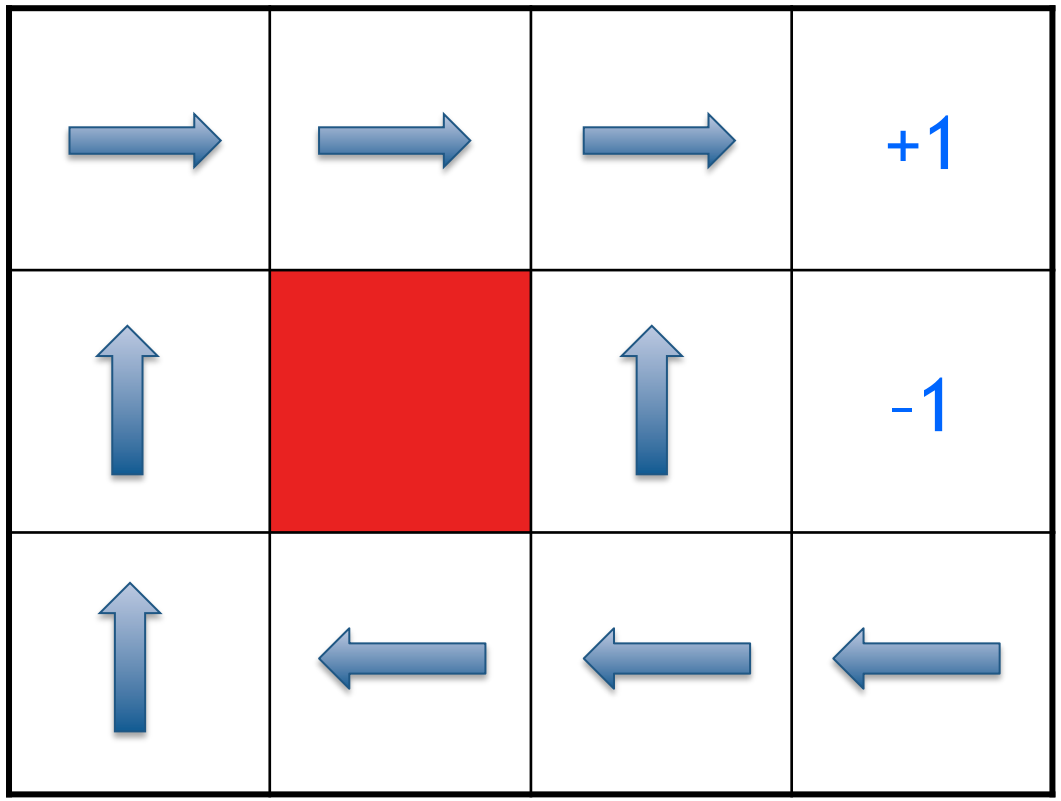
actions: UP, DOWN, LEFT, RIGHT

Stochastic actions: **UP**
80% move UP
10% move LEFT
10% move RIGHT

Reward:
+1 at [4,3]
-1 at [4,2]
-0.04 for each step

MDP: < S, A, P, r >

E.g.: Robot in a room



actions: UP, DOWN, LEFT, RIGHT

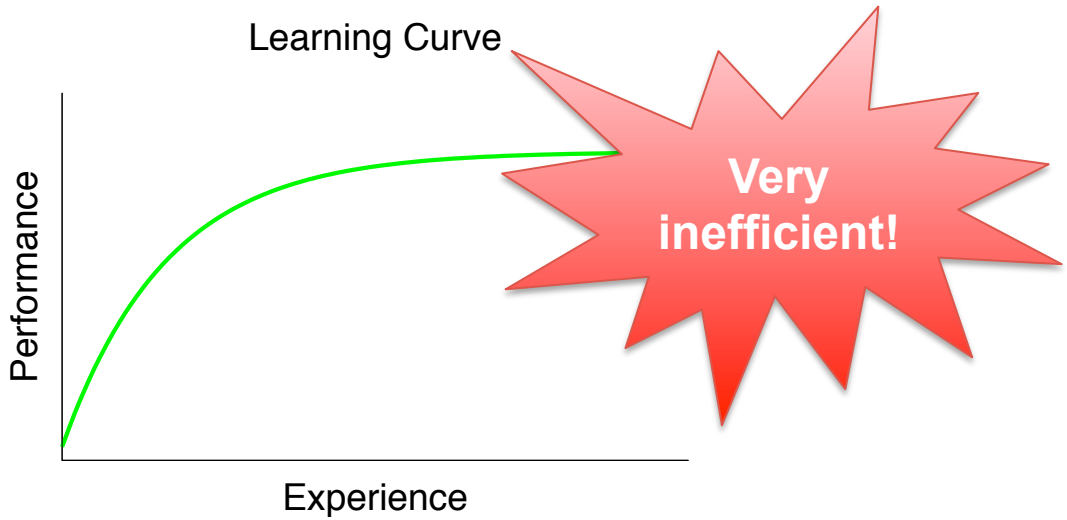
Stochastic actions: **UP**
80% move UP
10% move LEFT
10% move RIGHT

Reward:
+1 at [4,3]
-1 at [4,2]
-0.04 for each step

Optimal Policy

But…

- RL often requires **many samples**
- It takes **too long** to learn…
- The solution can usually only be applied to **one specific task** in a fixed setting

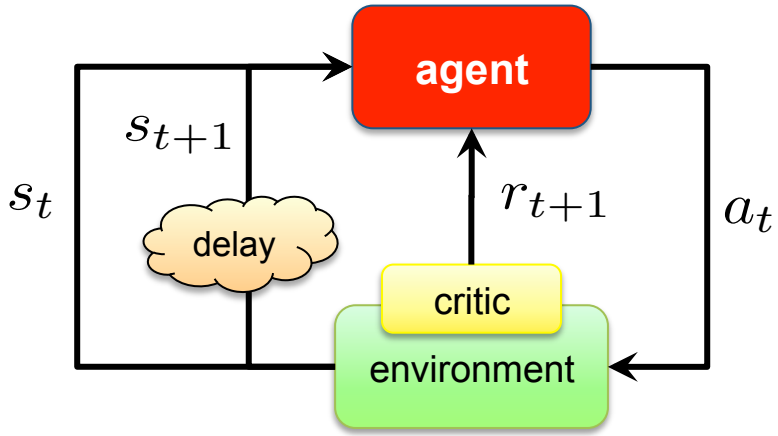


Outline

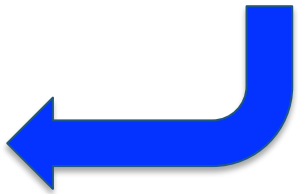
- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

Transfer in RL

New Task

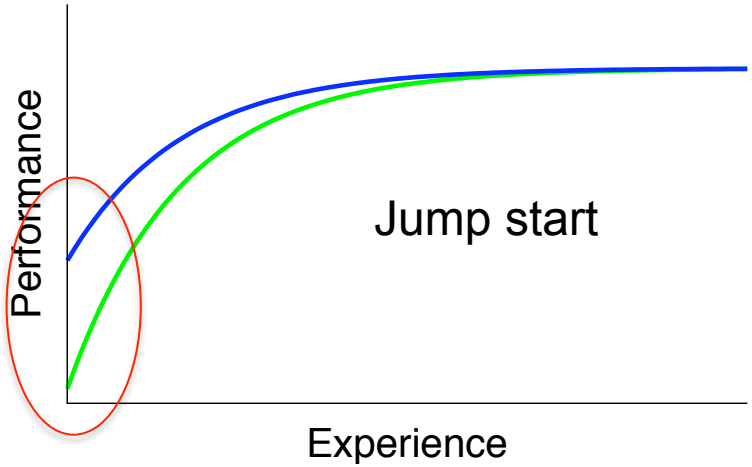


Previous Learned Task



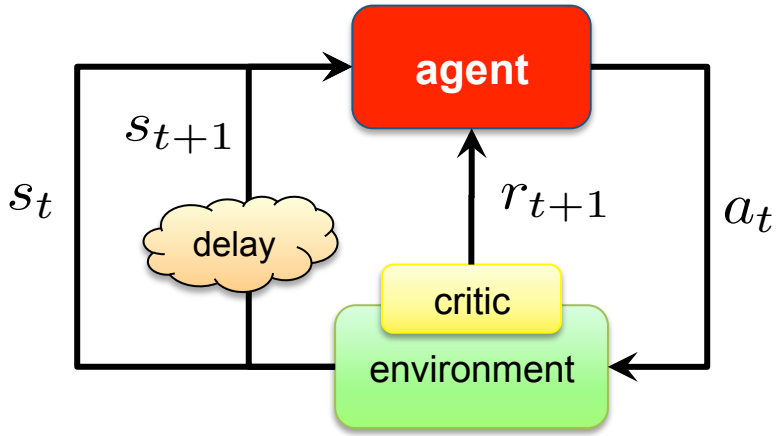
Knowledge Reuse

Offset Improvement

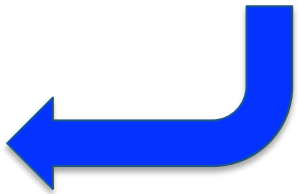


Transfer in RL

New Task

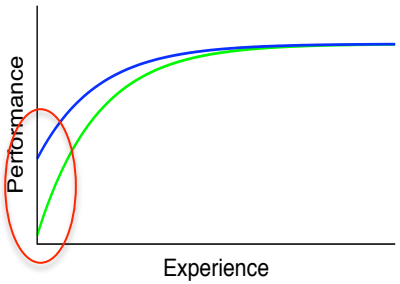


Previous Learned Task

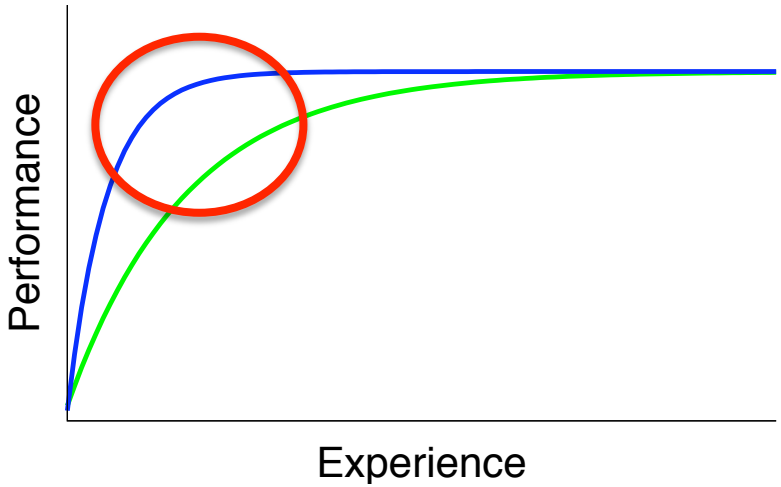


Knowledge Reuse

Offset Improvement



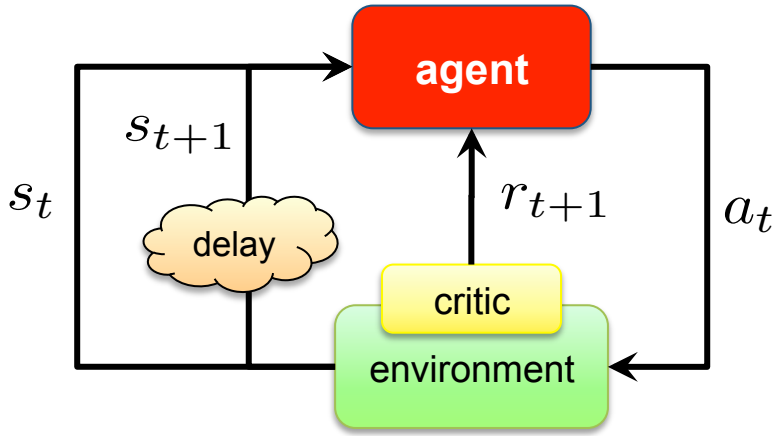
Speed Improvement



Transfer in RL

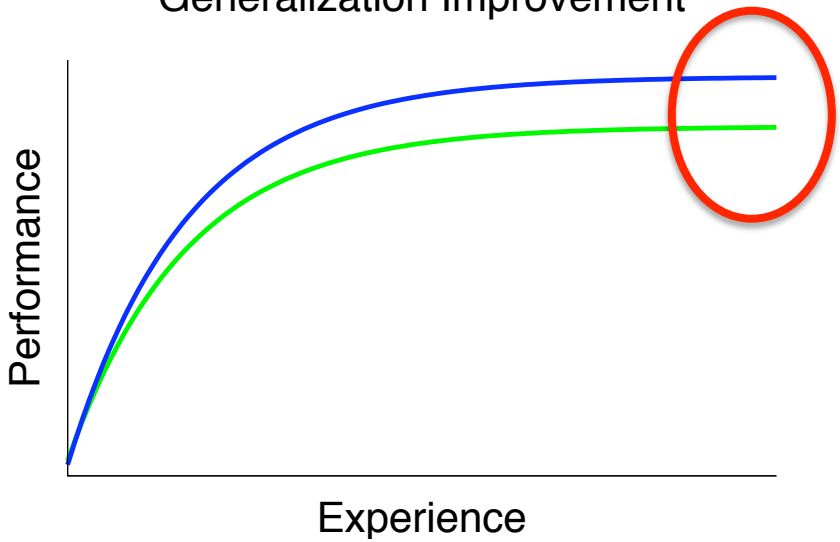
Previous Learned Task

New Task

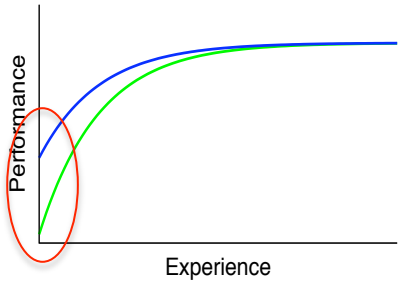


Knowledge Reuse

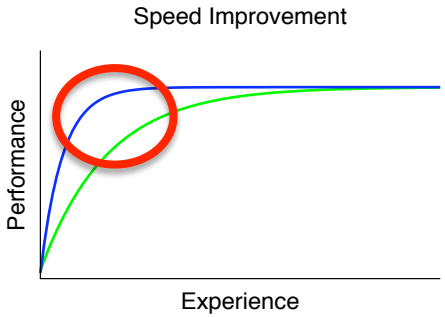
Generalization Improvement



Offset Improvement

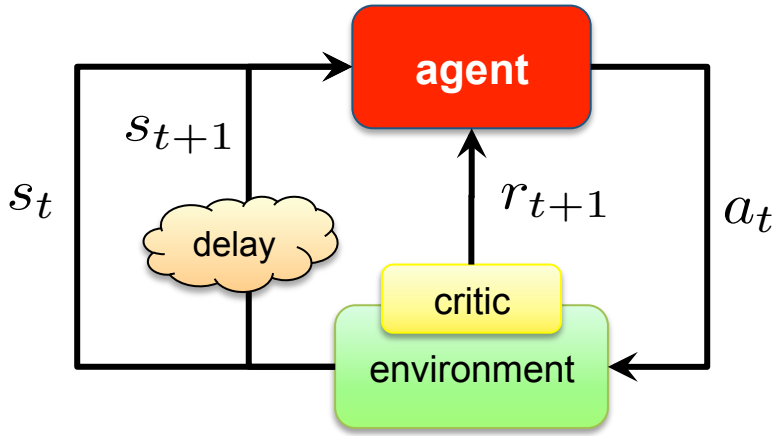


Speed Improvement

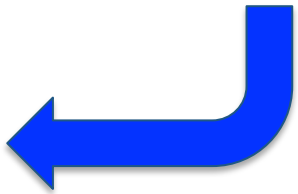


Transfer in RL

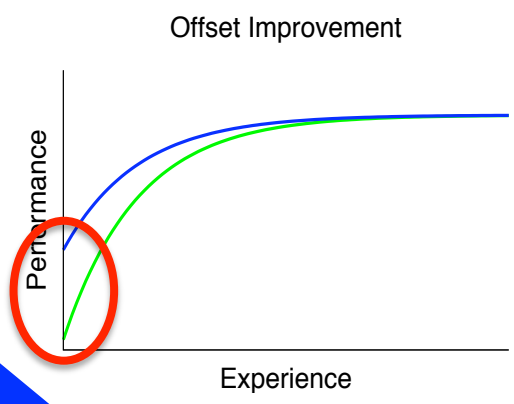
New Task

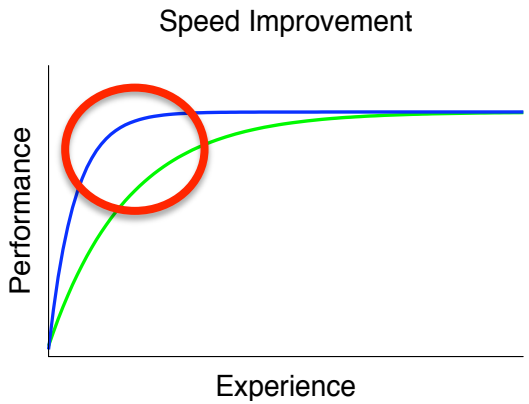


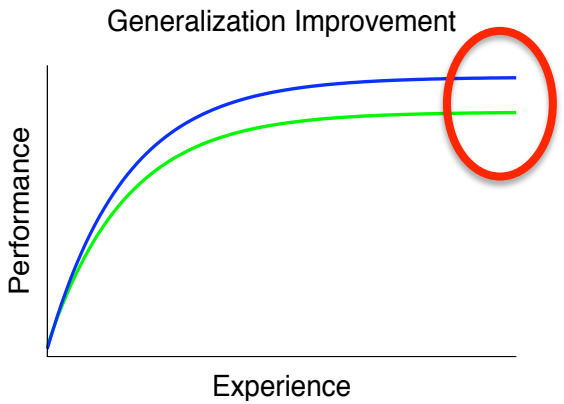
Previous Learned Task



Knowledge Reuse







Outline

- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

RL methods

- Initialize Q(s,a)
- Observe the current state ***s***
- Do until a stop condition is reached:
  - **select an action *a*** and execute it in ***s***
  - receive immediate reward ***r***
  - observe the new state ***s'***
  - update value function Q(s,a) and policy π
  - s ← s'

Heuristically Accelerated Learning – HAL

- **Proposal**: use heuristics in the choice of actions

$$\pi(s) = \begin{cases} \arg\max_a \hat{Q}(s,a) + \xi H_t(s,a)^\beta & prob. = \varepsilon, \\ a_{random} & prob. = 1 - \varepsilon \end{cases}$$

☺ Maintain convergence guarantee

☺ Good heuristics: fast convergence

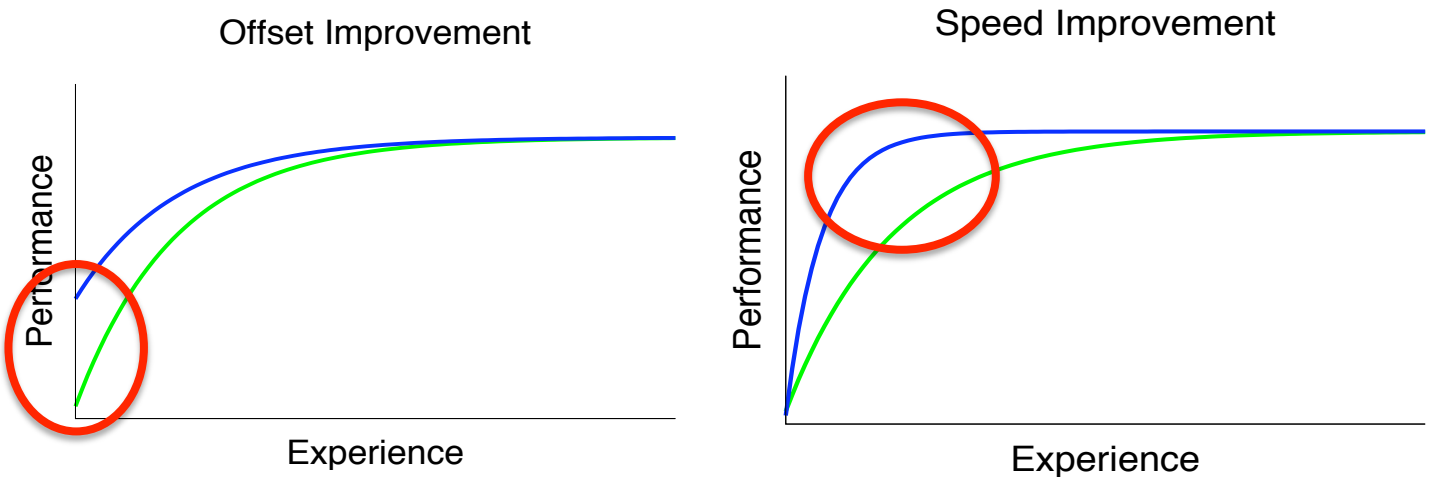
☹ With bad heuristics performance degrades,

☺ ... but the process recovers!!

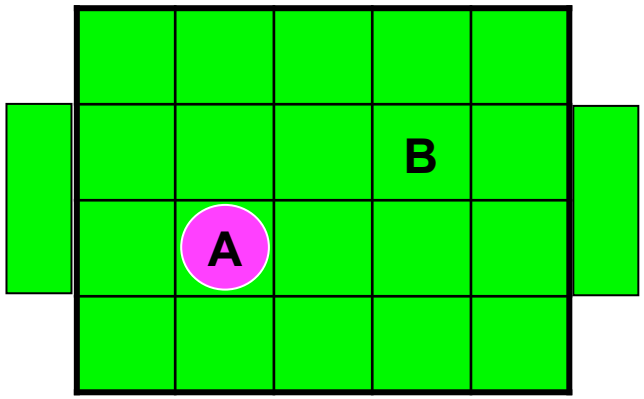
Heuristically Accelerated Learning – HAL

- **Proposal**: use heuristics in the choice of actions

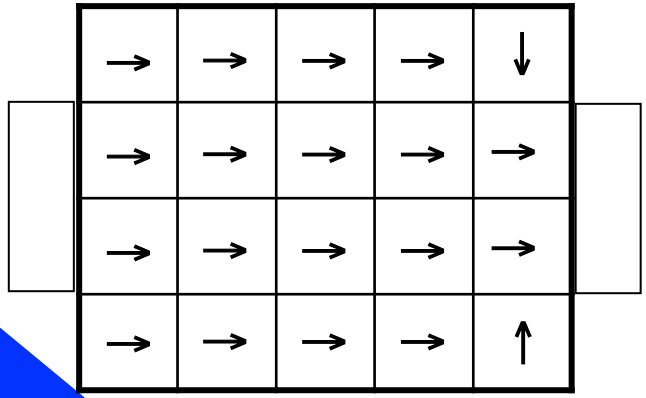
$$\pi(s) = \begin{cases} \arg\max_a \hat{Q}(s,a) + \xi H_t(s,a)^{\beta} & prob. = \varepsilon, \\ a_{random} & prob. = 1 - \varepsilon \end{cases}$$

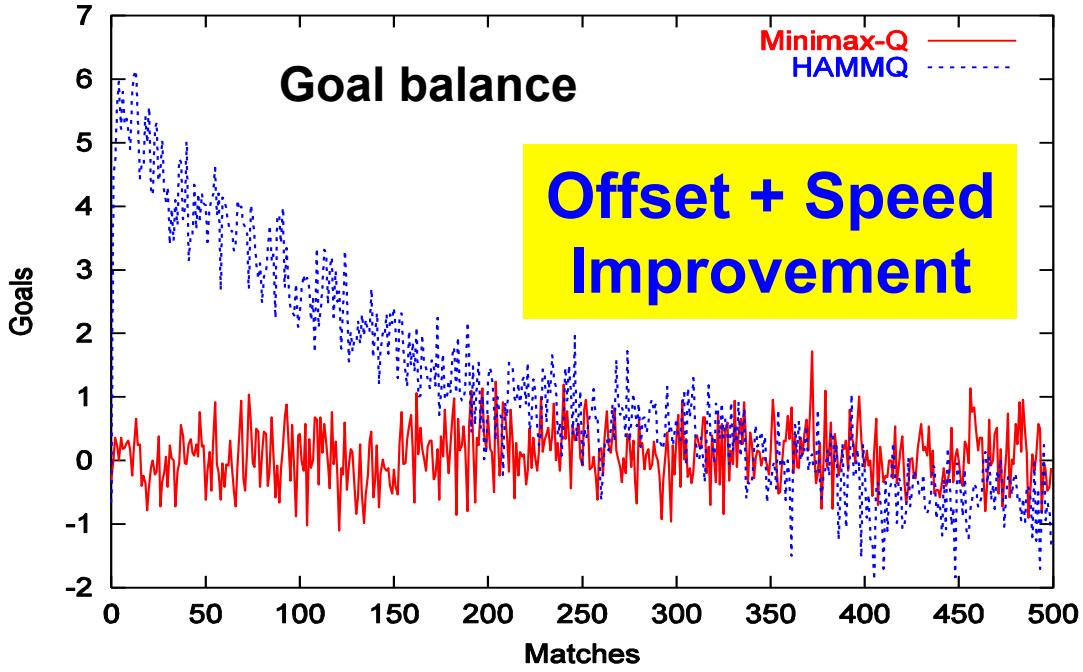


Results in soccer game with 2 players, both learning



Heuristics used:





BIANCHI, MARTINS, RIBEIRO, COSTA. ***IEEE Trans. on Cybernetics*** 2014
BIANCHI, RAC, RIBEIRO, CHC, COSTA, AHR. ***Journal of Heuristics***, 2007.
BIANCHI, COSTA, RIBEIRO. IJCAI 2007.

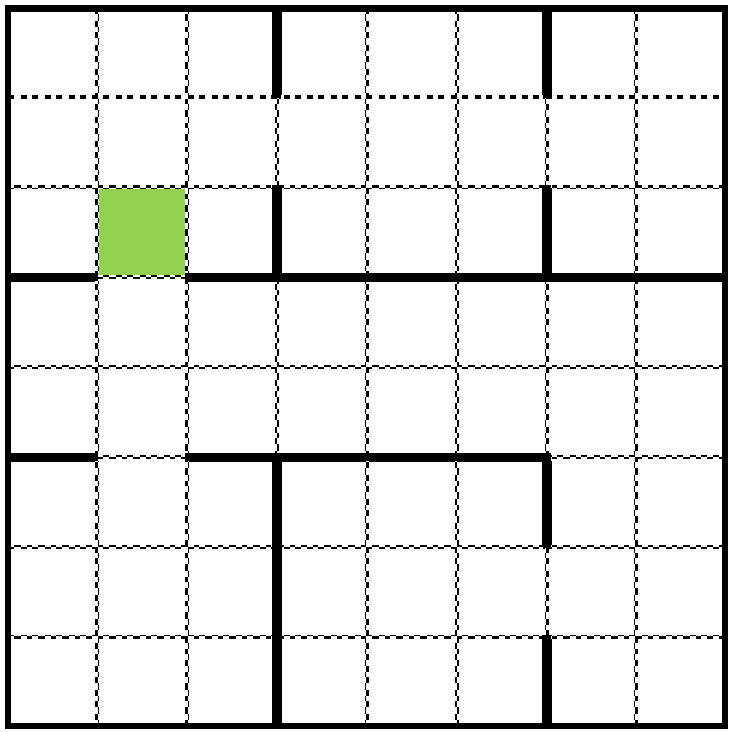
Outline

- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

Generalizing experiences

- The idea: improving the representation language
  - If we use a more powerful representation, we can generalize states and actions across tasks (and, therefore, generalize policies)

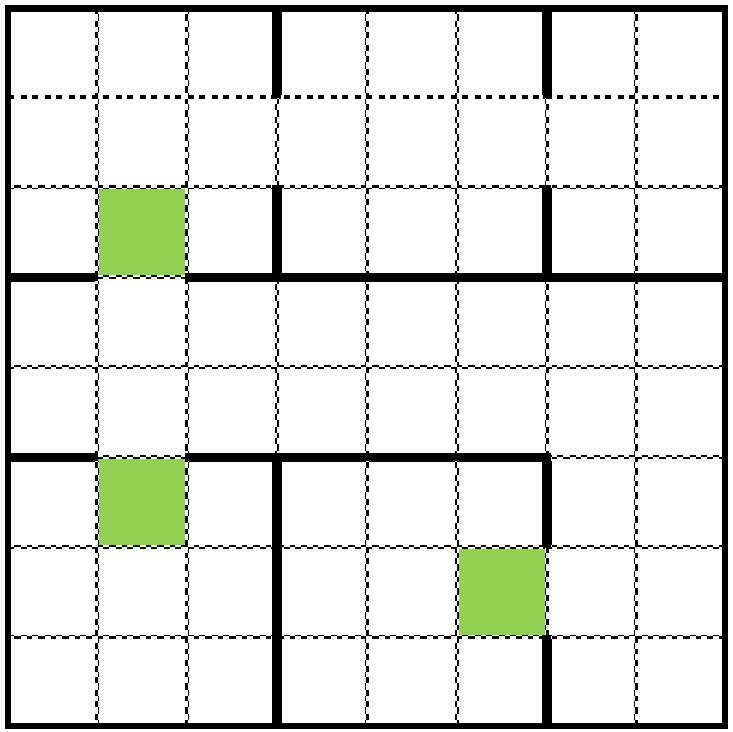
Representation: predicates



Ground state:

- inRoom(r_1)
- seeDoor(d_3)
- seeAdjCorridor(c_1)

Representation: predicates



Abstract state:

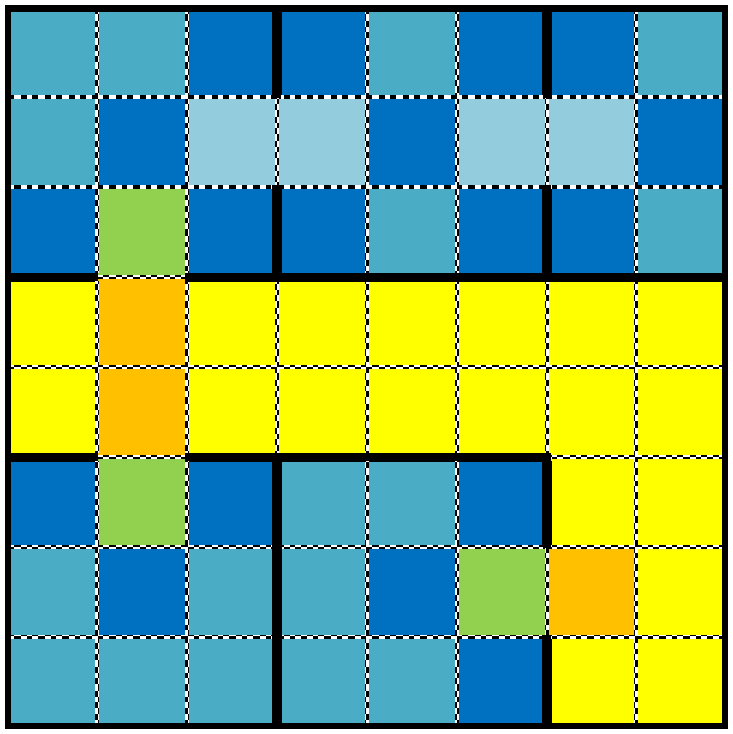
- inRoom(**X**)
- seeDoor(**Y**)
- seeAdjCorridor(**Z**)

Use of variables

1 abstract state comprises
a set of ground states

Relational MDP

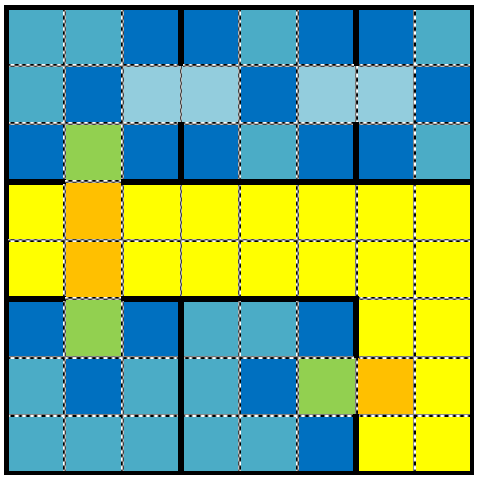
Abstraction → Generalization

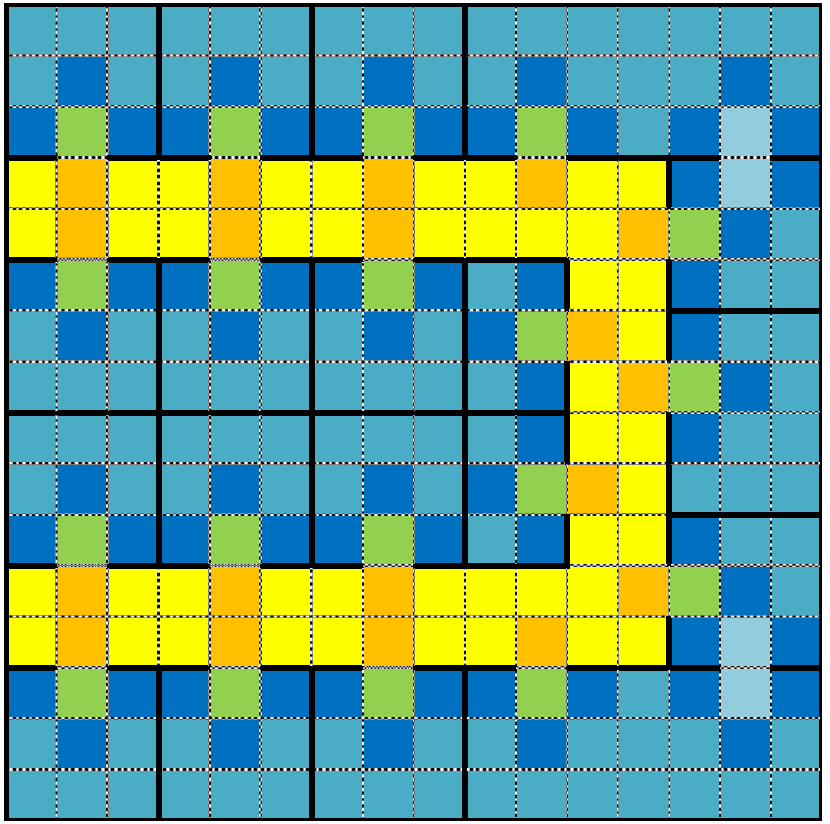


- Enables state aggregation
- Reduced space
- Enables transfer learning

Transfer

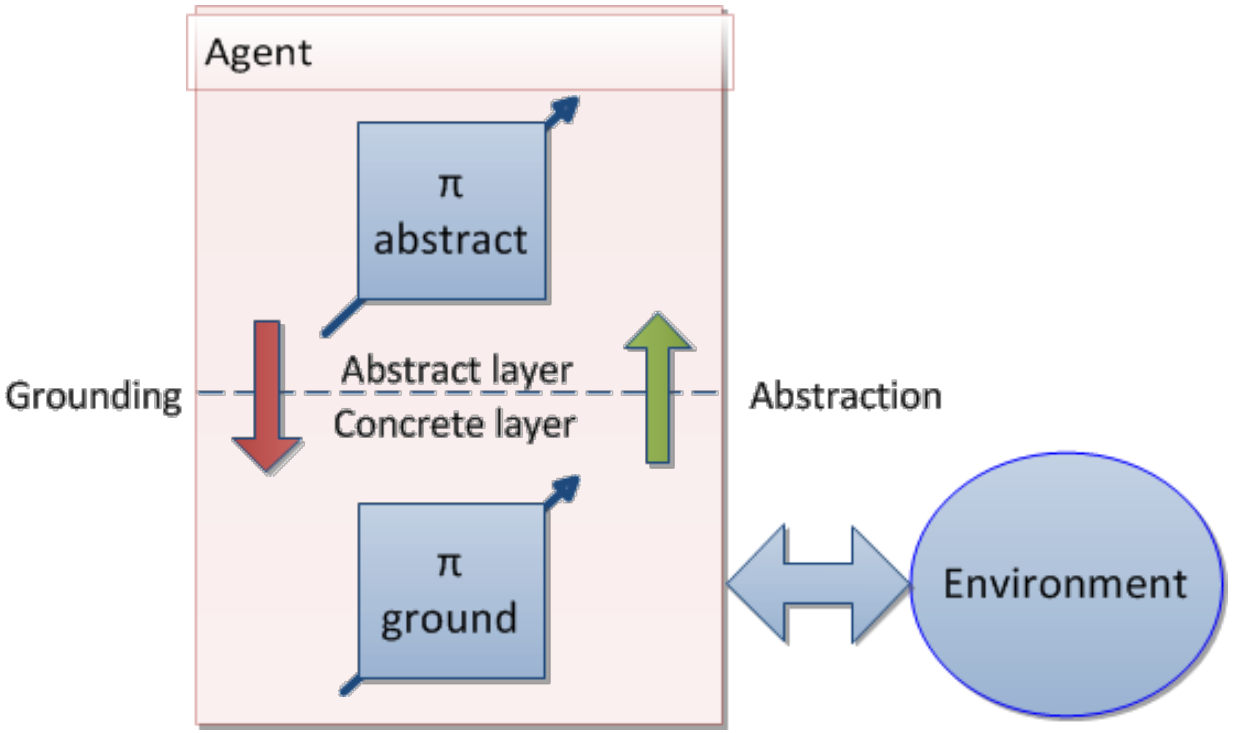
- Tasks described by the same predicates





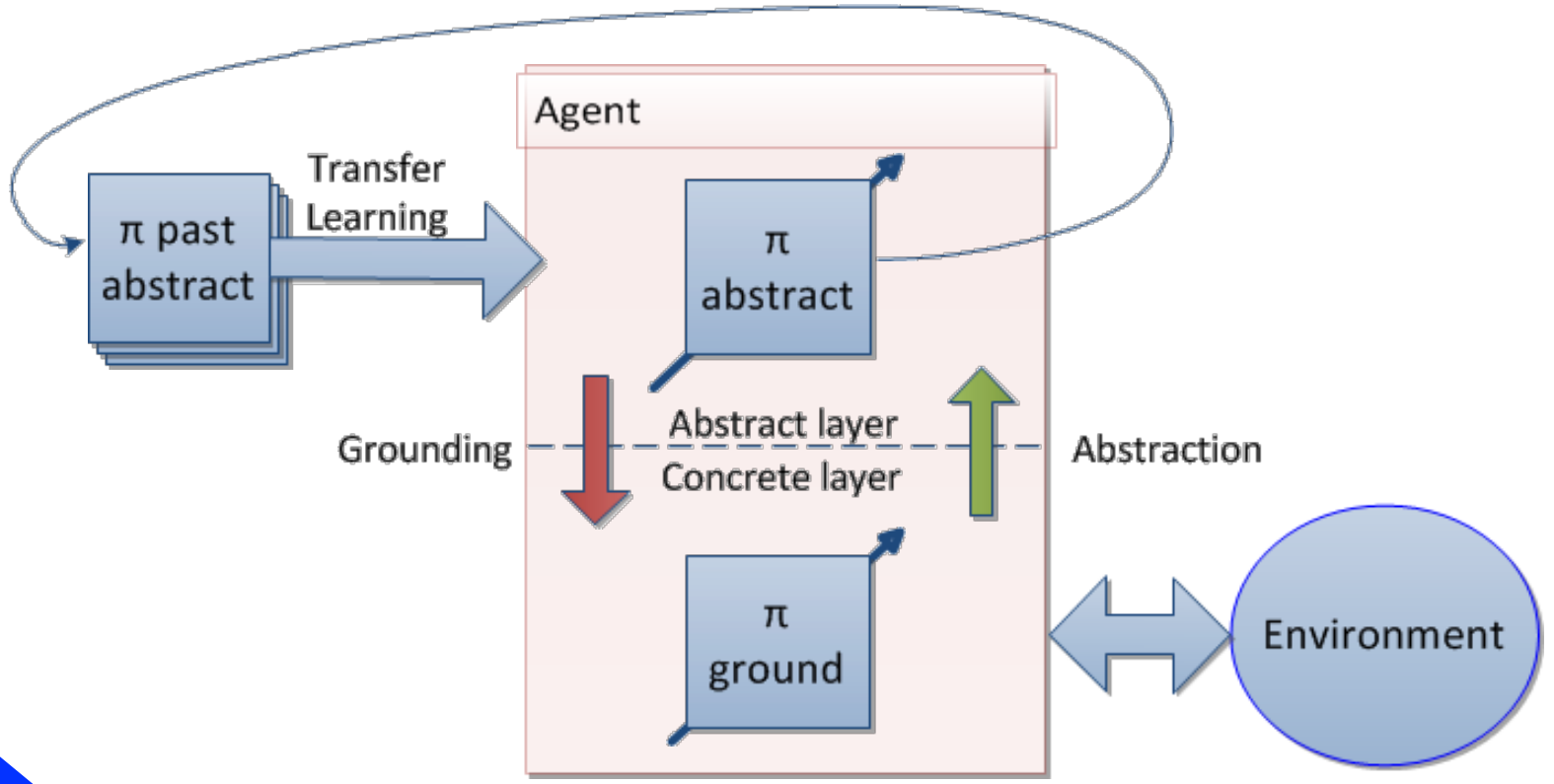
Agent Architecture

- Simultaneous 2-layer reinforcement learning (S2L-RL)

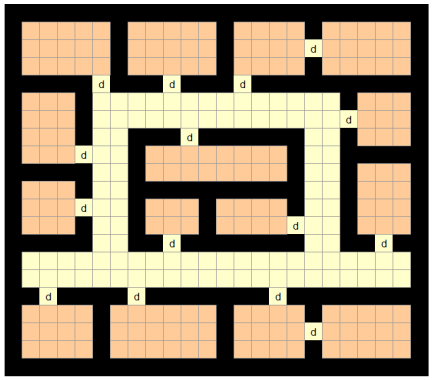


Agent Architecture

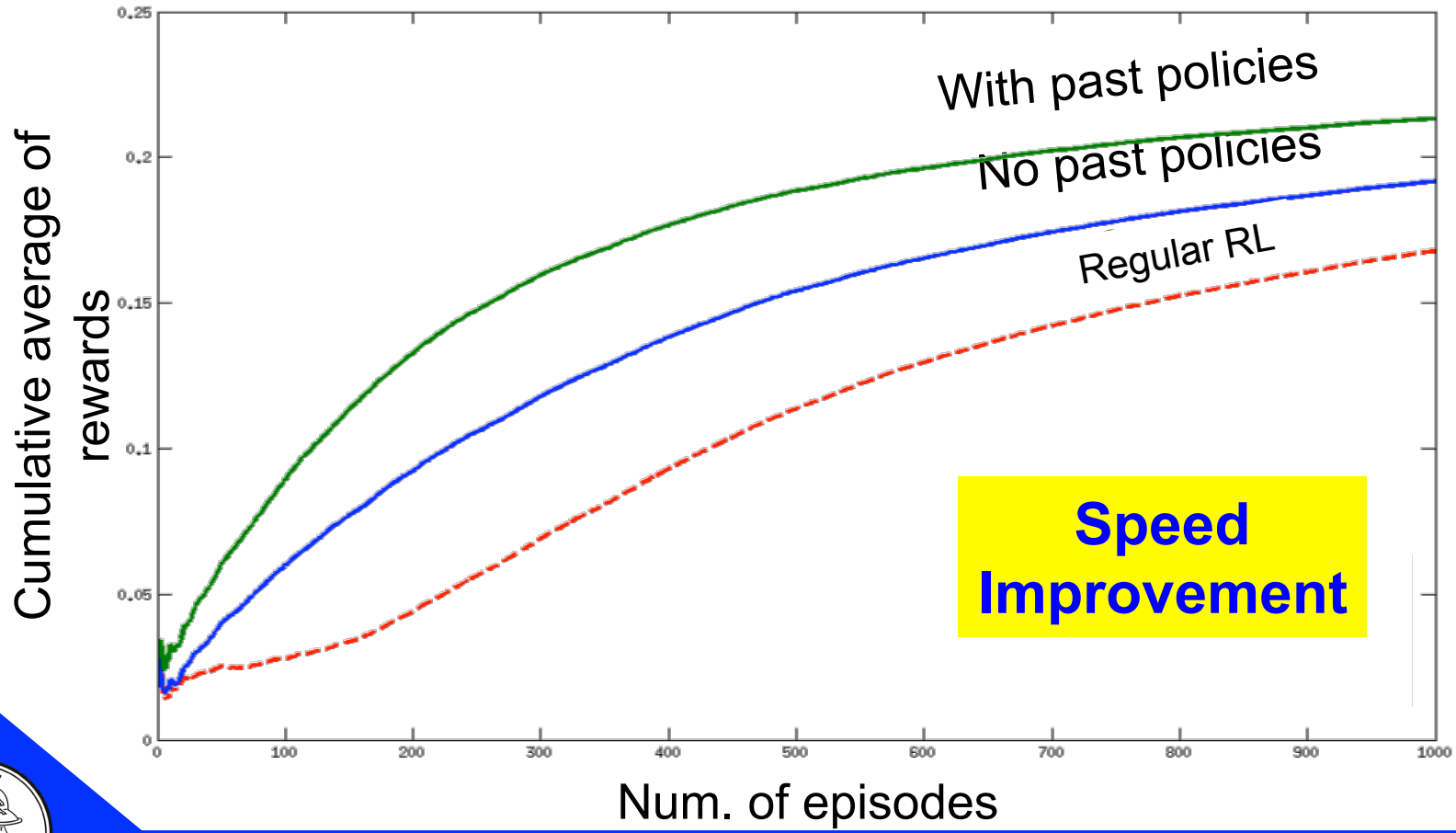
- Simultaneous 2-layer reinforcement learning (S2L-RL)



Results
Navigation task



KOGA, FREIRE, COSTA.
IEEE Trans. on Cybernetics 2015



Outline

- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Generalizing the Experience
- Initializing the value function
- Conclusions

RL algorithms

- **Initialize Q(s,a)**
- Observe the current state ***s***
- Do until a stop condition is reached:
  - select an action ***a*** and execute it in ***s***
  - receive immediate reward ***r***
  - observe the new state ***s'***
  - update value function Q(s,a) and policy π
  - s $\leftarrow$ s'

PITAM: Probabilistic Inter-TAsk Mappings

- Autonomously Define an Inter-Task Mapping
- Use of Object-Oriented MDPs
- Weight Q-value according to PITAM weight and initiate Q-table

$$\mathcal{X}(x_{target}) = x_{source}$$

Inter-task Mapping

State variables

$\mathcal{X}$

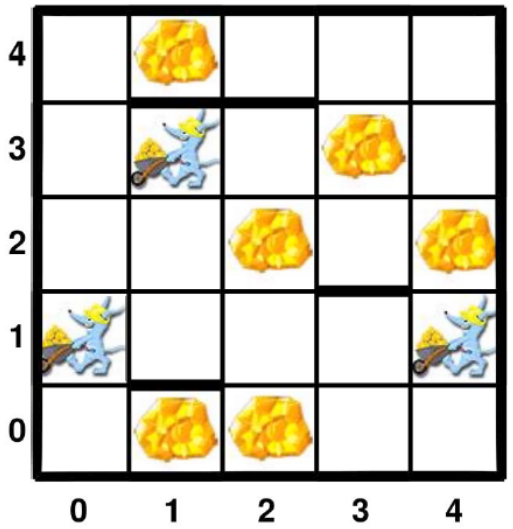
Usually specified by humans

PITAM: autonomously defined by exploring the OO representation

OO-MDP

- **Classes**: "Types" of objects
- **Attributes**: set of attributes composes a class
- **Objects**: Entities in the environment that belong to a class and have valuations for each attribute

Goldmine Domain

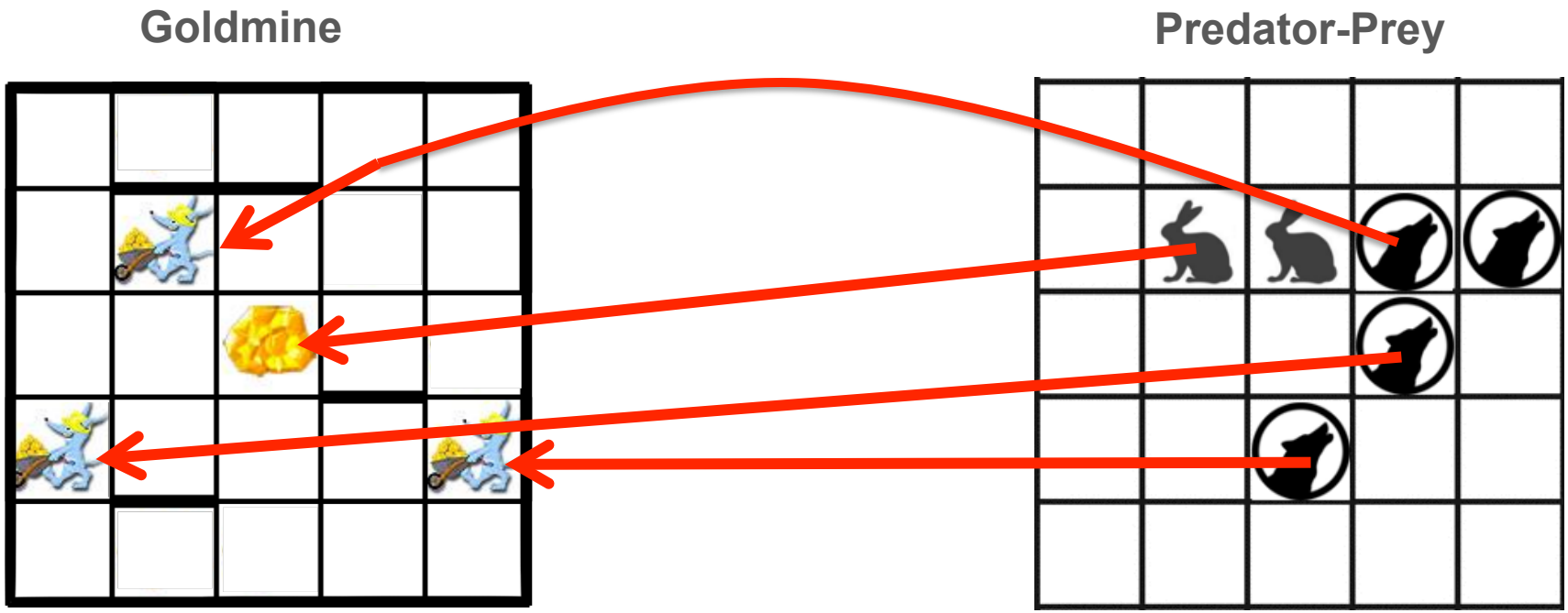


Classes: {Miner, Gold, Wall}

Object *id*	Attributes
$miner1$	$x = 0, y = 1$
$miner2$	$x = 1, y = 3$
$miner3$	$x = 4, y = 1$
$gold1$	$x = 1, y = 0$
	$\vdots$
$gold6$	$x = 4, y = 2$
$wall1$	$x = 1, y = 1,$ $pos = South$
	$\vdots$
$wall24$	$x = 4, y = 4,$ $pos = East$

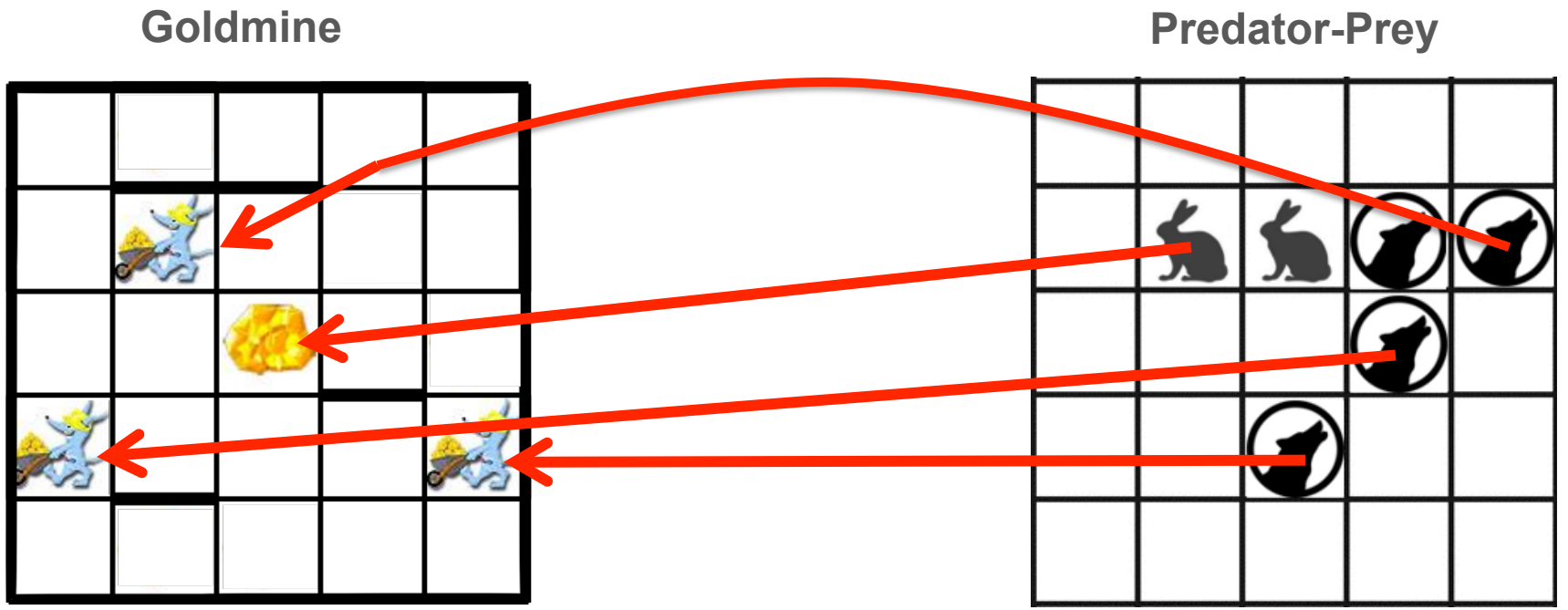
PITAM

- Mapping Prey to Gold, and Predator to Miner



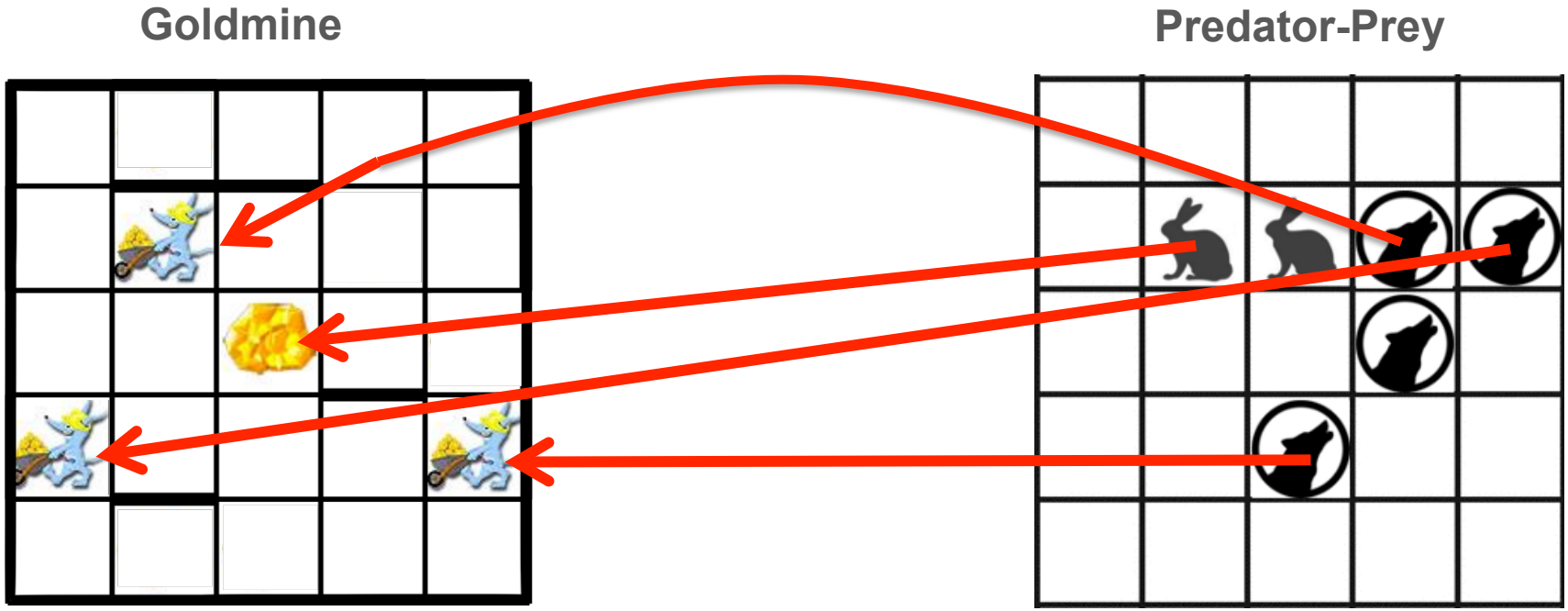
PITAM

- Mapping Prey to Gold, and Predator to Miner



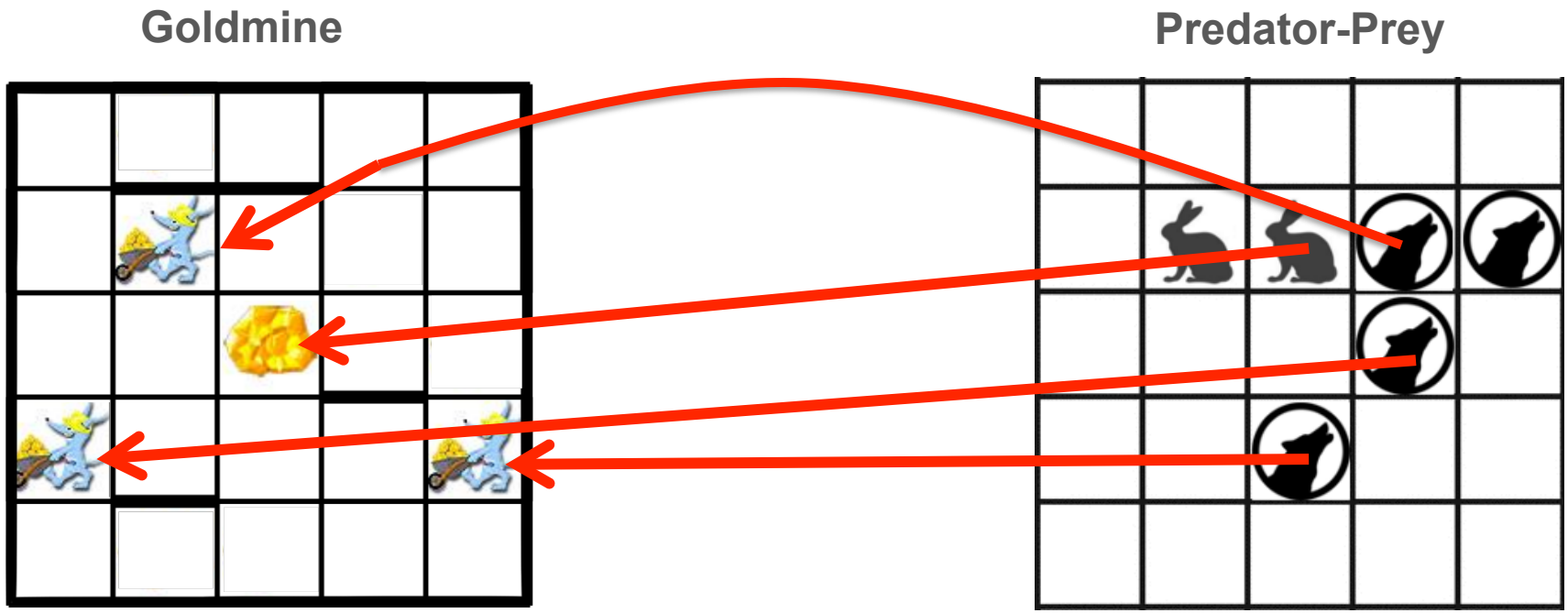
PITAM

- Mapping Prey to Gold, and Predator to Miner



PITAM

- Mapping Prey to Gold, and Predator to Miner



Results

Source Task: Goldmine Domain (simpler)
Target Task: Predator-Prey

Regular Learning: RL-learning from scratch

QManualMapping: Hand-coded Mapping using Q-value reuse (Taylor et al., JMLR 2007)

PITAM: our proposal

Steps to conclude the Task:

	Regular	QManualMapping	PITAM
offset	76.21	51.22	**30.48**
generalization	45.67	45.92	**29.48**

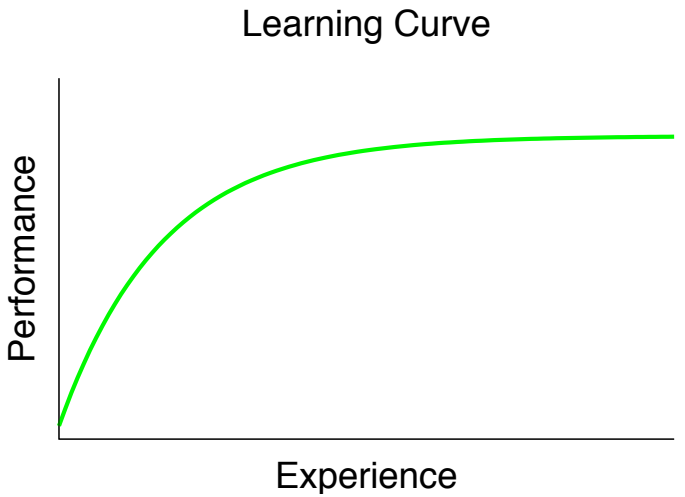
SILVA, F.L.; COSTA, AHR. TIRL2017.

Outline

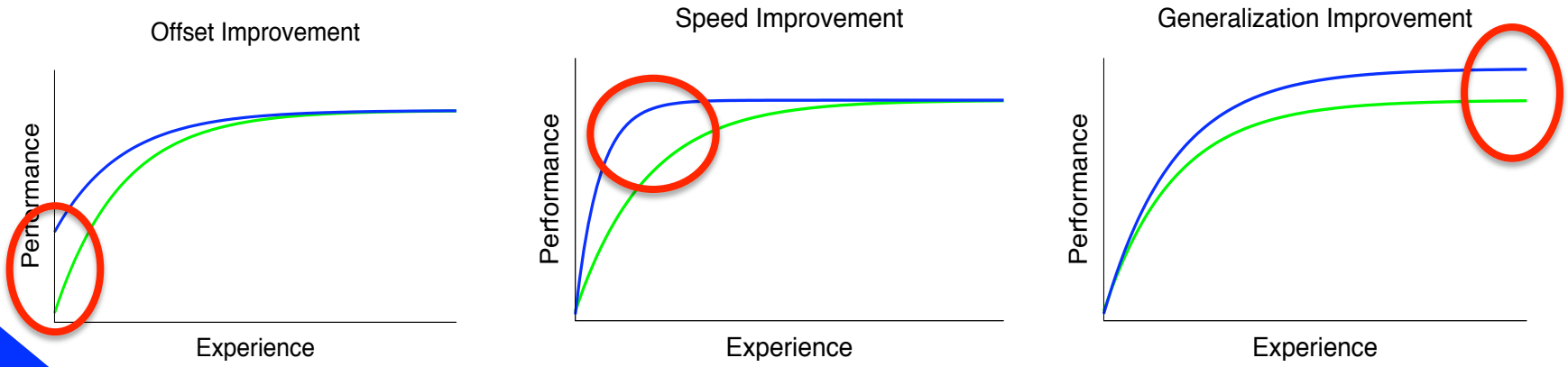
- Reinforcement Learning (RL)
- Transfer in RL
- Improving the Exploration Strategy
- Spreading the Experience
- Initializing the value function
- Conclusions

Conclusions

Without Transfer



With Transfer



Some Real-world Applications

- Controlling Gene Regulatory Networks
  - » BRACIS 2016
- Energy Management Systems for Smart Homes
  - » IJCAI 2015
- Recommender Systems
  - » JBCS2015
- Coordination of EV Charging
  - » On-going work

Future Work

- Explore methods of approximation to handle real-world problems
- Explore languages to represent problems and solutions
- Automatic attribute discovery to better represent problems and transfer of knowledge
- Automatic discovery of similarity between tasks

Thanks!!

Anna Helena Reali Costa
Escola Politécnica
Universidade de São Paulo
anna.reali@usp.br