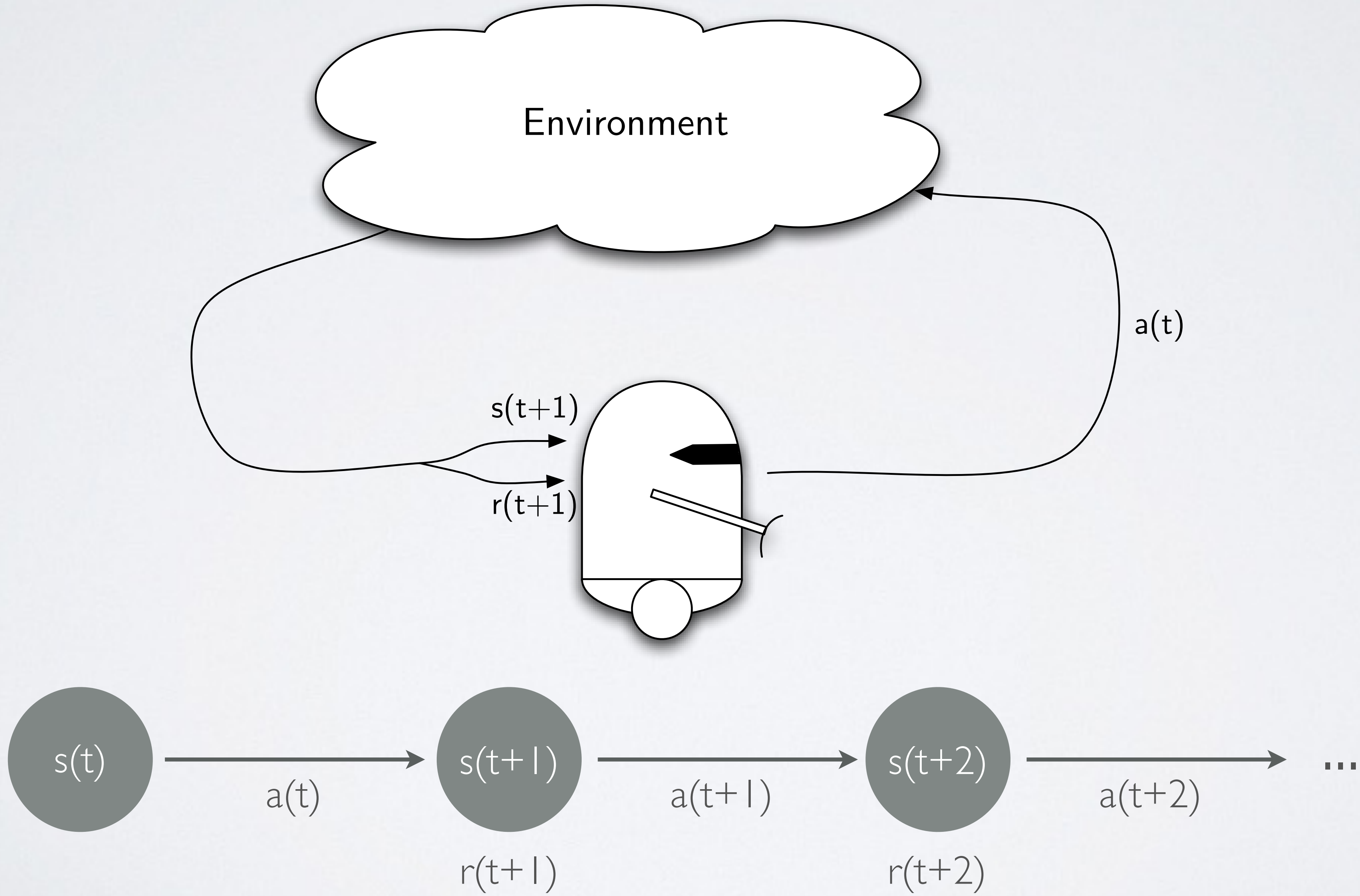
MULTI-AGENT REINFORCEMENT LEARNING

Sparse Interactions

REINFORCEMENT LEARNING

- Agent acting in an unknown environment, learning to maximise a numerical reward signal



MARKOV DECISION PROCESS

- SINGLE AGENT!!!

$$M = \langle S, A, T, R \rangle$$

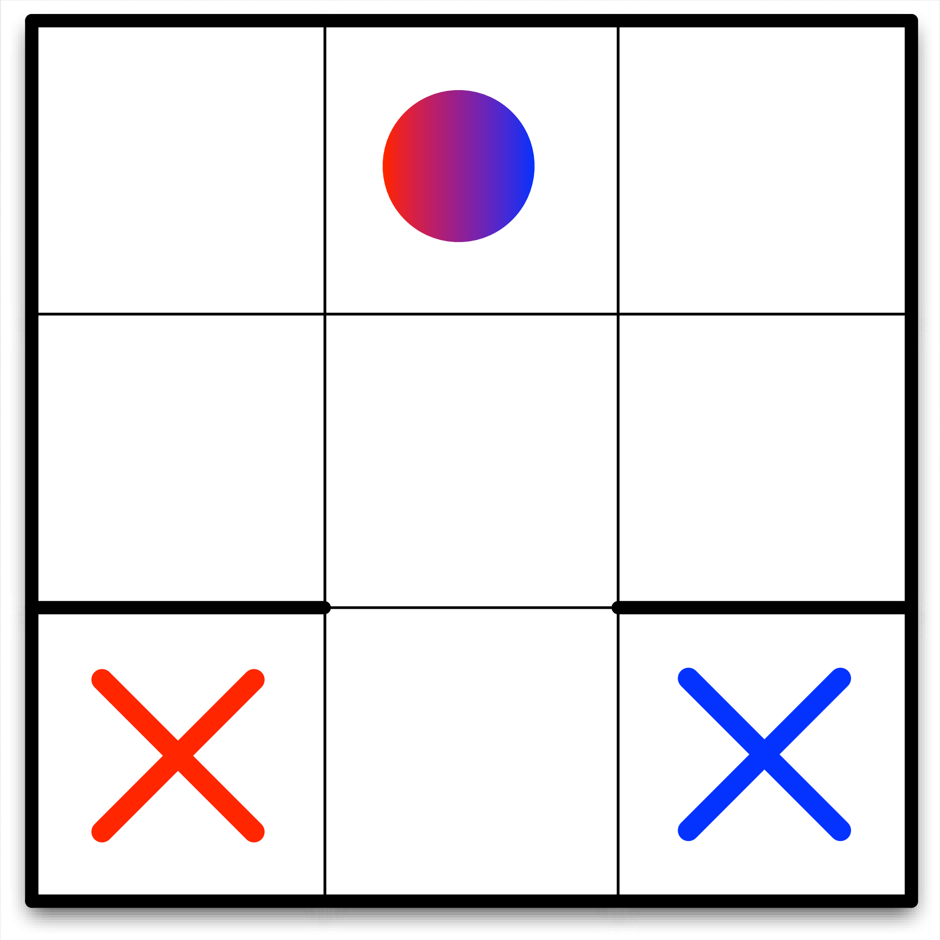
- States S = set of states of the agent
- Actions A = set of actions the agent can take
- Transition function $T : S \times A \rightarrow S$
- Reward function $R : S \times A \times S \rightarrow \mathbb{R}$

Q-LEARNING

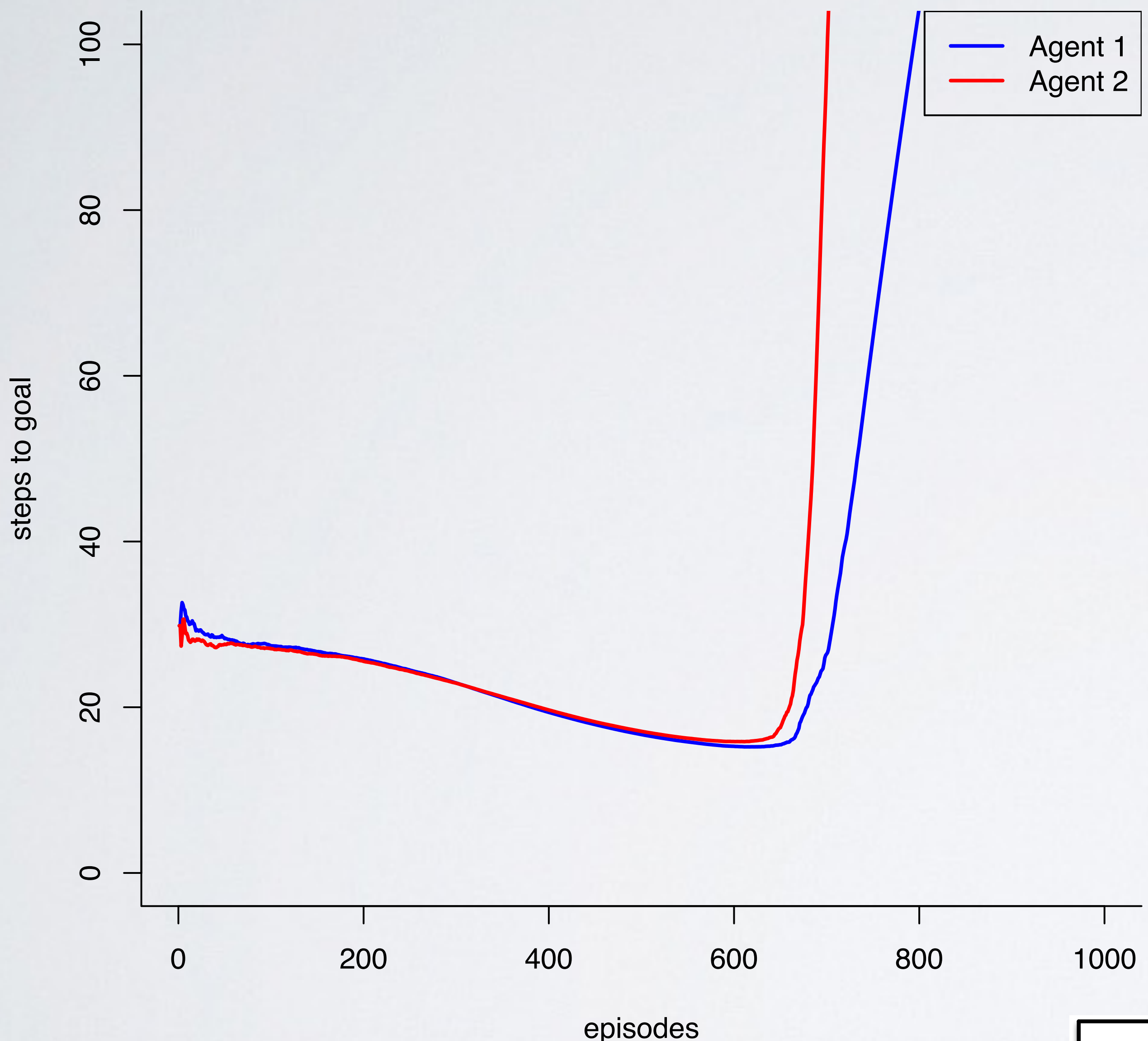
- model-free, reinforcement learning algorithm
- Stores Q-values for every state-action pair
- Update rule:

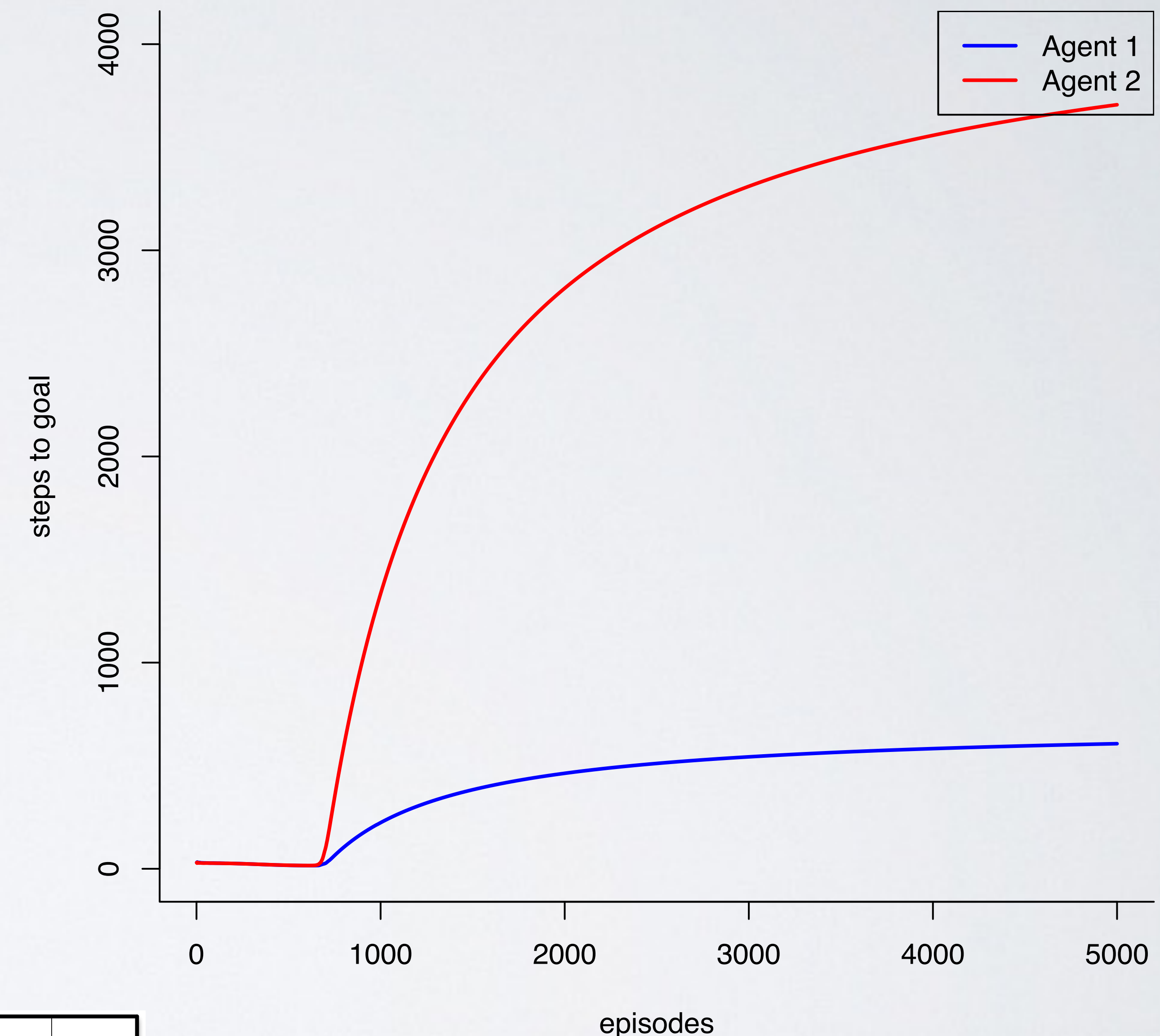
$$Q(s,a) = Q(s,a) + \alpha \left(r_t + \gamma \underset{a'}{\operatorname{argmax}} Q(s',a') - Q(s,a) \right)$$

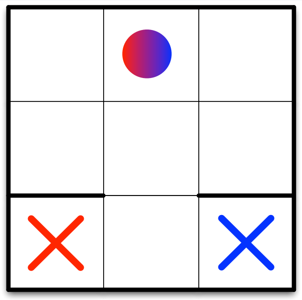
SIMPLE EXAMPLE



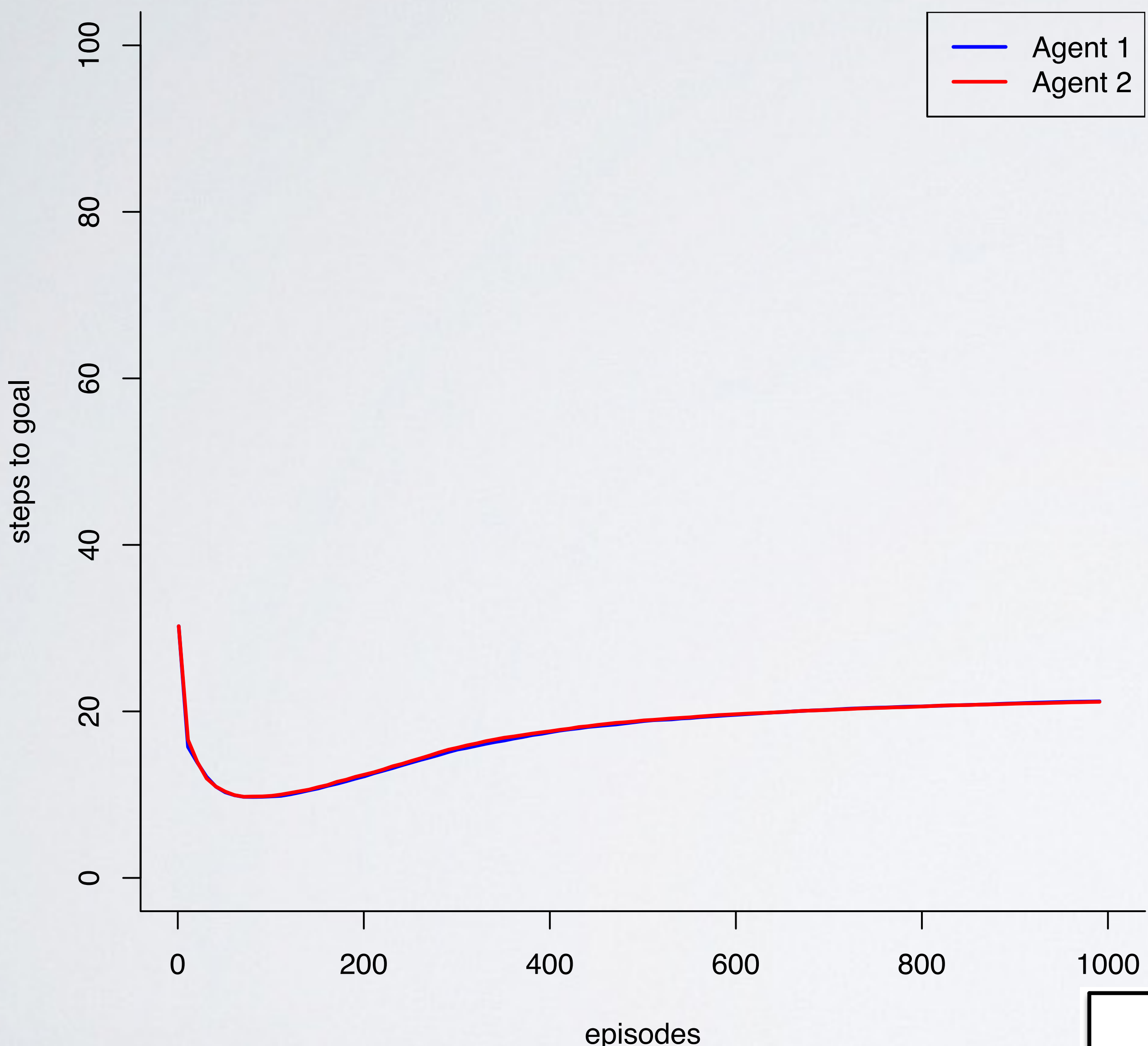
RL WITH BOLTZMANN EXPLORATION

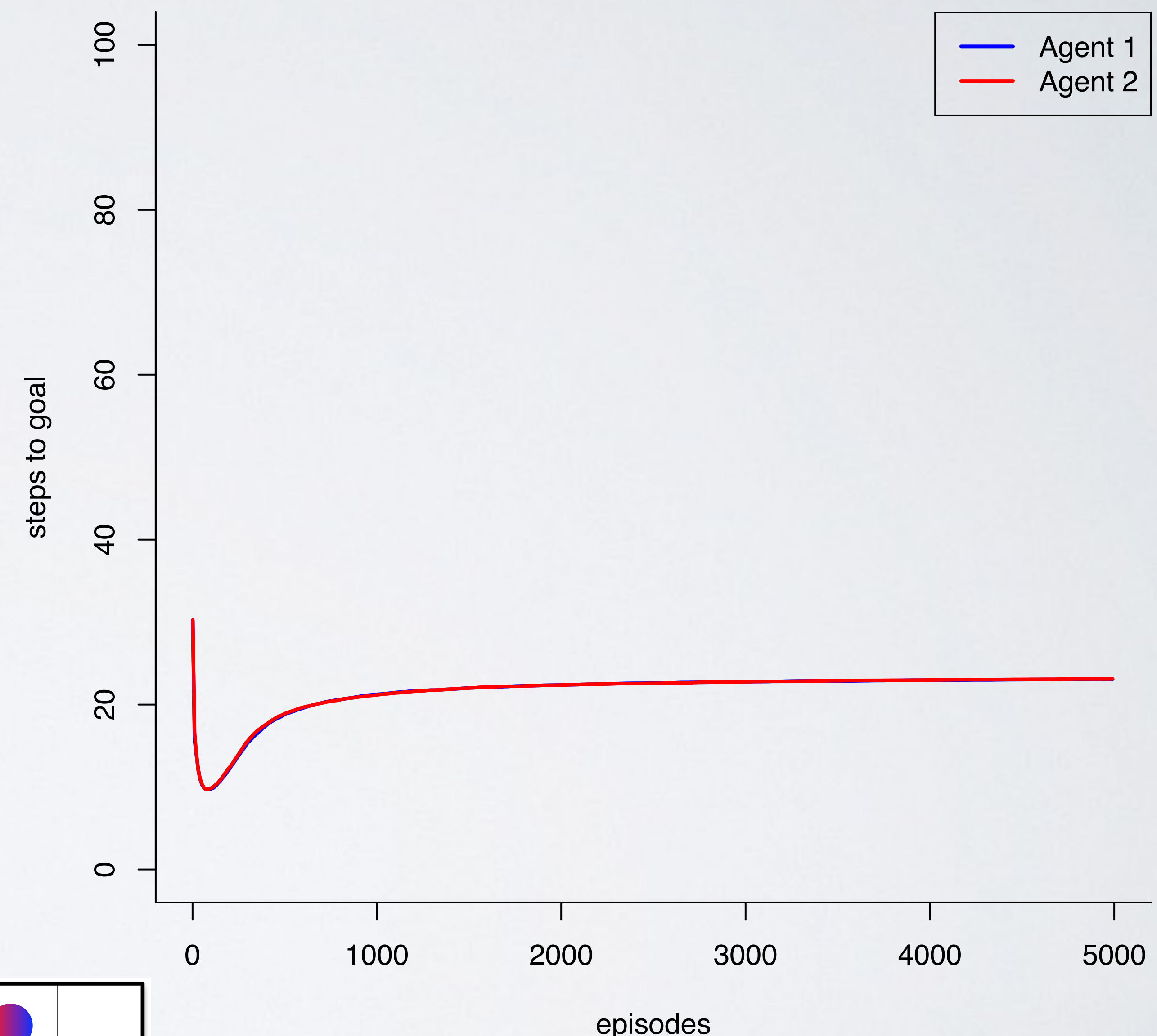


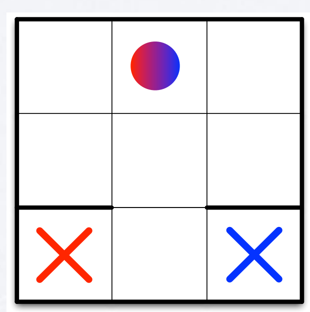




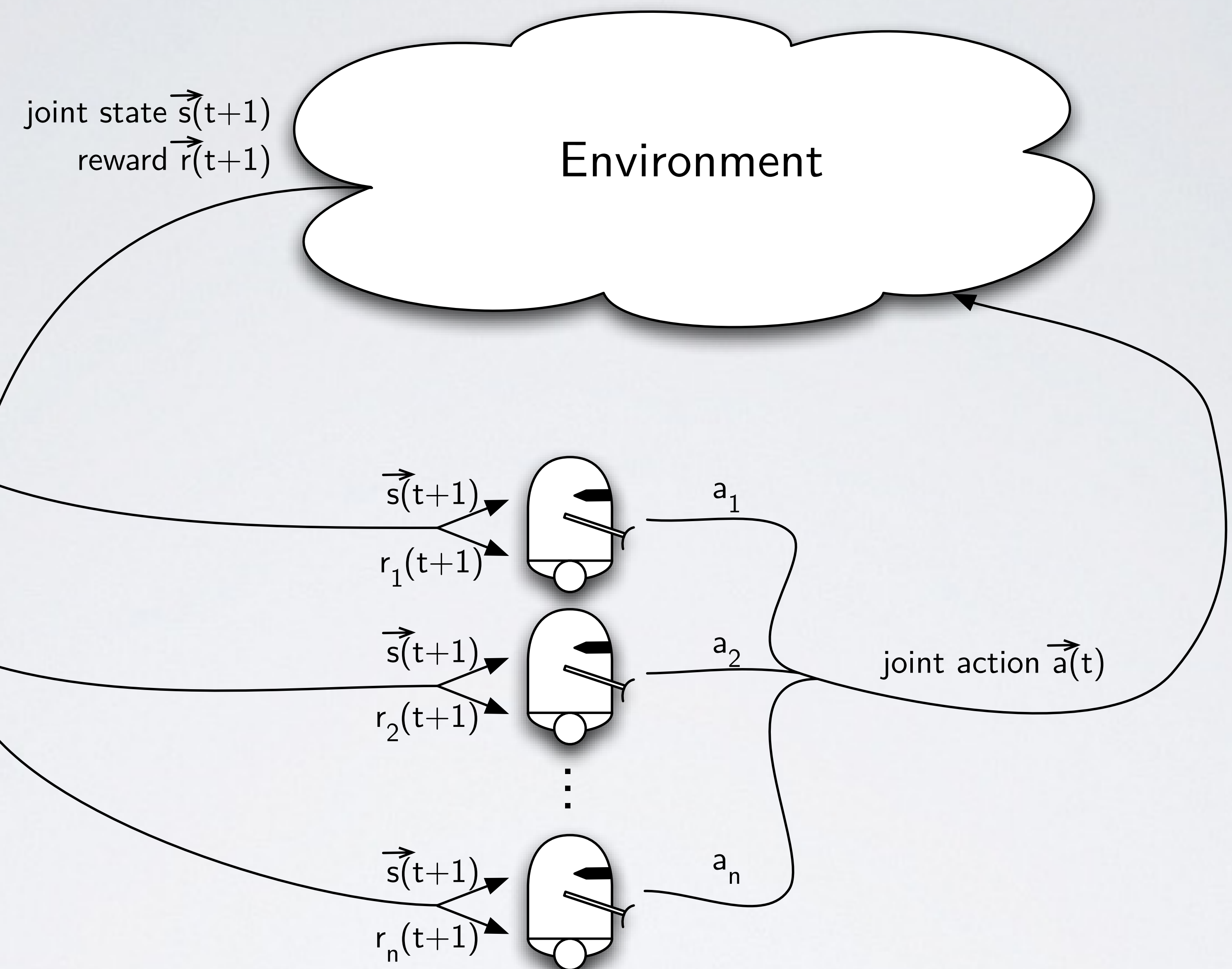
RL WITH **E**-GREEDY (**E** = 0.9)





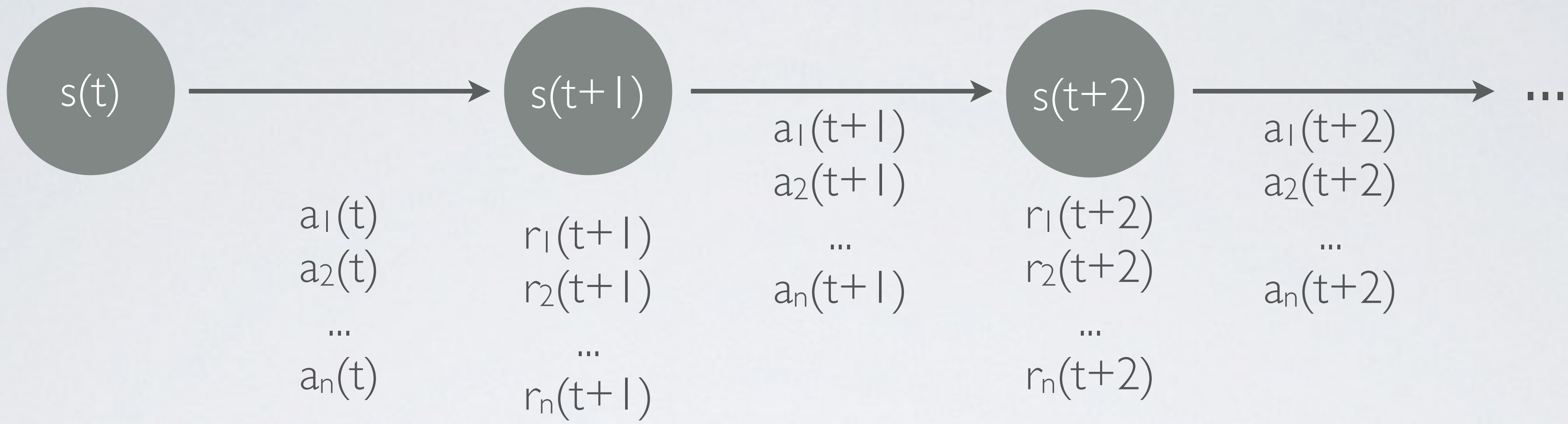


MULTI-AGENT REINFORCEMENT LEARNING



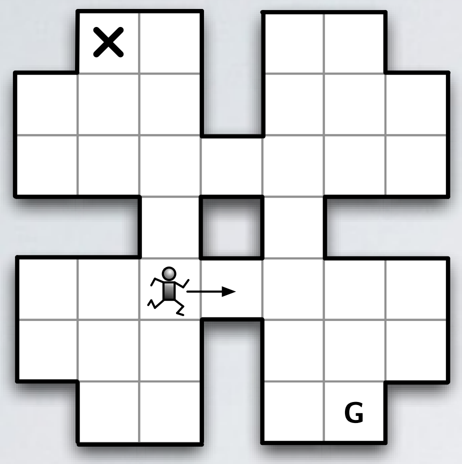
- Agents influence each other
- Possibly conflicting interests
- Observations
- Expensive communication

MARKOV GAMES



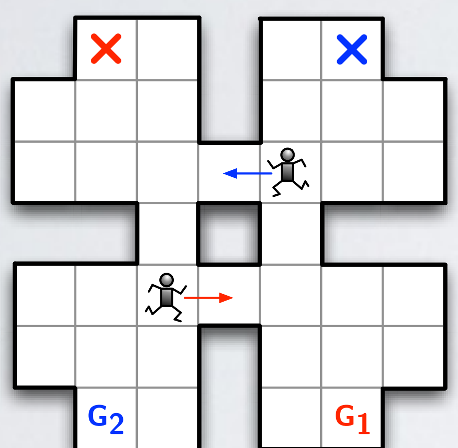
- n the number of agents
- $S = s_1, \ldots, s_N$ a finite set of states
- $A = A_1, \ldots, A_N$ with A_k the action set of agent k
- $T = S \times A_1 \times \ldots \times A_N \times S \rightarrow [0, 1]$ the transition function
- $R_k = S \times A_1 \times \ldots \times A_N \times S \rightarrow \mathbb{R}$ the reward function of agent k

SPARSE INTERACTIONS

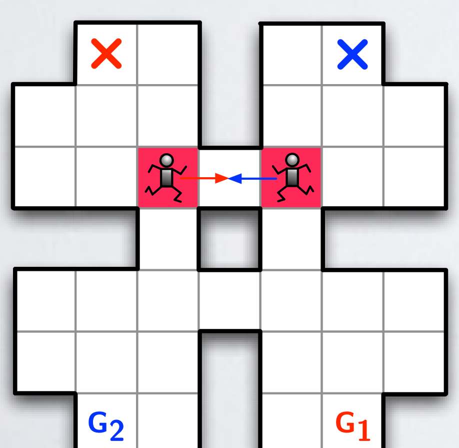


1 agent

Transitions & rewards are only dependent on 1 agent

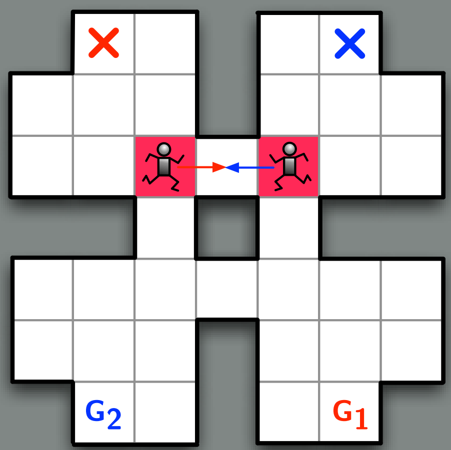


2 agents
Far away and not interacting with each other
Transitions & rewards are independent of state/action of other agents



2 agents
Close to each other and interacting!!!
i.e. transitions & rewards are dependent

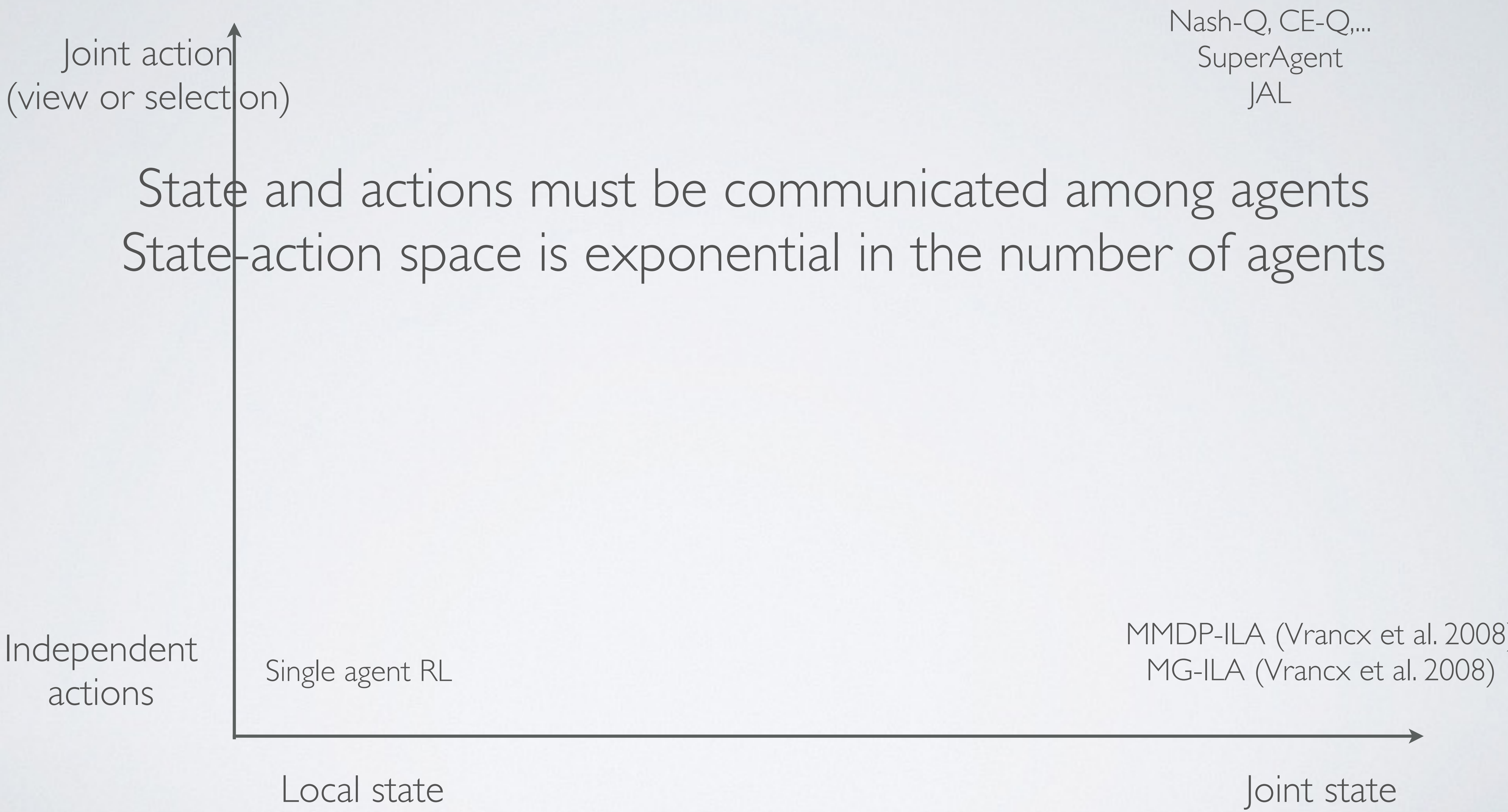
SPARSE INTERACTIONS



2 agents
Close to each other and interacting!!!
i.e. transitions & rewards are dependent

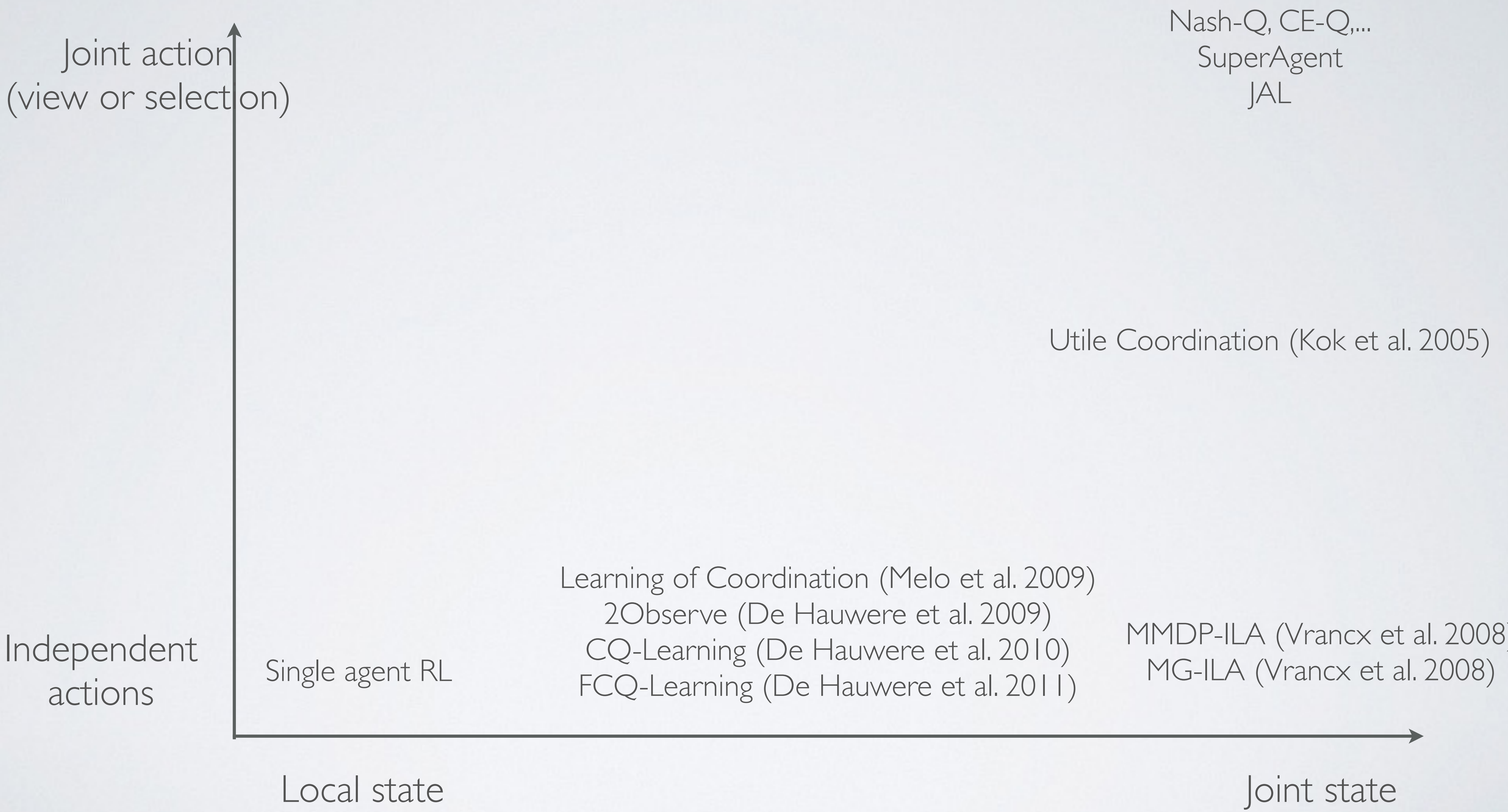
Assumptions:
Agents can do something useful alone
Interactions are sparse
f.i. Air traffic control, automated warehouses, ...

TAXONOMY BASED ON STRATEGIC INTERACTIONS



State and actions must be communicated among agents
State-action space is exponential in the number of agents

TAXONOMY BASED ON STRATEGIC INTERACTIONS

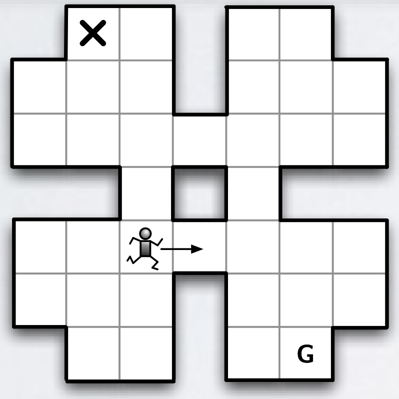


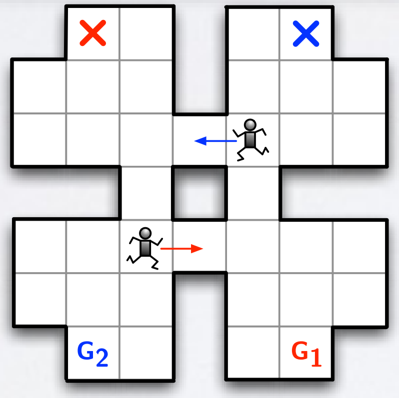
INTUITION OF SPARSE INTERACTIONS

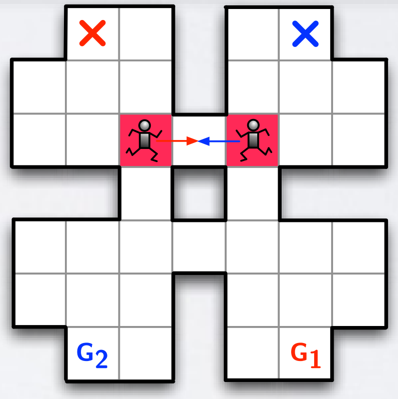
Is there influence from another agent?

No → Act independently, as if single-agent.

Yes → Use a multi-agent technique to coordinate.







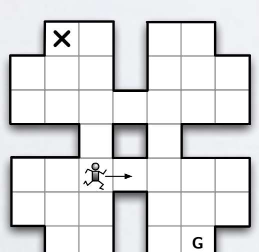
When should agents observe the state information of other agents to avoid coordination problems?

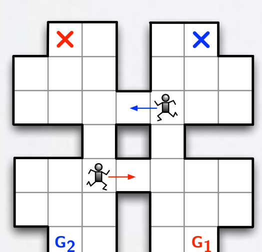
MODELING INTERACTIONS

- Dynamics of the system are a Markov game
- Model sparse interactions as a DEC-SIMDP (Melo et al., 2010)

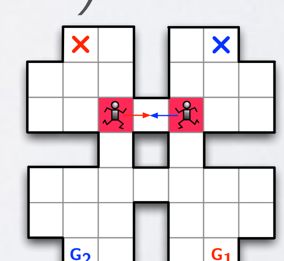
$$\Gamma = (\underbrace{M^k}, (\underbrace{M^{I,l}}, S^{I,l}))$$

MDP for each agent k in the absence of other agents (containing local states)





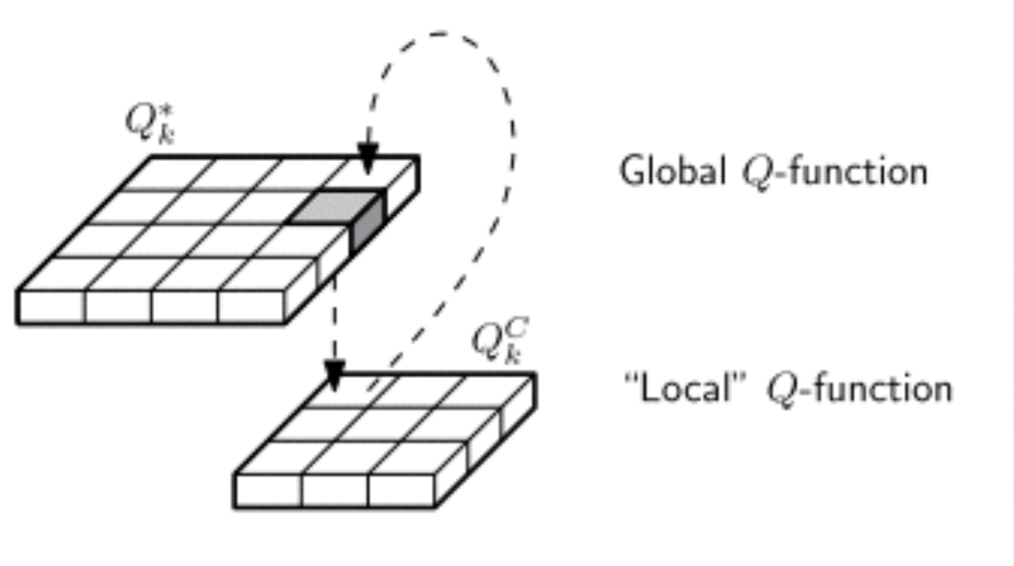
Team Markov game for the local interaction between K agents in L interaction states (containing system states)



OUTLINE

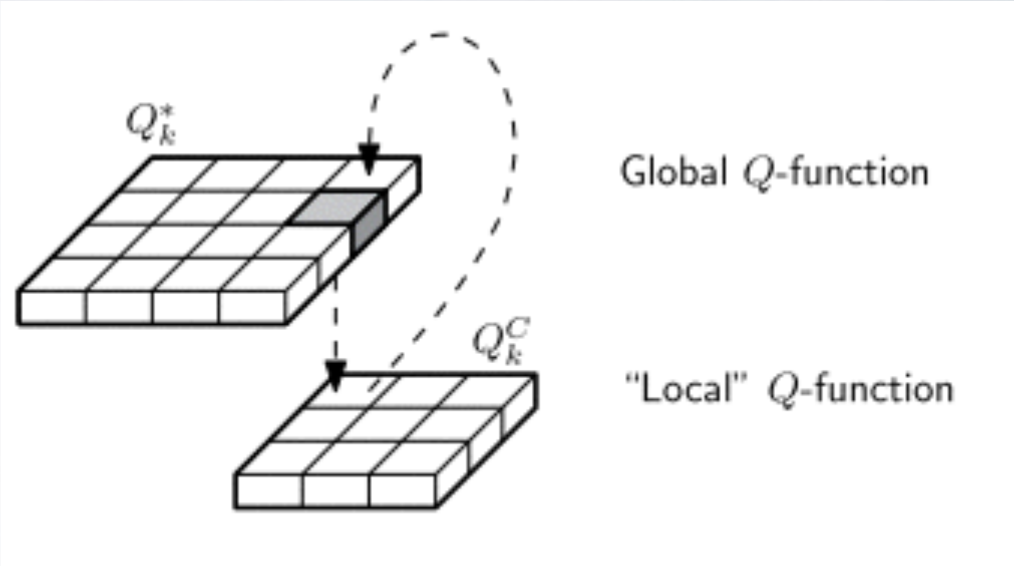
Learning of Coordination
2Observe
CQ-Learning
FCQ-Learning
Transfer learning

Learning of Coordination



LEARNING OF COORDINATION

- Add Pseudo COORDINATE action
- External Active Perception
- Cost for coordination

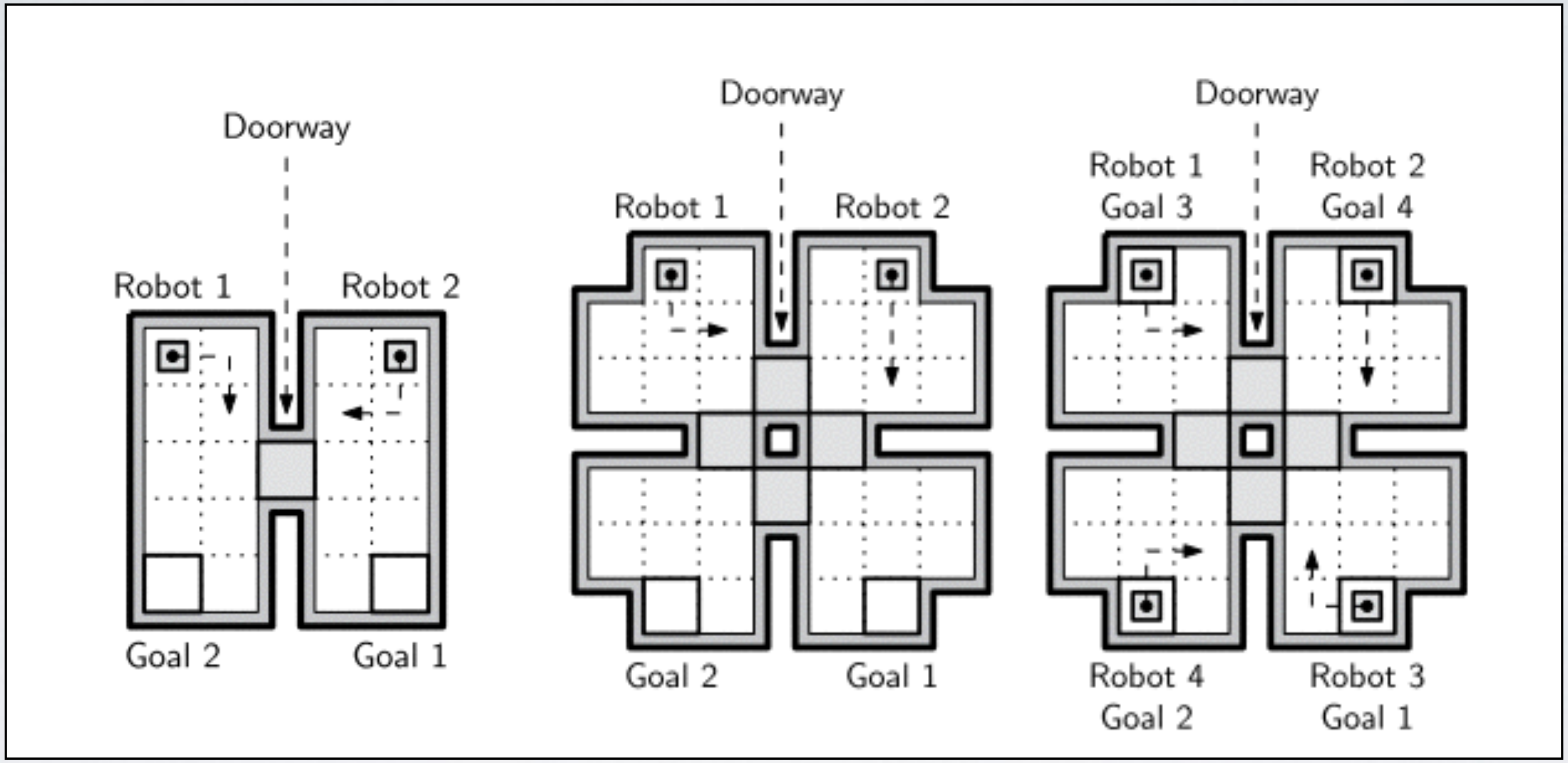


THE ALGORITHM

Algorithm 1 Learning algorithm for agent k

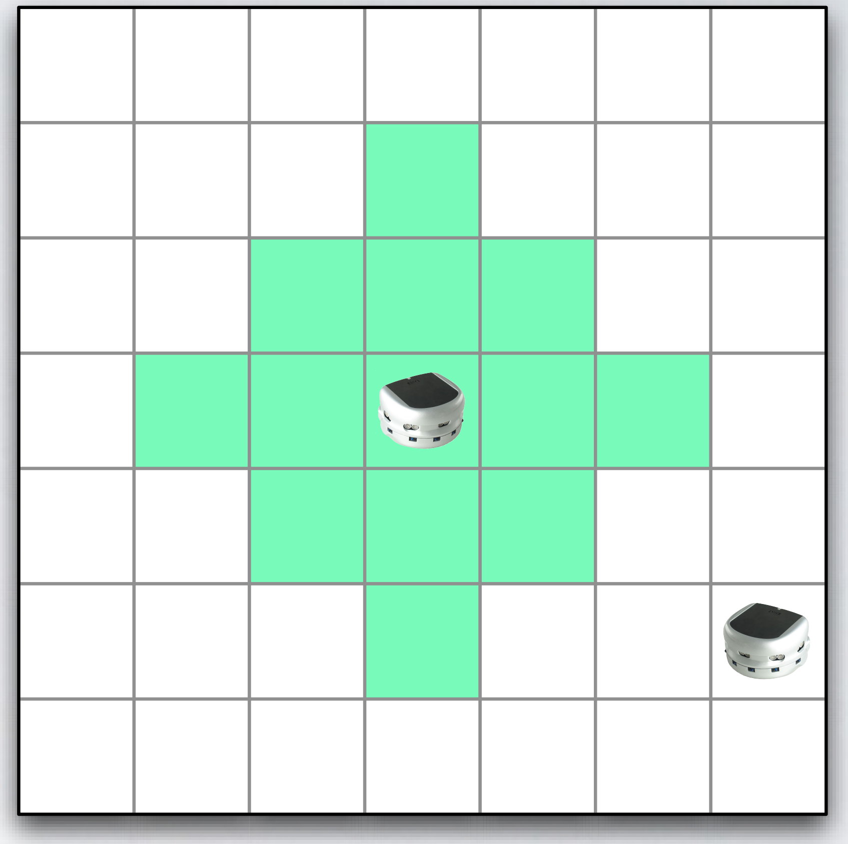
Initialize Q_k^* and Q_k^C;
Set $t = 0$;
while (FOREVER) **do**
    Choose $A_k(t)$ using π_e;
    if $A_k(t) =$ COORDINATE **then**
        if $ActivePercept =$ TRUE **then**
            $\hat{A}_k(t) = \pi_g(Q_k^C, X(t))$;
        else
            $\hat{A}_k(t) = \pi_g(Q_k^*, X_k(t))$;
        end if
        Sample $R_k(t)$ and $X_k(t+1)$;
        if $ActivePercept =$ TRUE **then**
            `QLUpdate`$\big(Q_k^C; X(t), \hat{A}_k(t), R_k(t), X_k(t+1), Q_k^*\big)$;
        end if
    else
        Sample $R_k(t)$ and $X_k(t+1)$;
    end if
    `QLUpdate`$\big(Q_k^*; X_k(t), A_k(t), R_k(t), X_k(t+1), Q_k^*\big)$;
    $t = t + 1$;
end while

RESULTS

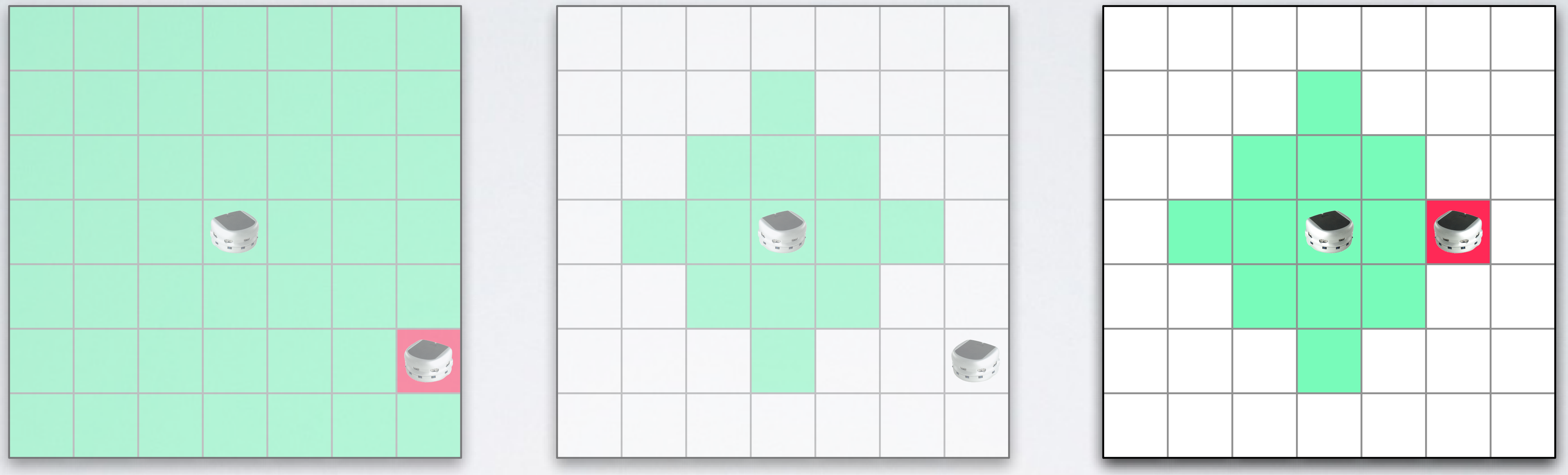


	Learning	Time to goal	Miscoord.
Env. 1 (2R)	Indiv.	10.02 ± 1.57	0.40 ± 0.52
	Non-coop.	10.02 ± 1.58	0.41 ± 0.54
	Coop.	9.94 ± 1.57	0.00 ± 0.00
Env. 2 (2R)	Indiv.	12.45 ± 1.68	0.12 ± 0.33
	Non-coop.	12.45 ± 1.77	0.12 ± 0.33
	Coop.	12.51 ± 1.72	0.00 ± 0.00
Env. 2 (4R)	Indiv.	12.46 ± 1.75	0.47 ± 0.59
	Non-coop.	12.49 ± 1.74	0.49 ± 0.59
	Coop.	12.49 ± 1.77	0.00 ± 0.00

2Observe

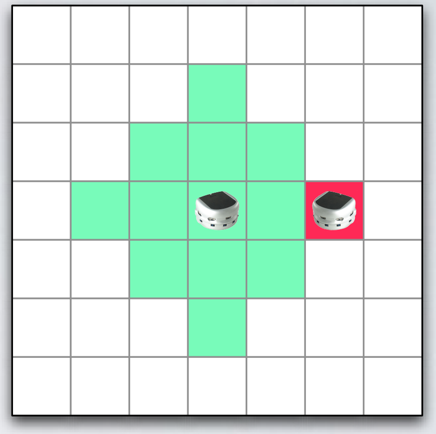


PROBLEM SETTING



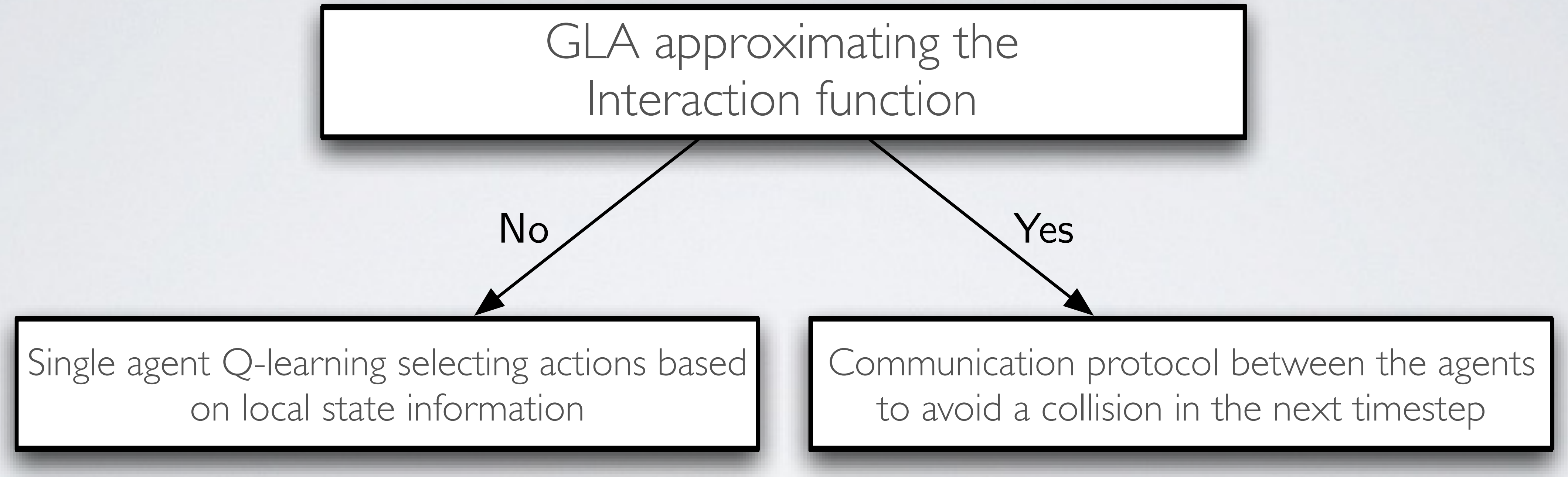
- Learn when to act upon sensory input
- Adaptive obstacle avoidance
- Save energy

INTERACTIONS AS A FUNCTION

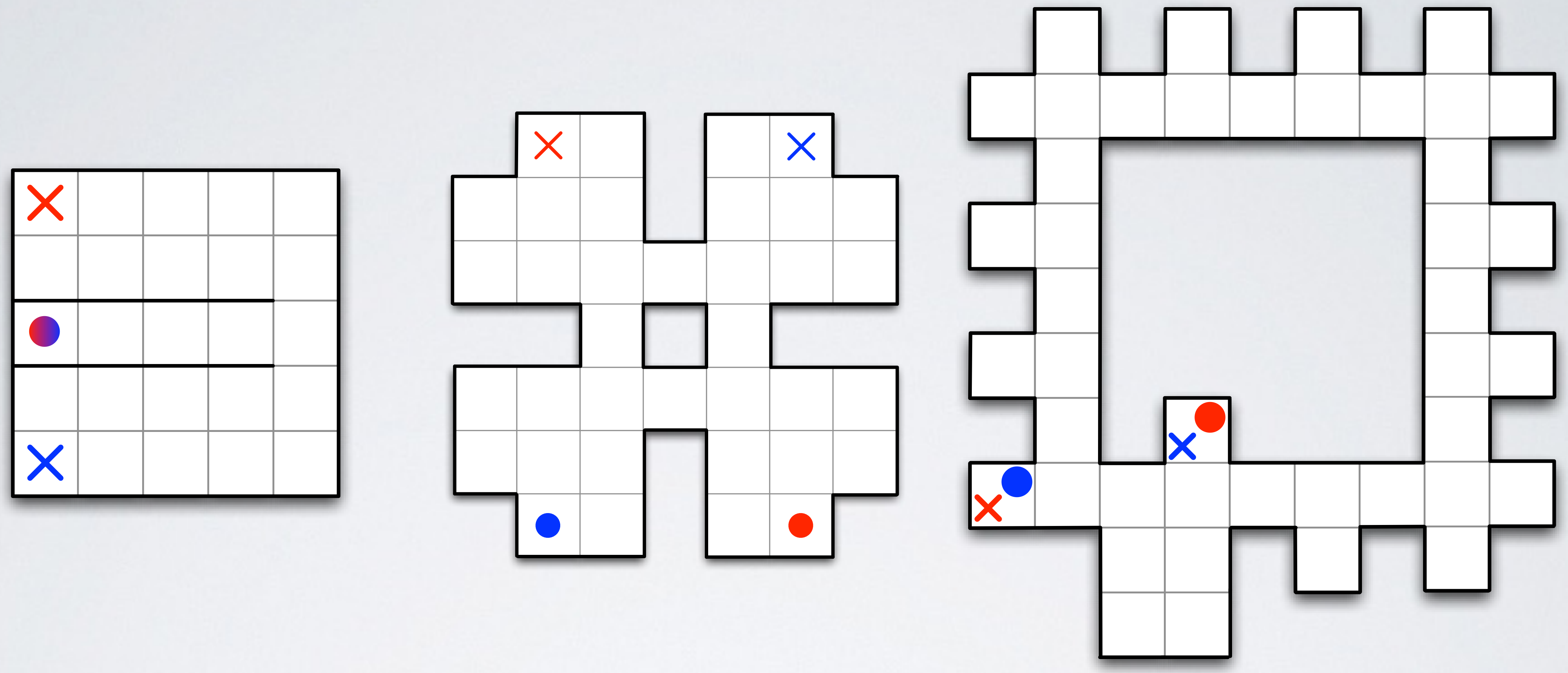


- State space contains sensor data
- Sensor information is only partly relevant
- Interaction area is relative to the agent
- Special kind of sparse interactions, modeled as a DEC-LIMDP (Section 4.2)
- $I_k: S_k \rightarrow S_1 \times \ldots \times S_M$
- Approximating this function using a generalized learning automaton: 2Observe

SOLUTION METHOD: 2OBSERVE

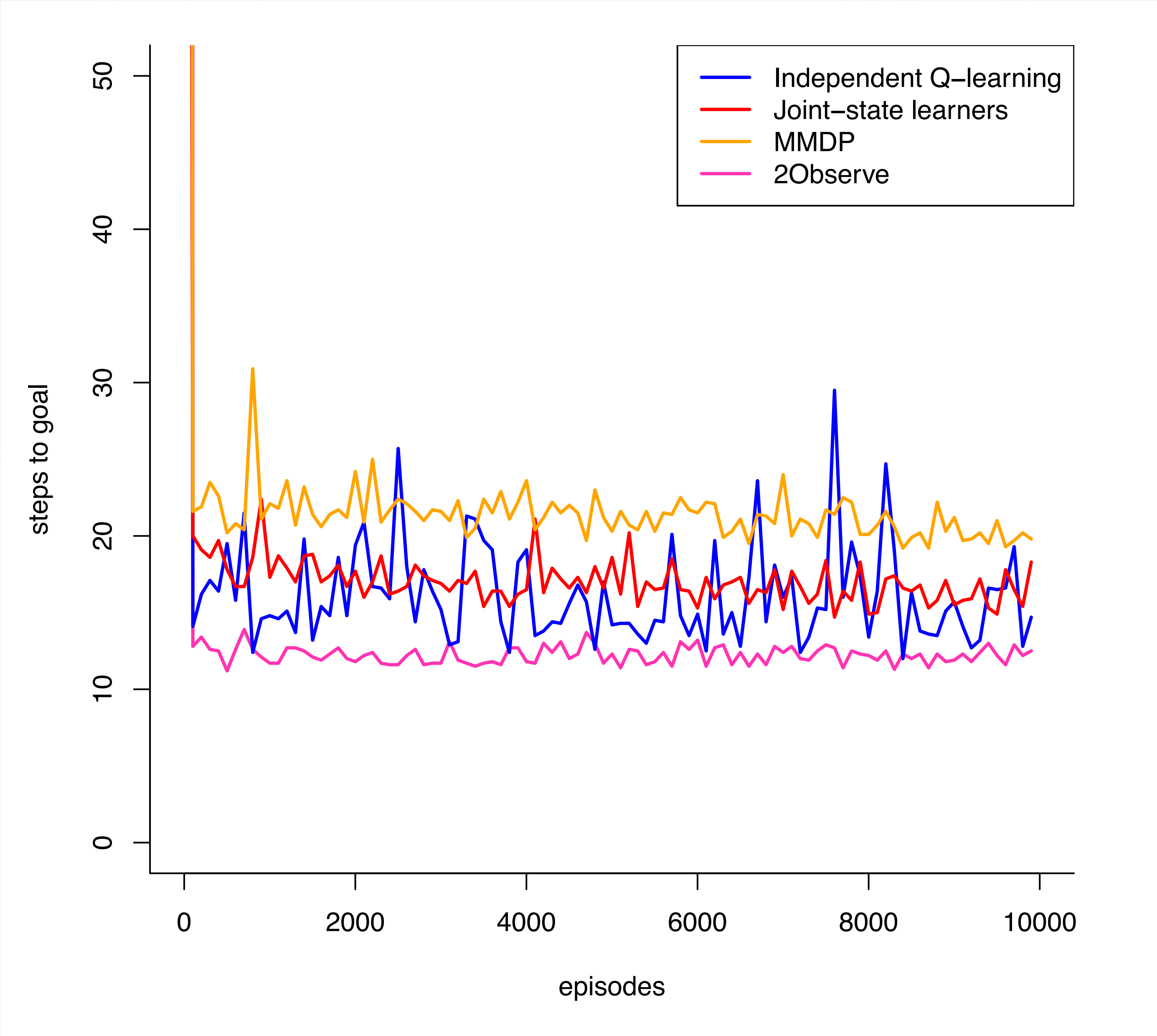


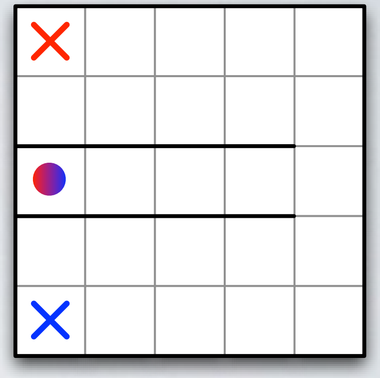
EXPERIMENTAL SETTING

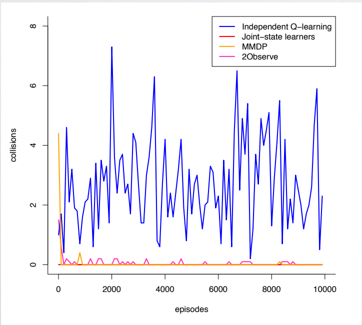


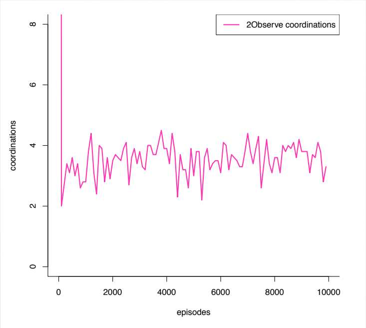
- Reach goal
- Avoid collisions

EXPERIMENTAL RESULTS (TUNNELTOGOAL)

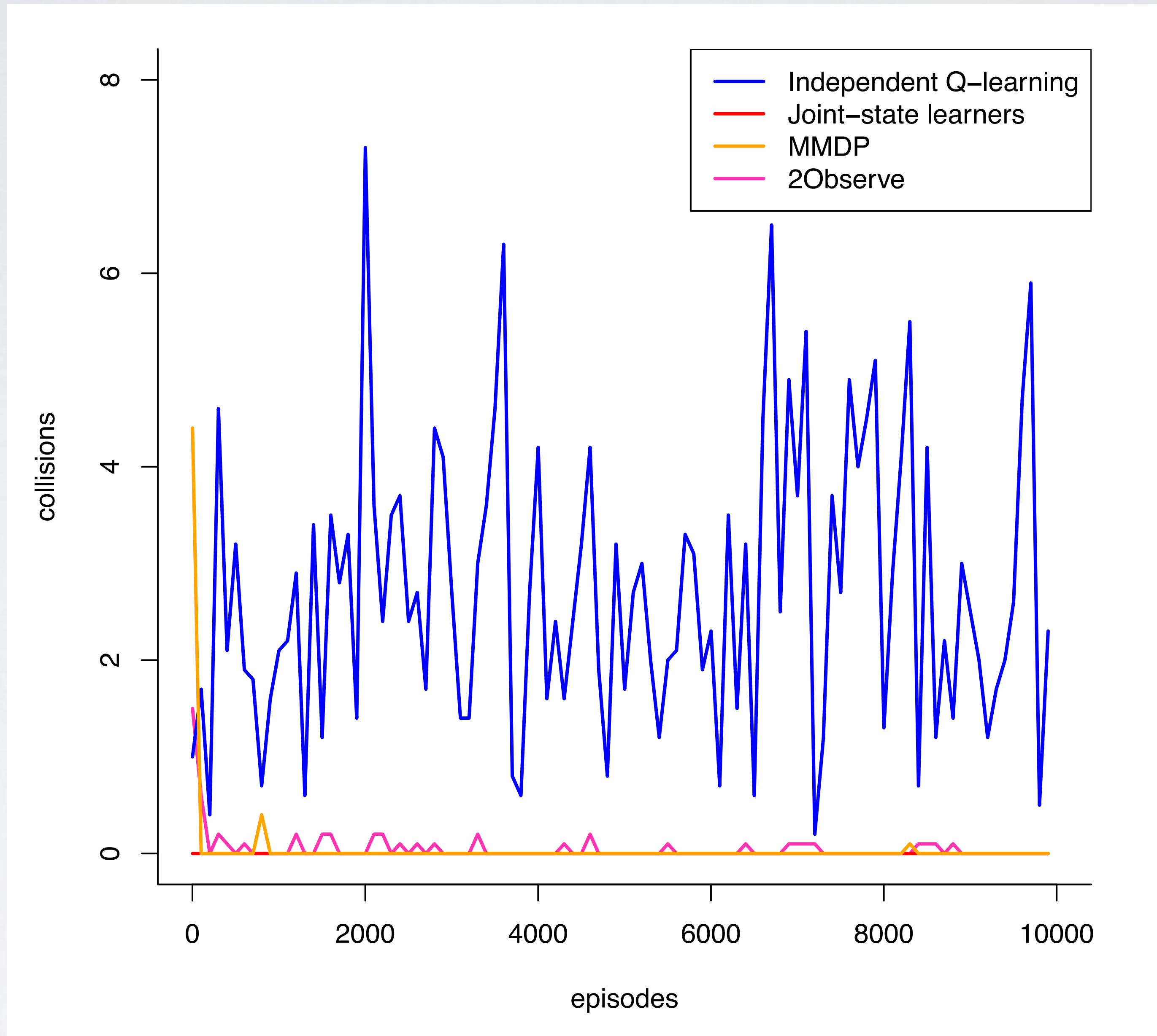


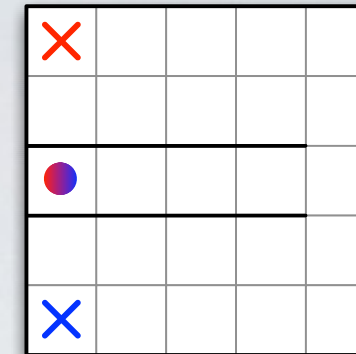


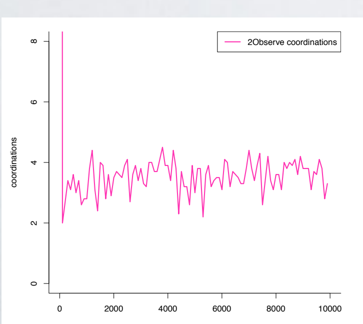


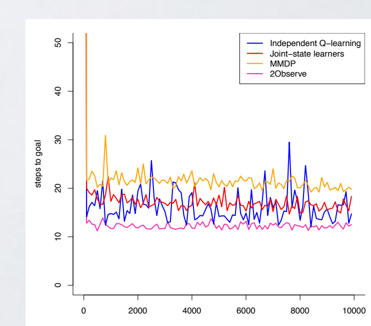


EXPERIMENTAL RESULTS (TUNNELTOGOAL)

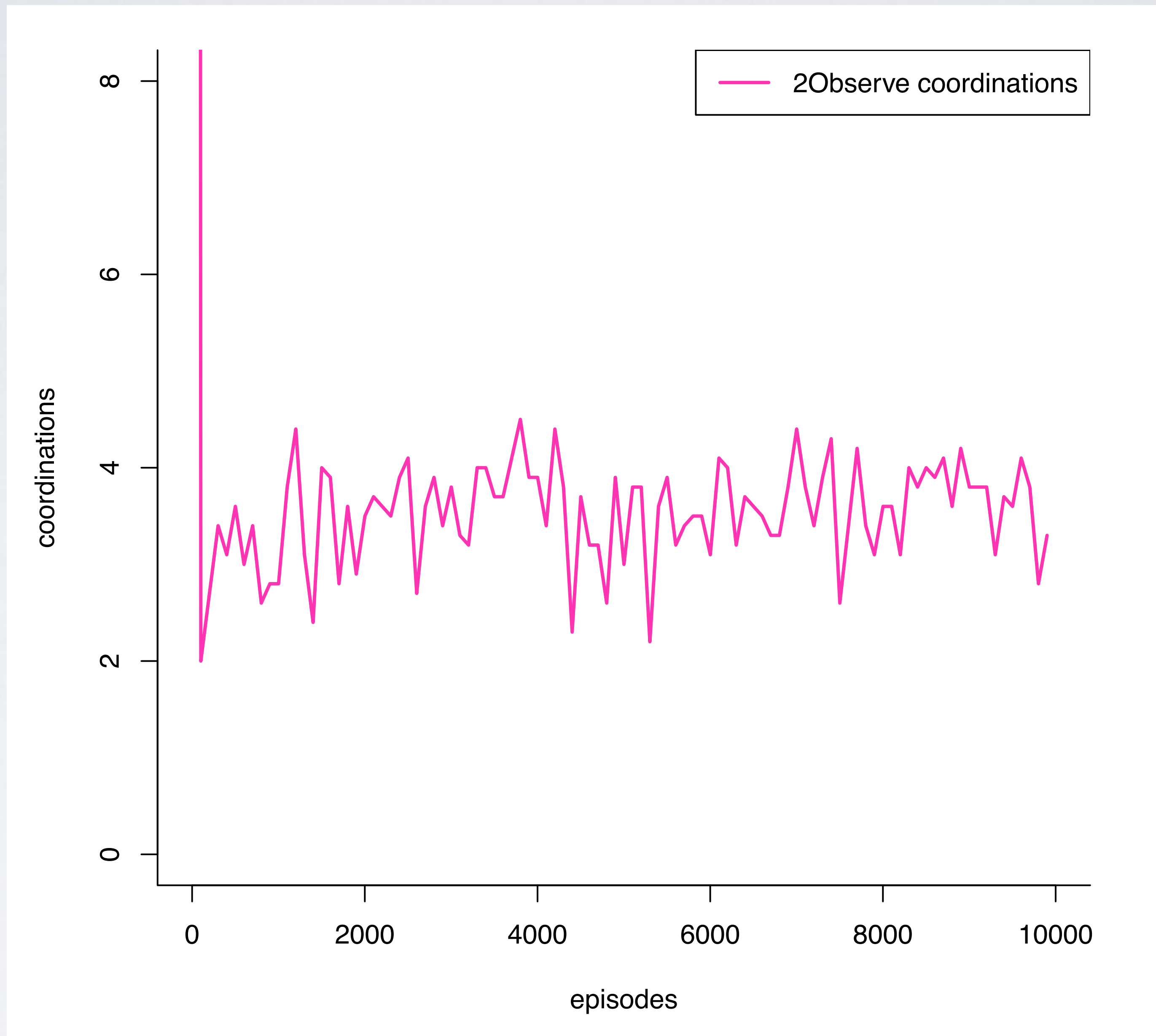


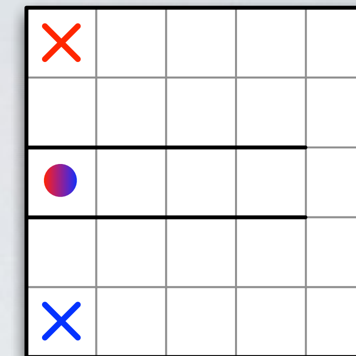


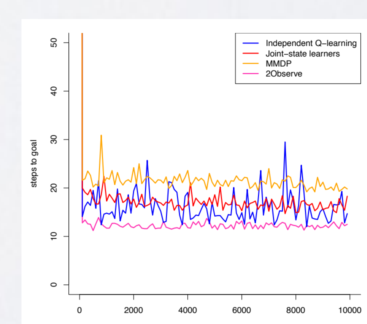


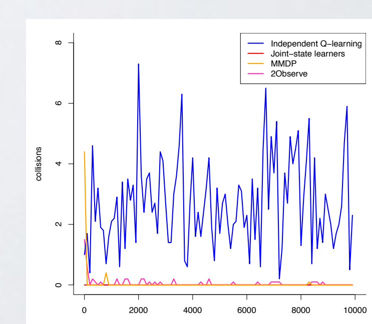


EXPERIMENTAL RESULTS (TUNNELTOGOAL)

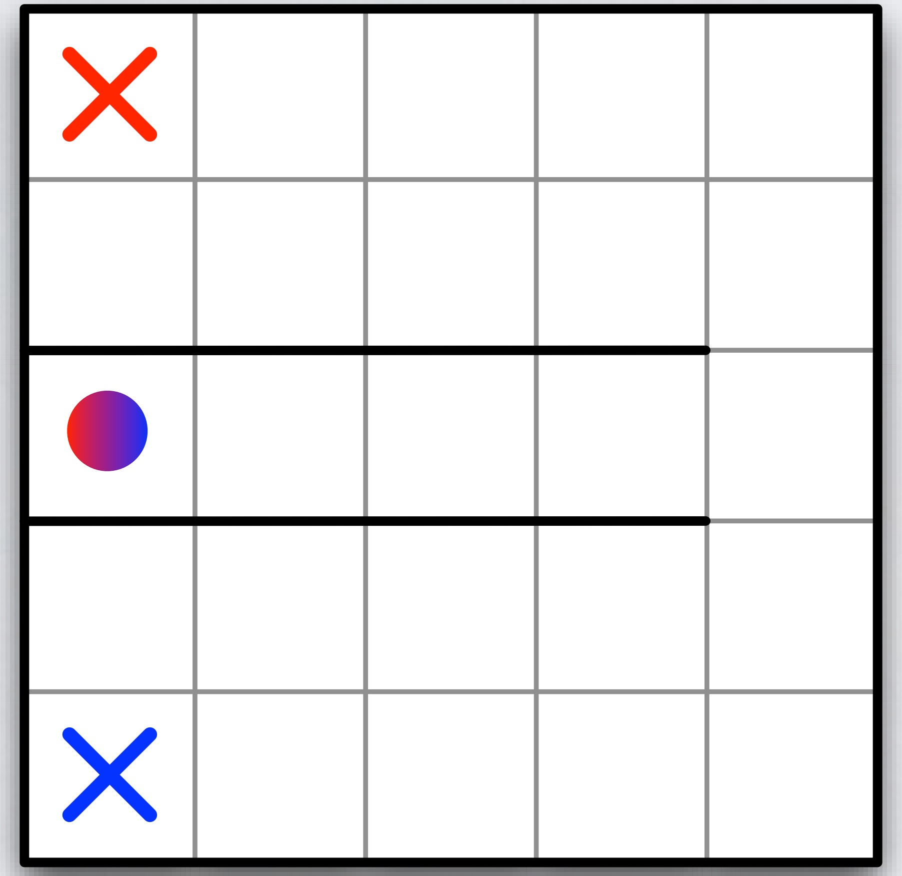






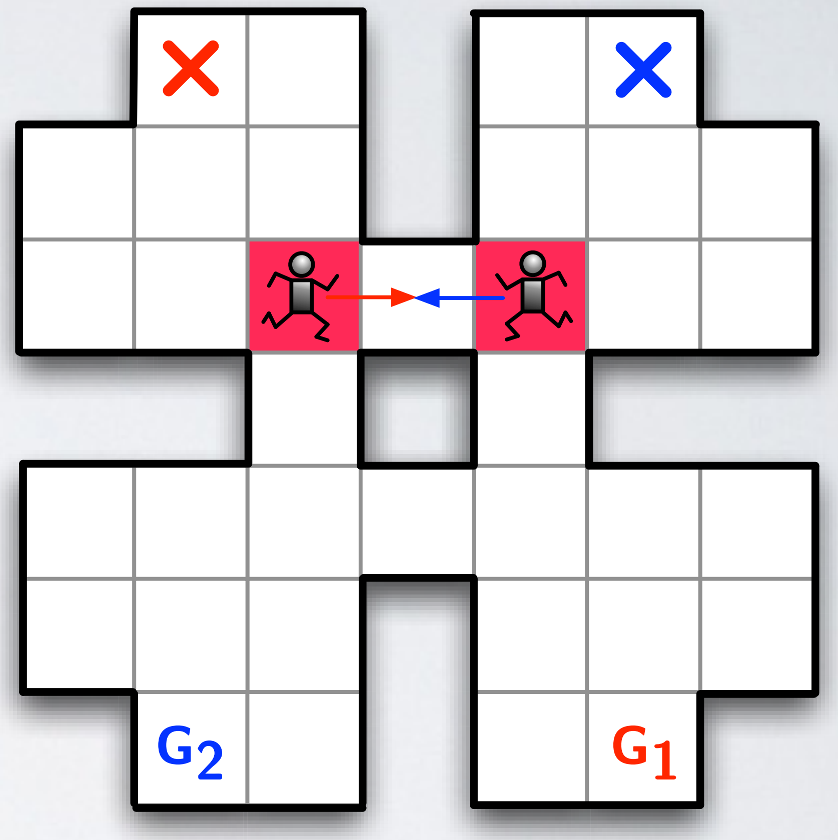


EXPERIMENTAL RESULTS (2) (TUNNELTOGOAL)

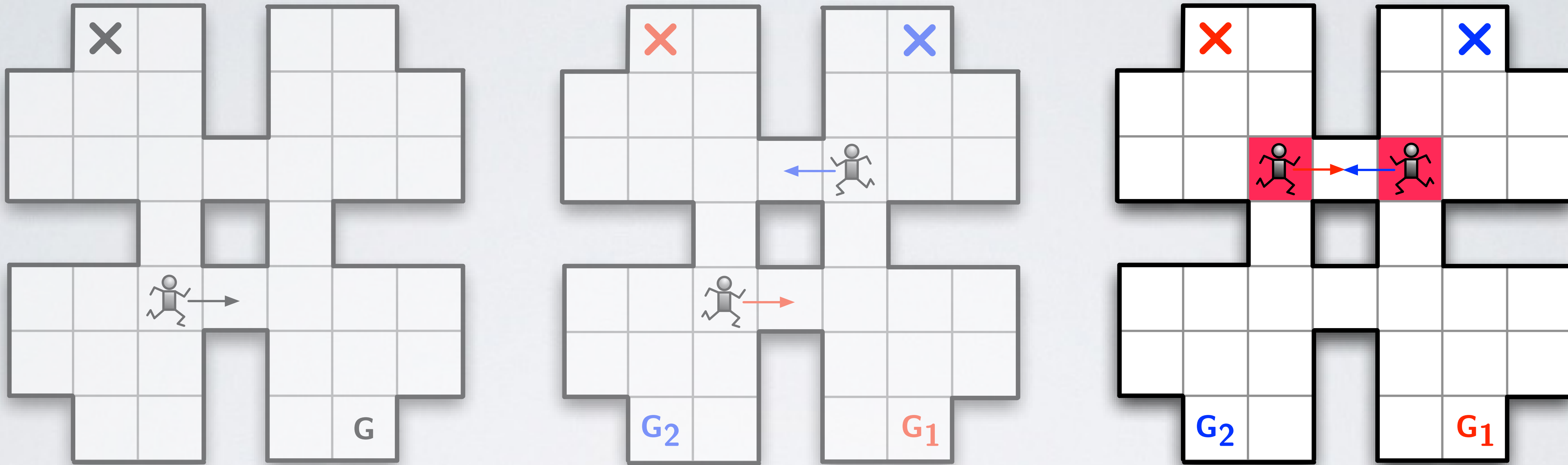


- Interactions are relative to the agent
- GLA can approximate this interaction area

CQ-Learning

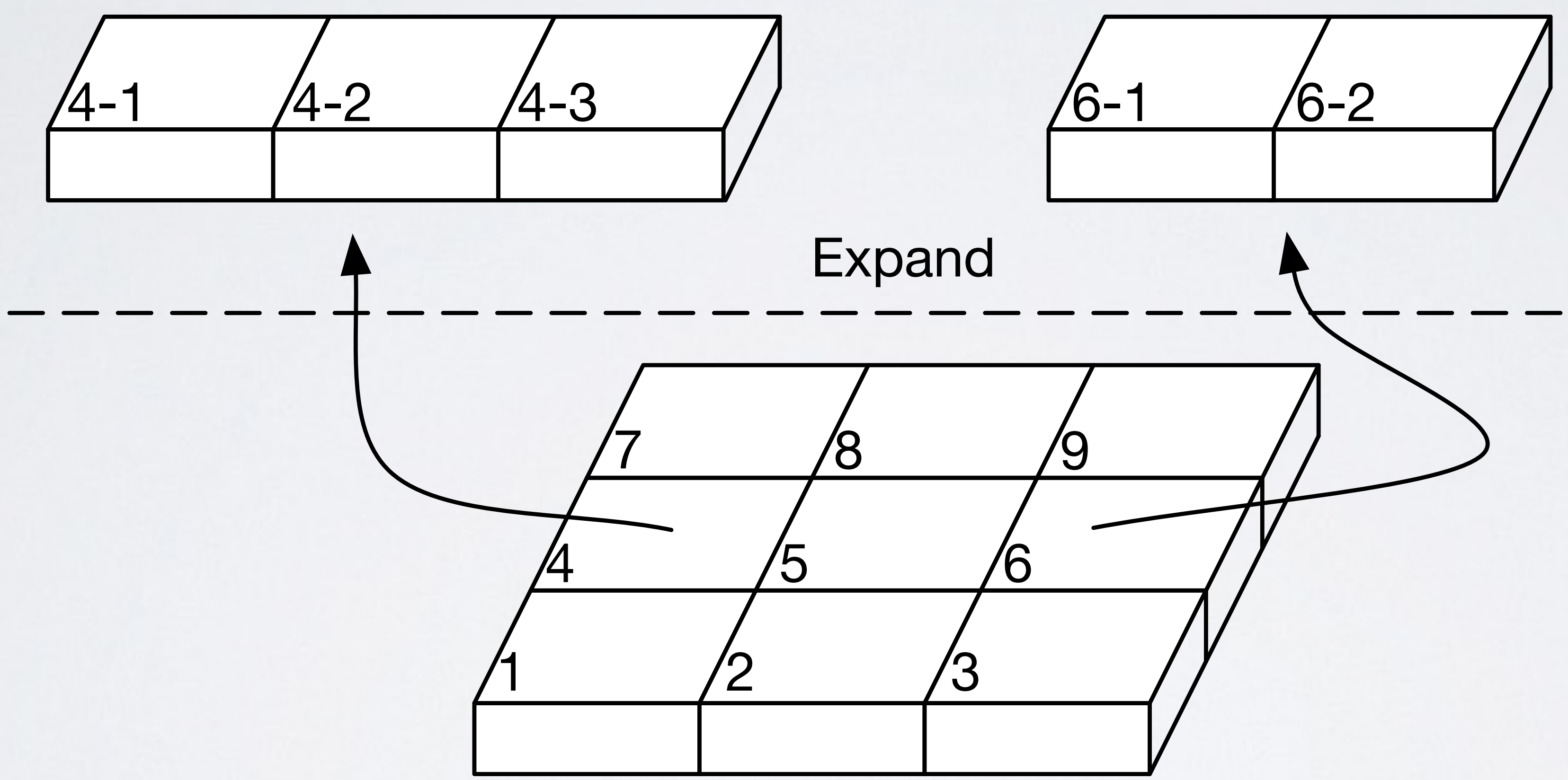


PROBLEM SETTING

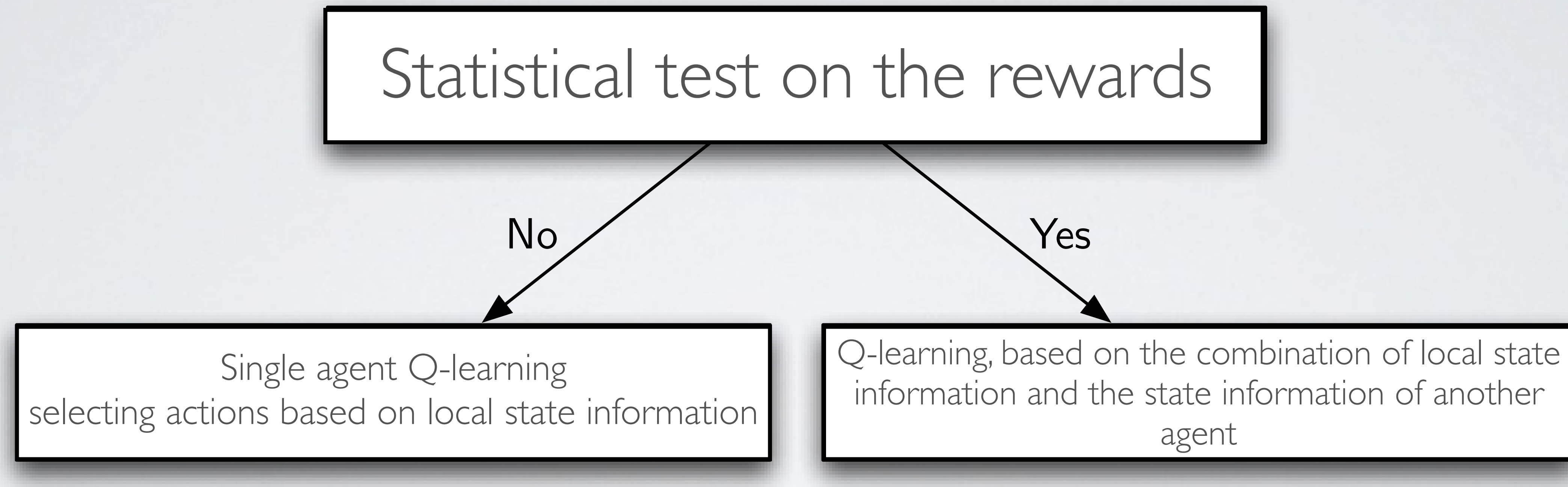


- Agents only interact where their policies interfere
- Locally adapt policy

REPRESENTATION IDEA



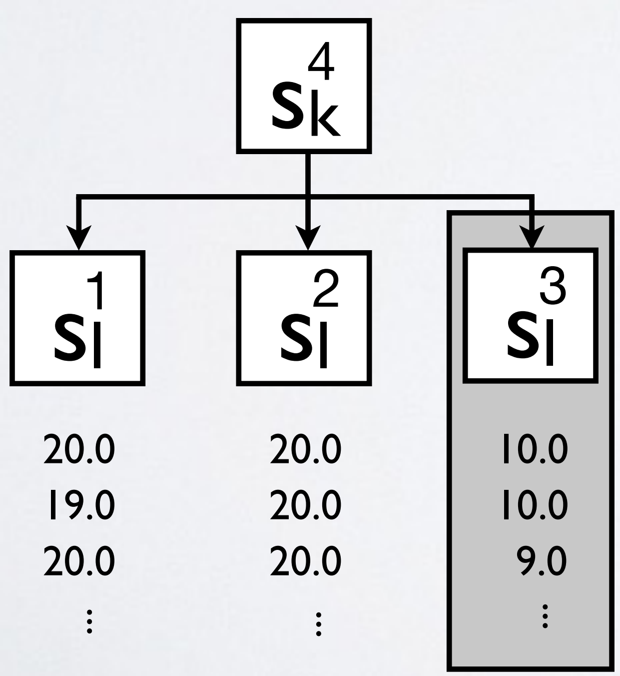
SOLUTION METHOD: CQ-LEARNING



CQ-LEARNING : STATISTICAL TESTS

	S_k^1	S_k^2	S_k^3	S_k^4
	11.0	20.0	15.0	10.0
	10.0	20.0	15.0	19.0
	9.0	20.0	14.8	9.0
	10.0	19.0	15.0	20.0
	11.0	20.0	14.9	20.0
	⋮	⋮	⋮	⋮
Expected reward:	**10.0**	**20.0**	**15.0**	**20.0**

Expand

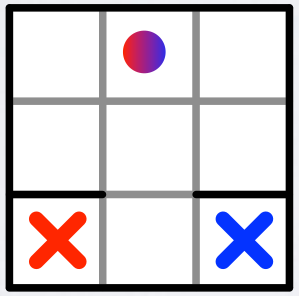


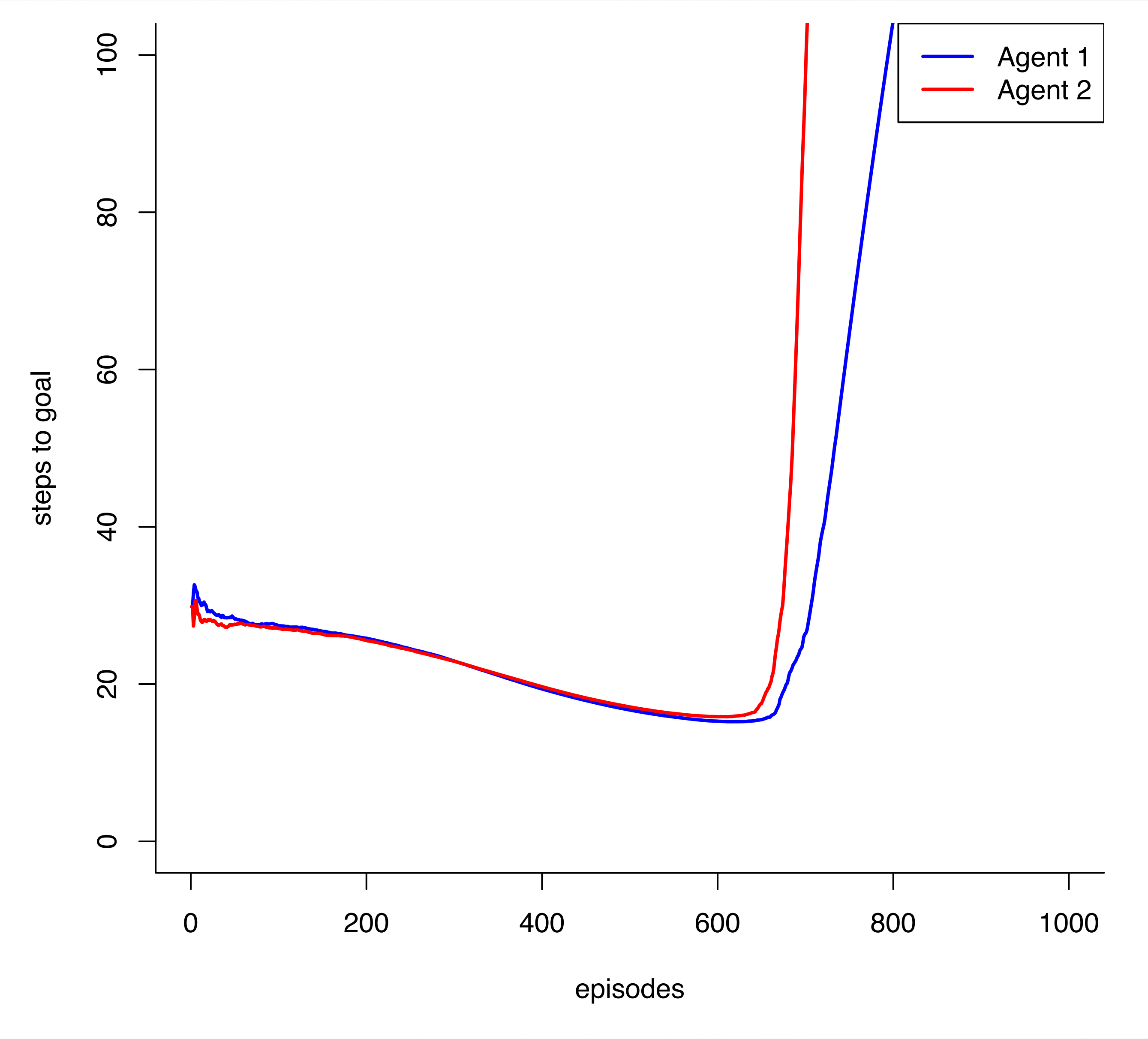
- Agents have been learning alone in the environment
- Agent k acts independently using only local state information (s_k) in a multi-agent environment
- Perform statistical test against baseline
- Samples its rewards, based on the state information of other agents & performs the same test

$$S_k^4 \Rightarrow \langle S_k^4, S_l^3 \rangle$$

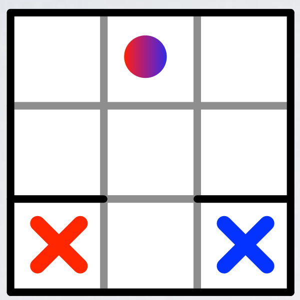
30

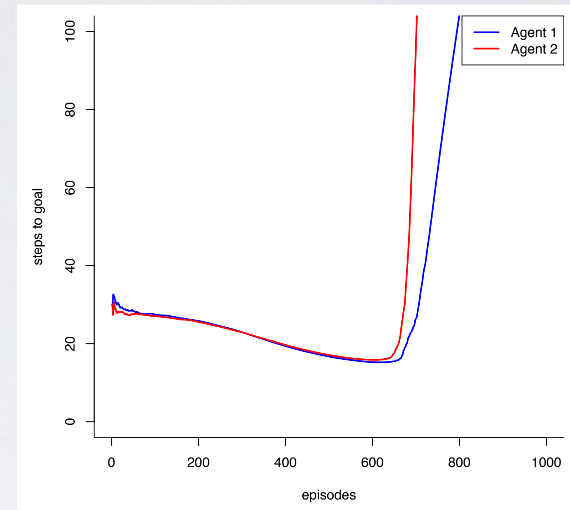
CQ-LEARNING BASELINE FOR STATISTICAL TESTS





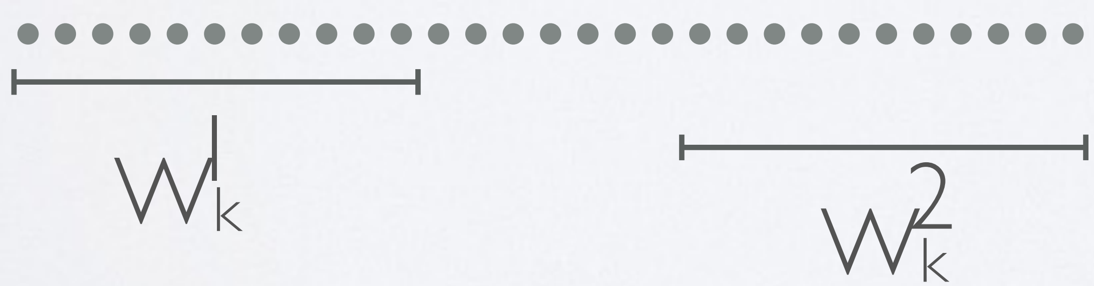
CQ-LEARNING   BASELINE FOR STATISTICAL TESTS





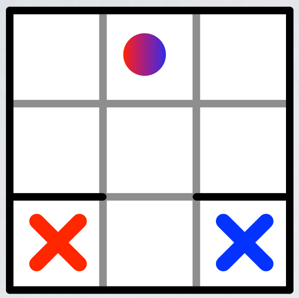
Initial rewards (sliding window)

for a particular state action pair:

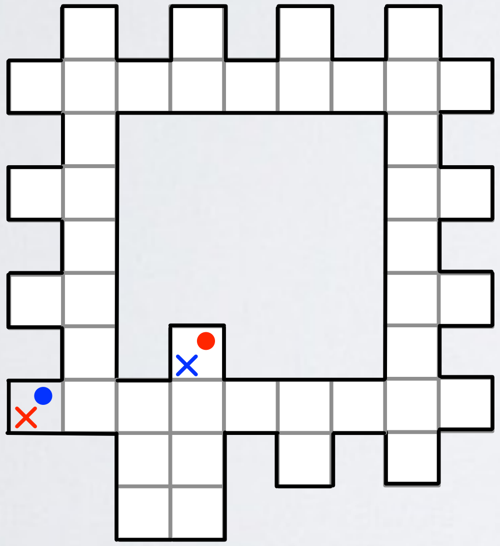


Compare W_k^1 against W_k^2

EXPERIMENTAL RESULTS (1)



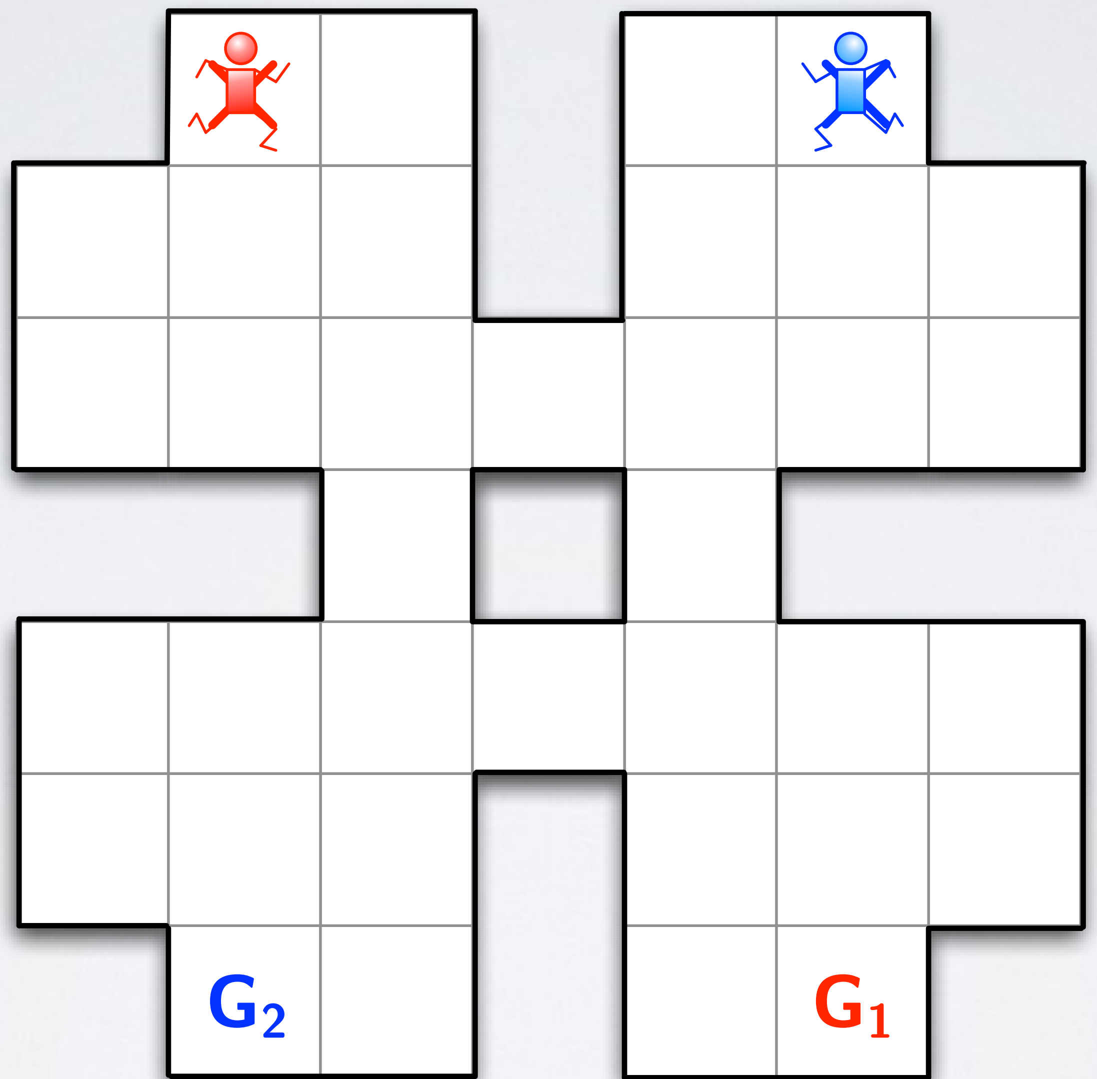
Env	Alg	#states	#actions	#coll	#steps
Grid_game_2	Indep	9	4	2.7	22.2 ± 17.9
	JS	81	4	0.1	4.0 ± 0.2
(min steps: 3)	JSA	81	16	0.0	4.7 ± 0.1
	LOC	9.9 ± 0.5	5	0.1	4.0 ± 0.4
	CQ	10 ± 0.0	4	0.0	$\mathbf{3.6 \pm 0.3}$
	CQ_NI	10.9 ± 2.0	4	0.1	4.0 ± 0.3



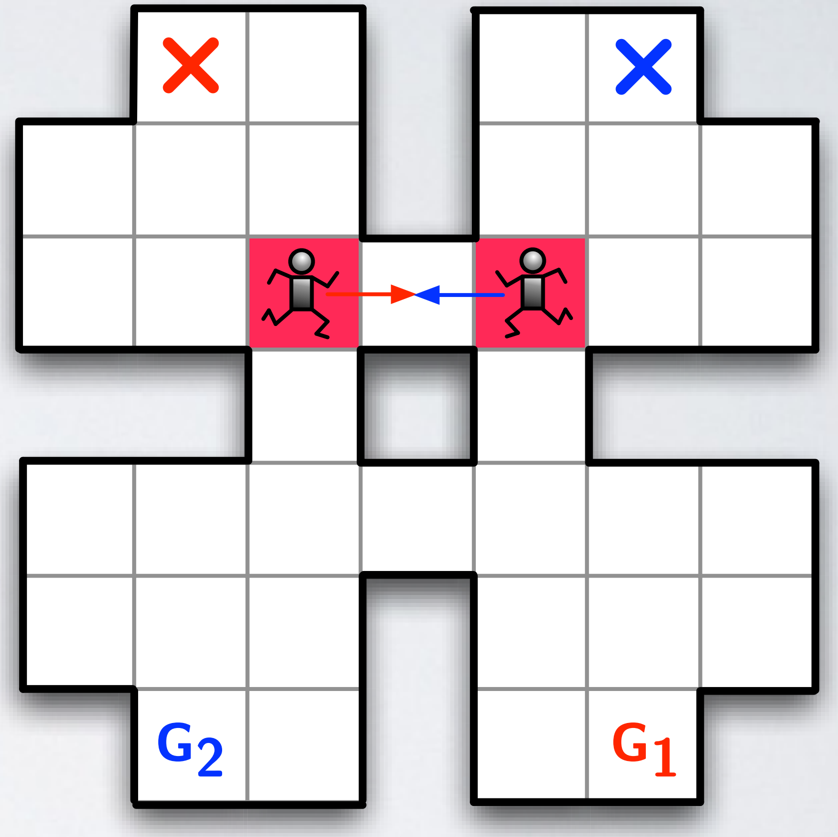
Env	Alg	#states	#actions	#coll	#steps
ISR	Indep	43	4	0.4	9.3 ± 44.8
	JS	1849	4	0.1	5.7 ± 1.6
(min steps: 4)	JSA	1849	16	0.0	7.6 ± 1.4
	LOC	51.3 ± 82.3	5	0.2	6.7 ± 7.5
	CQ	49.0 ± 2.3	4	0.1	$\mathbf{5.1 \pm 0.7}$
	CQ_NI	49.9 ± 7.8	4	0.1	6.0 ± 1.9

EXPERIMENTAL RESULTS (2)

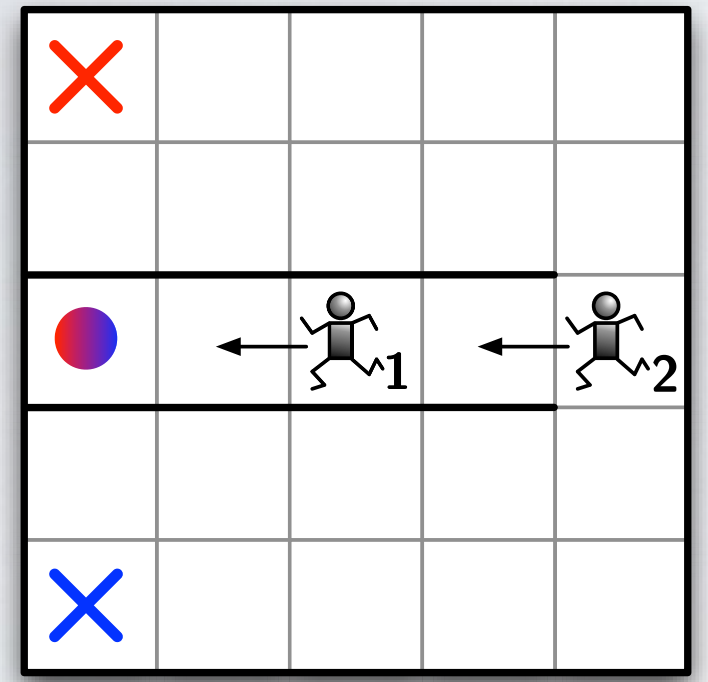
- Sample run



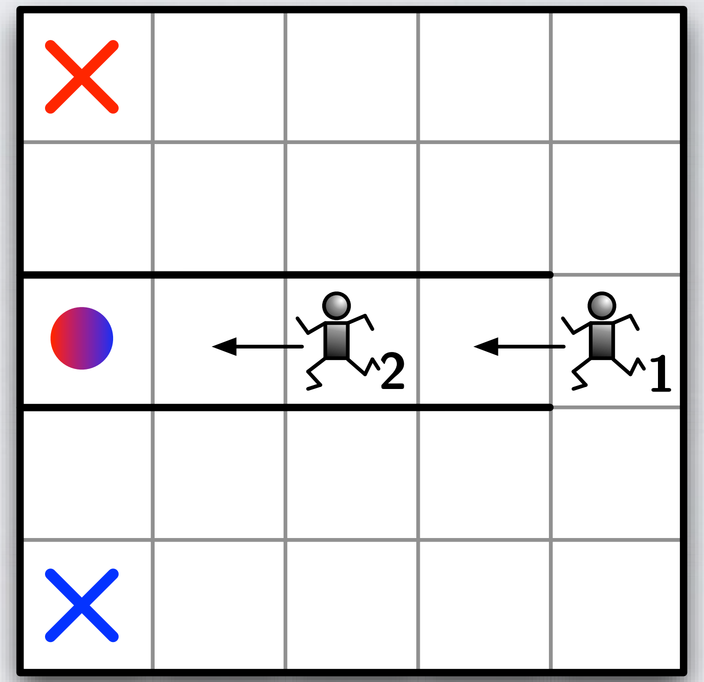
FCQ-Learning



PROBLEM SETTING



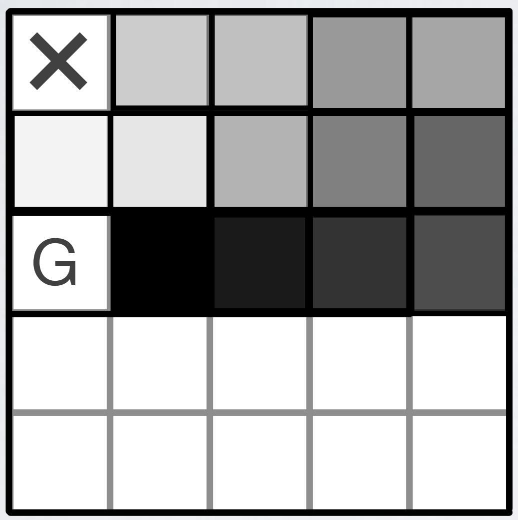
Reward: +20

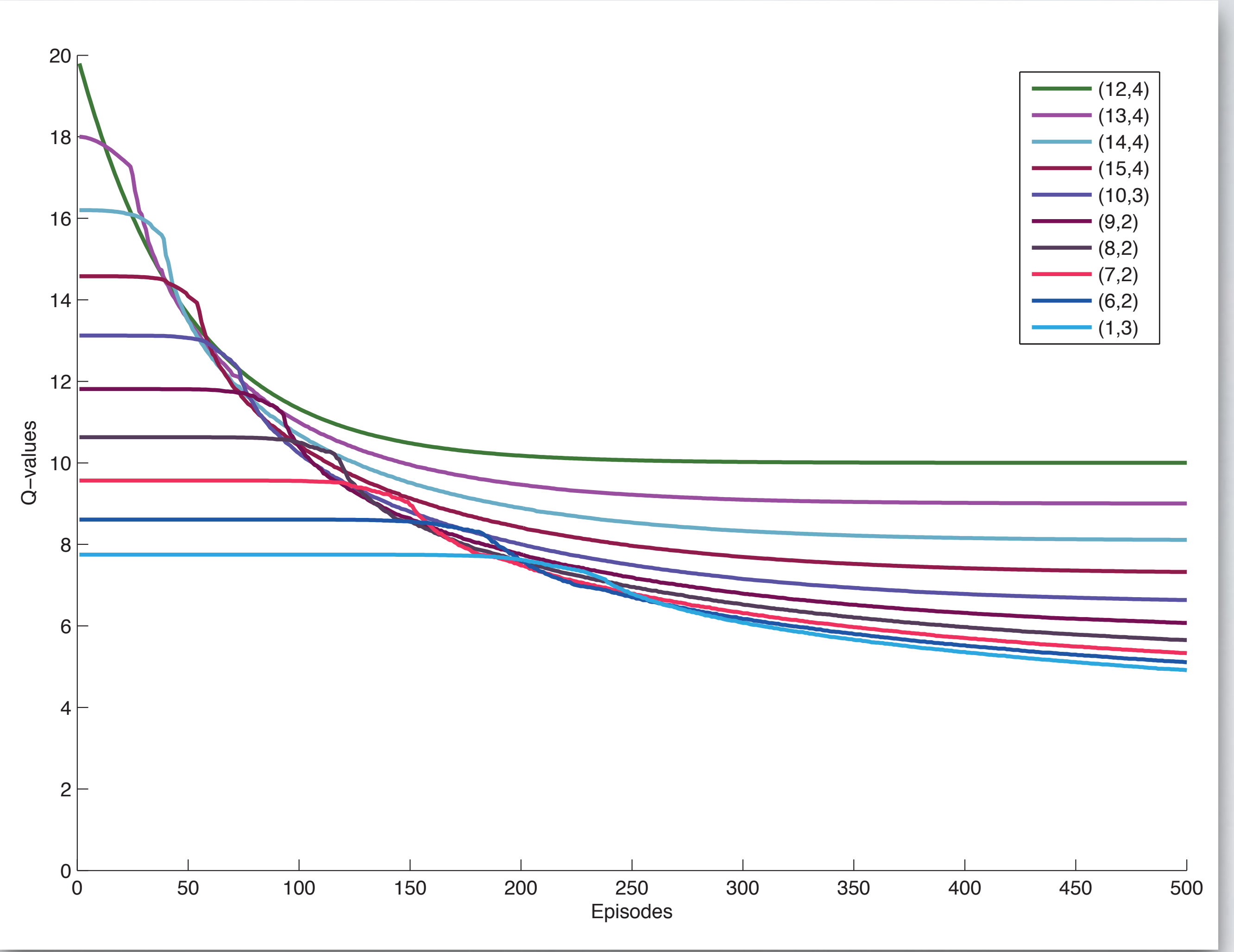


Reward: +10

- Reflected in immediate reward signal
- Too late to solve the problem

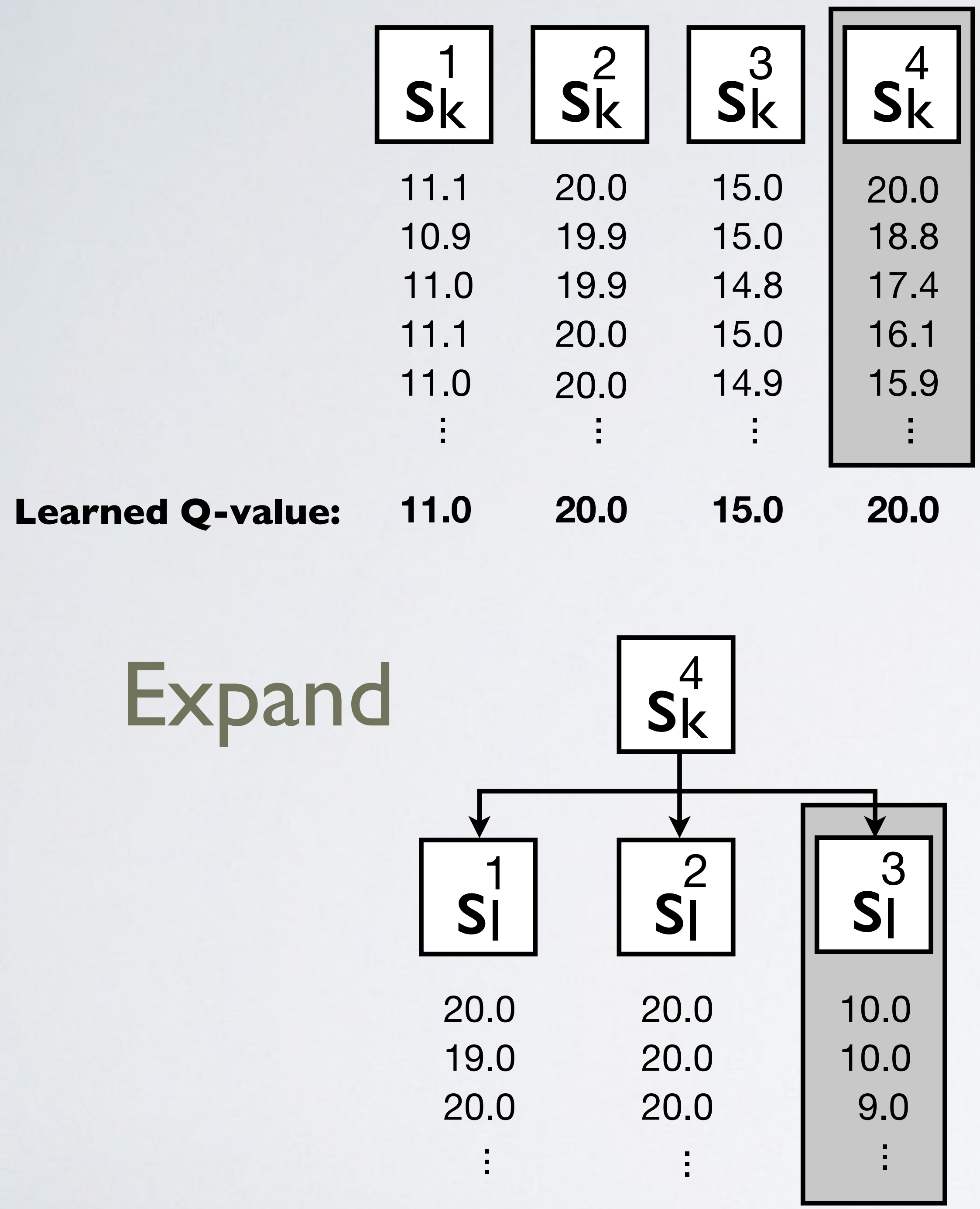
DETECTING RELEVANT STATES





- Changes in reward signal are reflected in the Q-values

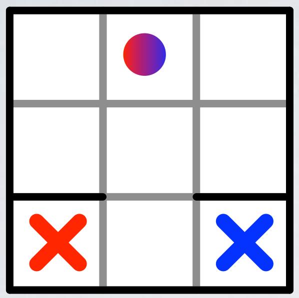
FCQ-LEARNING STATISTICAL TESTS



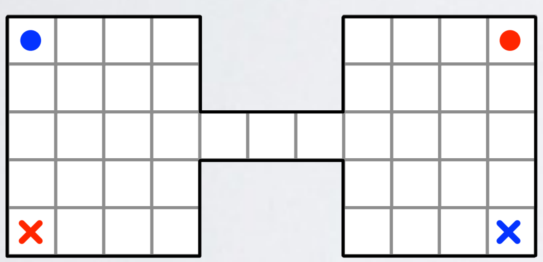
- Agent k has been learning alone, and its Q-values have converged
- Agent k acts independently using only local state information (s_k) in a multi-agent environment
- Performs statistical test against the single agent Q-values
- Samples rewards monte carlo and perform a comparison test to determine what information should be included

$$s_k^4 \Rightarrow \langle s_k^4, s_l^3 \rangle$$

EXPERIMENTAL RESULTS

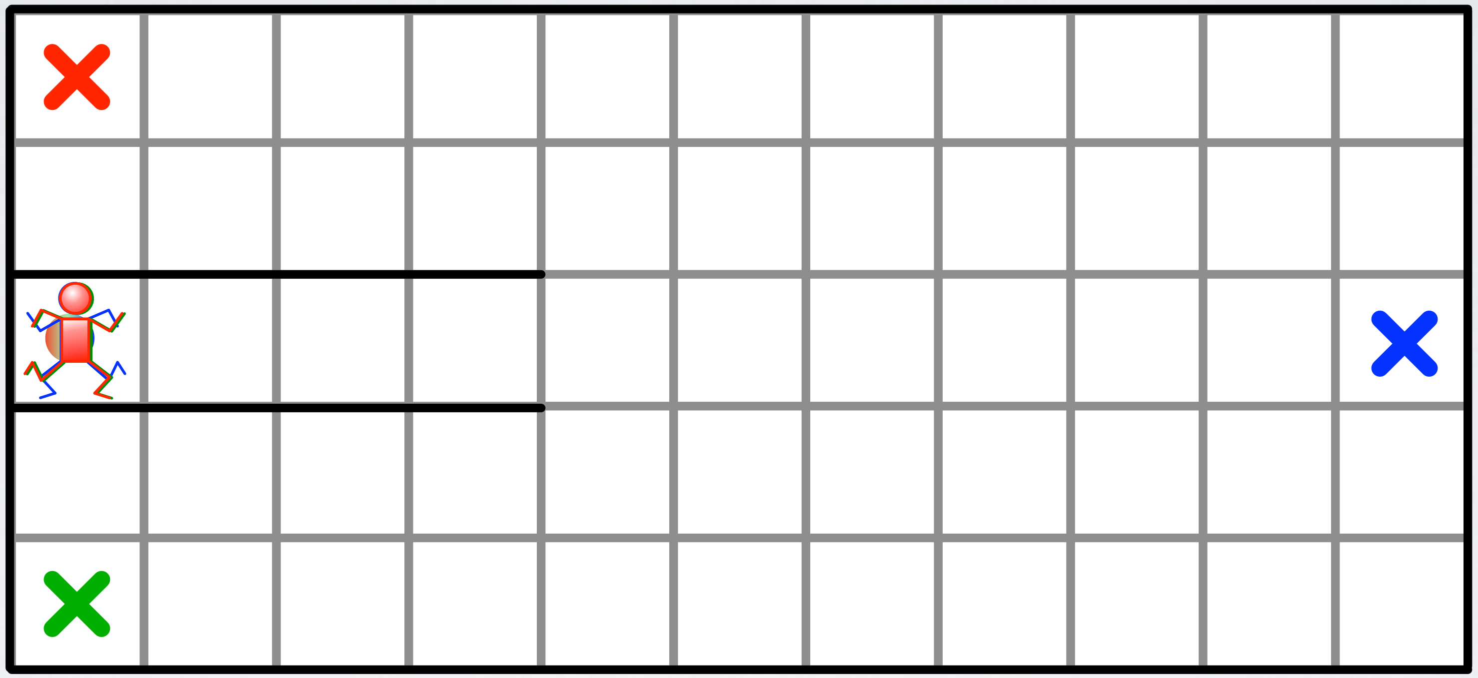


Environment	Algorithm	#states	#actions	#collisions	#steps	reward
Grid_game_2	Indep	9	4	2.4 ± 0.0	22.7 ± 30.4	-24.3 ± 35.6
	JS	81	4	0.1 ± 0.0	$\mathbf{6.3 \pm 0.3}$	$\mathbf{18.2 \pm 0.6}$
	LOC	9.0 ± 0.0	5	1.8 ± 0.0	10.3 ± 2.7	-6.8 ± 8.0
	FCQ	19.4 ± 4.4	4	0.1 ± 0.0	8.1 ± 13.9	17.6 ± 3.7
	FCQ_NI	21.7 ± 3.1	4	0.1 ± 0.0	7.1 ± 6.9	17.9 ± 0.7



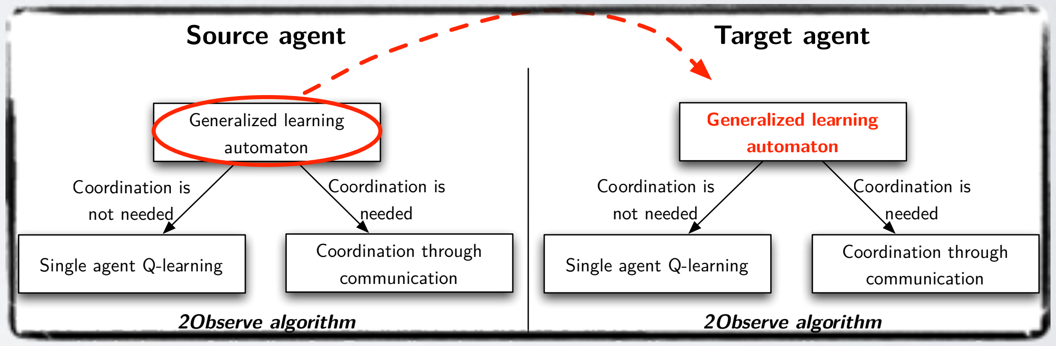
Environment	Algorithm	#states	#actions	#collisions	#steps	reward
Bottleneck	Indep	43	4	n.a.	n.a.	n.a.
	JS	1849	4	0.0 ± 0.0	23.3 ± 30.8	13.1 ± 36.1
	LOC	54.0 ± 0.8	5	1.7 ± 0.6	$167.2 \pm 19,345.1$	$-157.5 \pm 10,3$
	FCQ	124.5 ± 32.8	4	0.1 ± 0.0	$\mathbf{17.3 \pm 1.3}$	$\mathbf{16.6 \pm 0.4}$
	FCQ_NI	135.0 ± 88.7	4	0.2 ± 0.0	19.2 ± 5.6	15.4 ± 2.3

EXPERIMENTAL RESULTS



- Order to reach the goal:
- Red Agent +20
- Blue Agent +20
- Green Agent +20

Transfer Learning



TRANSFER LEARNING

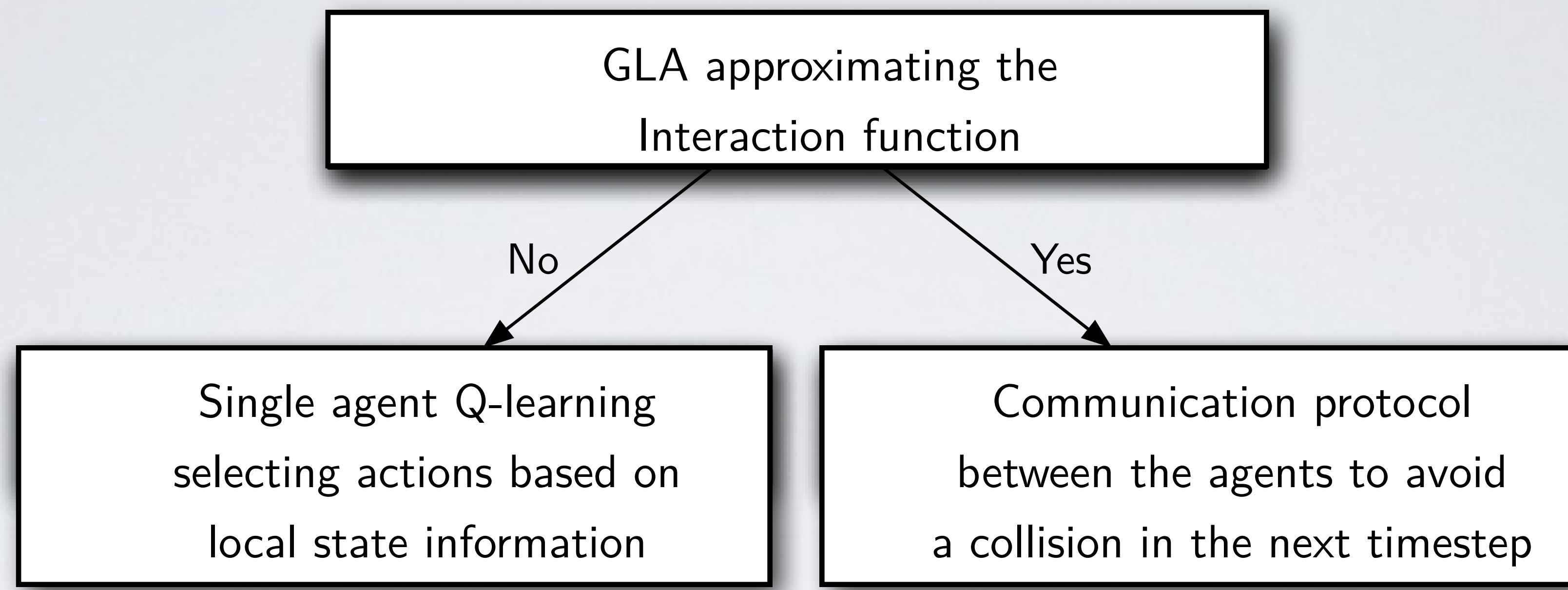
"Transfer of learning occurs when learning in one context enhances (positive transfer) or undermines (negative transfer) a related performance in another context."

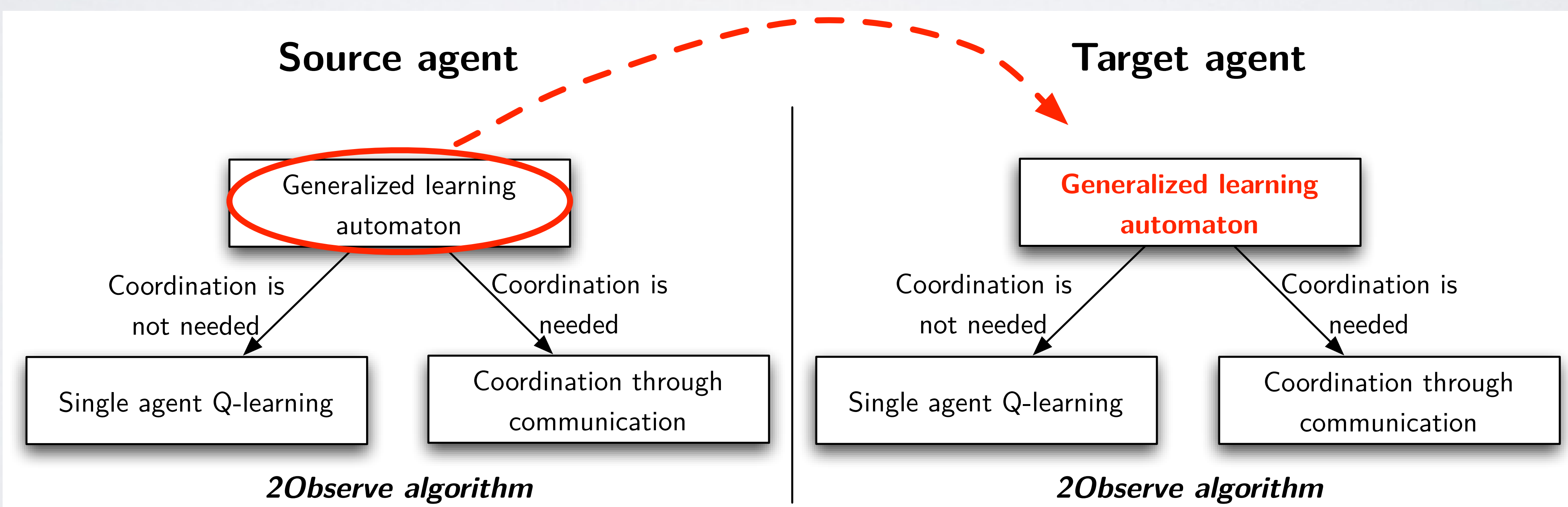
(D. Perkins, G. Salomon, Transfer of Learning, 1992, International Encyclopedia of Education)

MOTIVATIONS FOR TRANSFER LEARNING

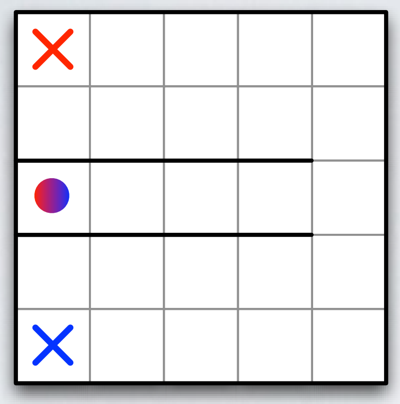
- Learning tabula rasa can be extremely slow
  - Lots of data / time may be needed
  - Every algorithm has biases: why use an uninformed bias?
- Humans always use past knowledge
  - What knowledge is relevant?
  - How can it be effectively leveraged?

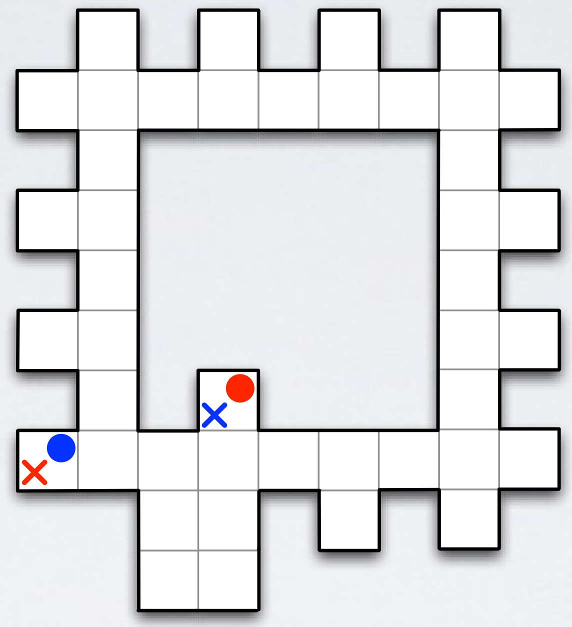
TRANSFER LEARNING WITH 2OBSERVE

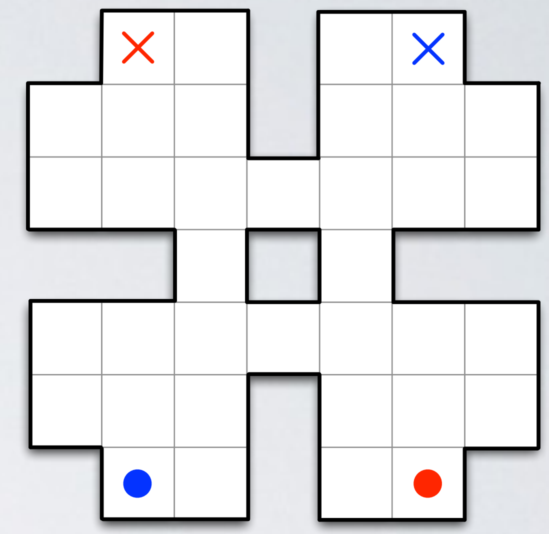


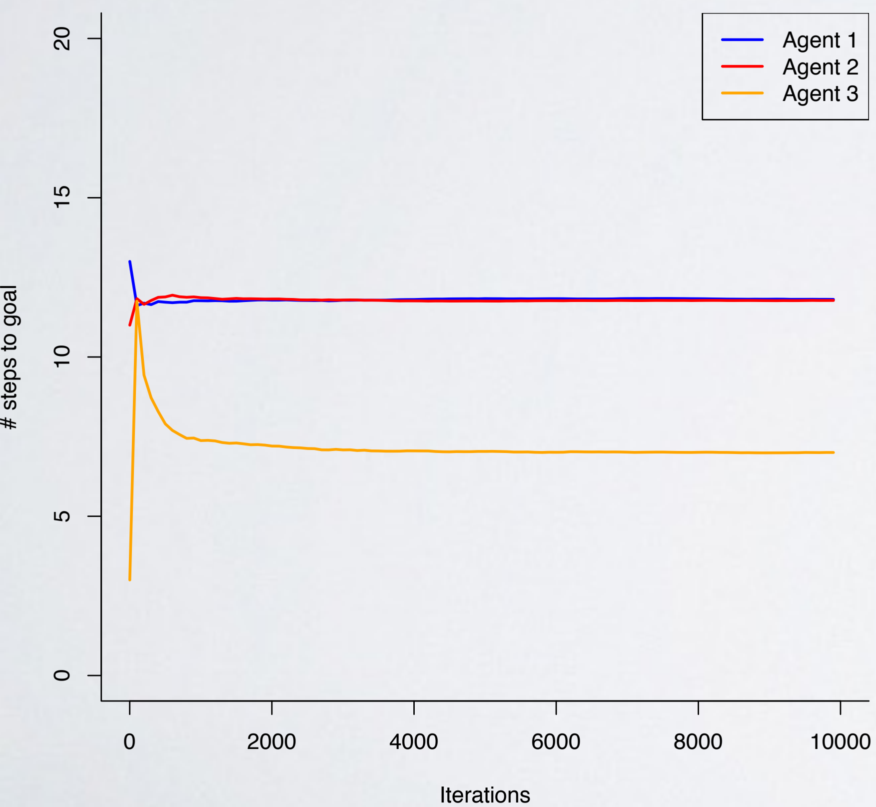


RESULTS

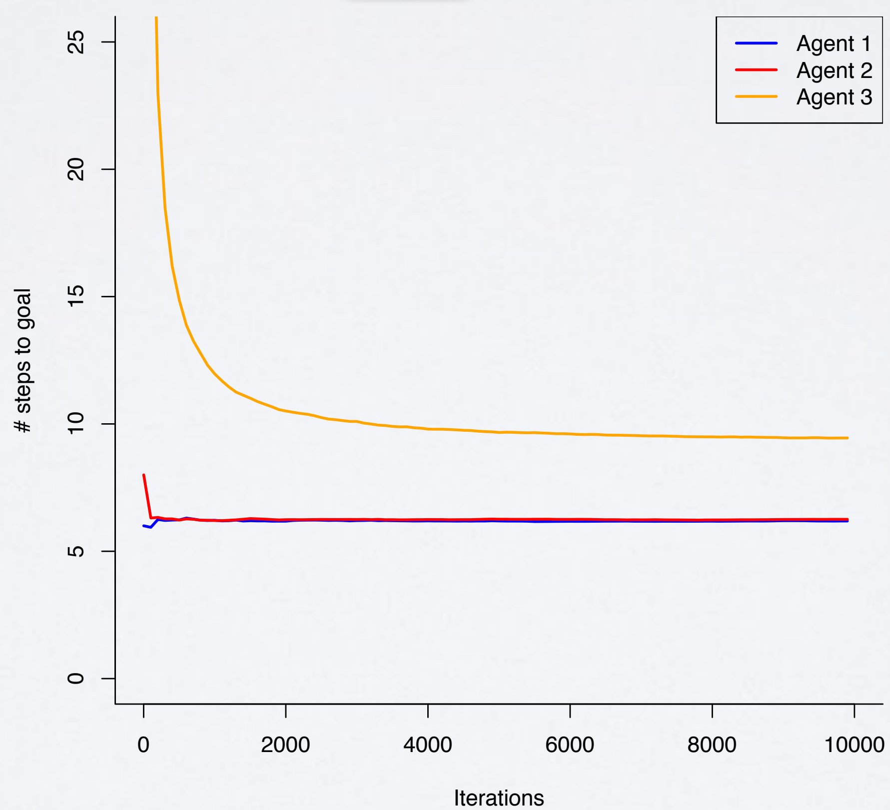




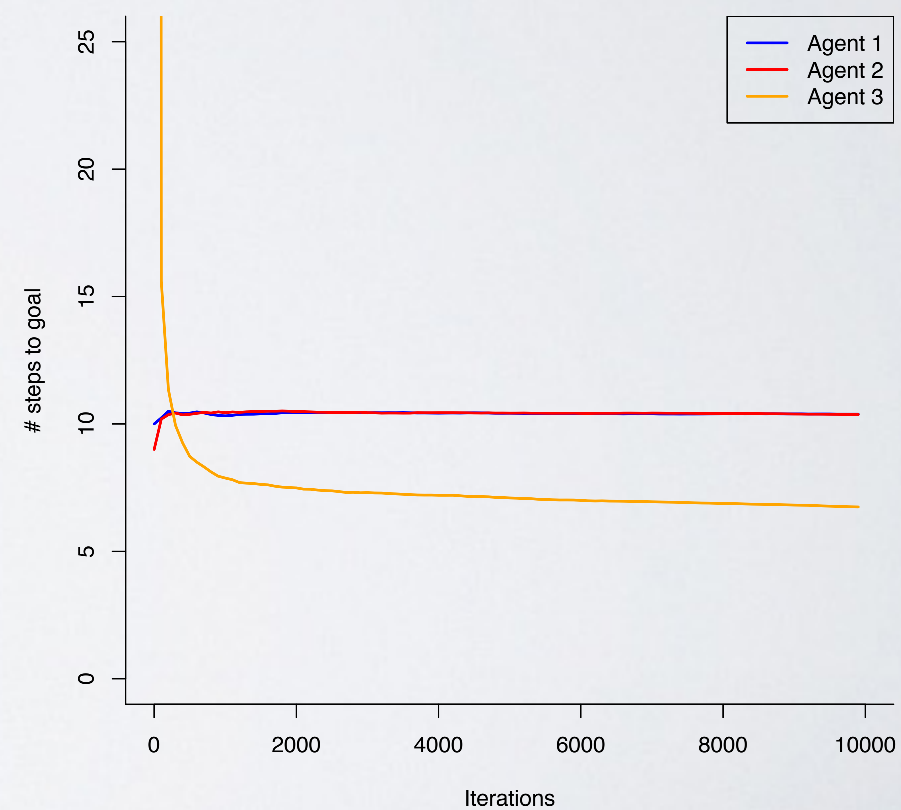




Agent 1
Agent 2
Agent 3
steps to goal
0
5
10
15
20
0
2000
4000
6000
8000
10000
Iterations

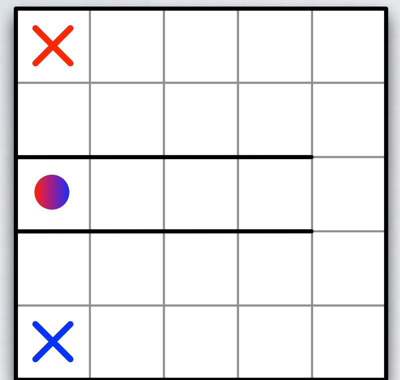


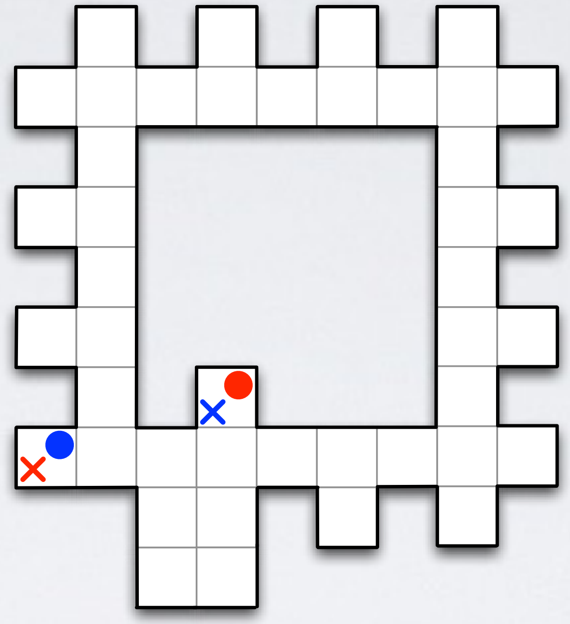
Agent 1
Agent 2
Agent 3
steps to goal
0
5
10
15
20
25
0
2000
4000
6000
8000
10000
Iterations

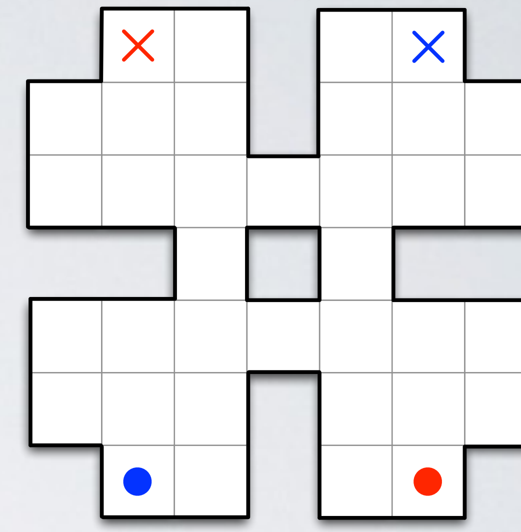


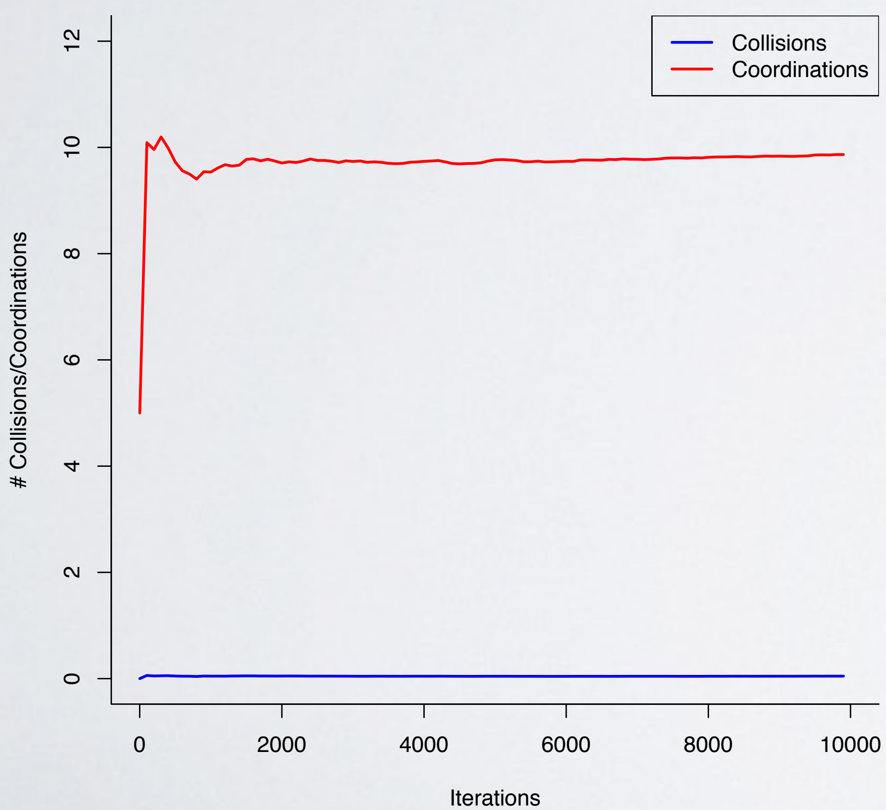
Agent 1
Agent 2
Agent 3
steps to goal
0
5
10
15
20
25
0
2000
4000
6000
8000
10000
Iterations

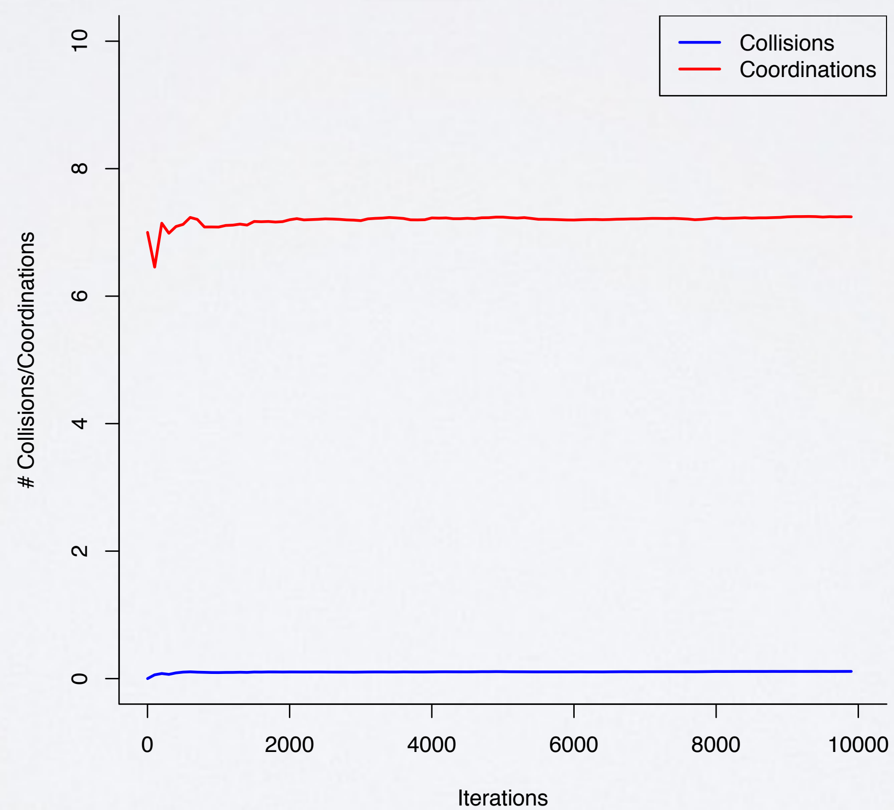
RESULTS (COORDINATION)

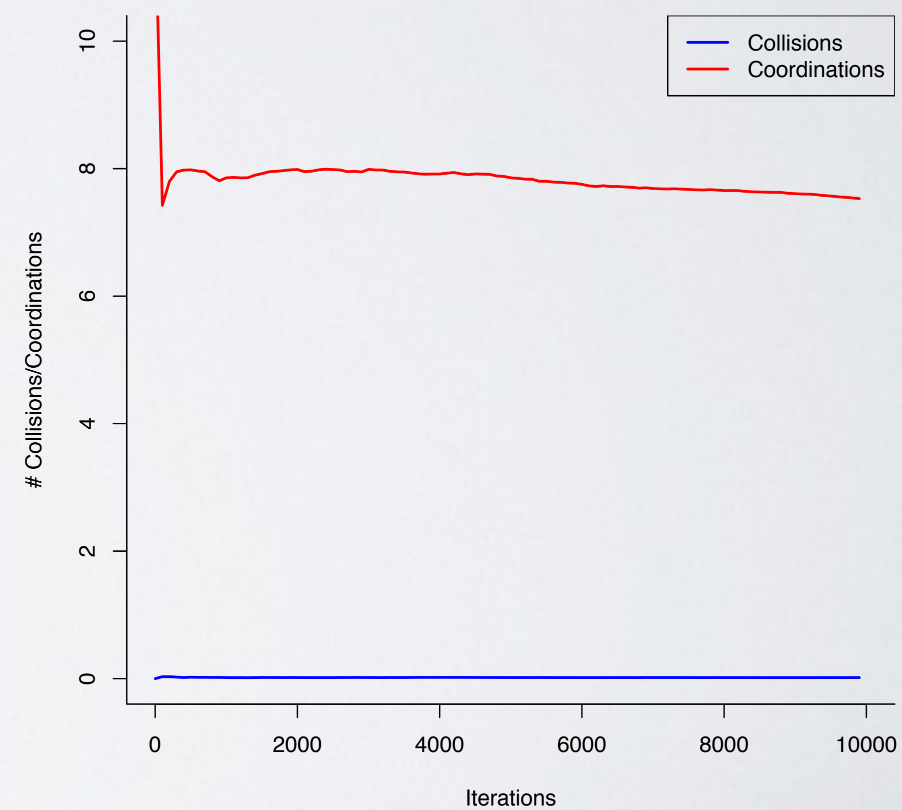






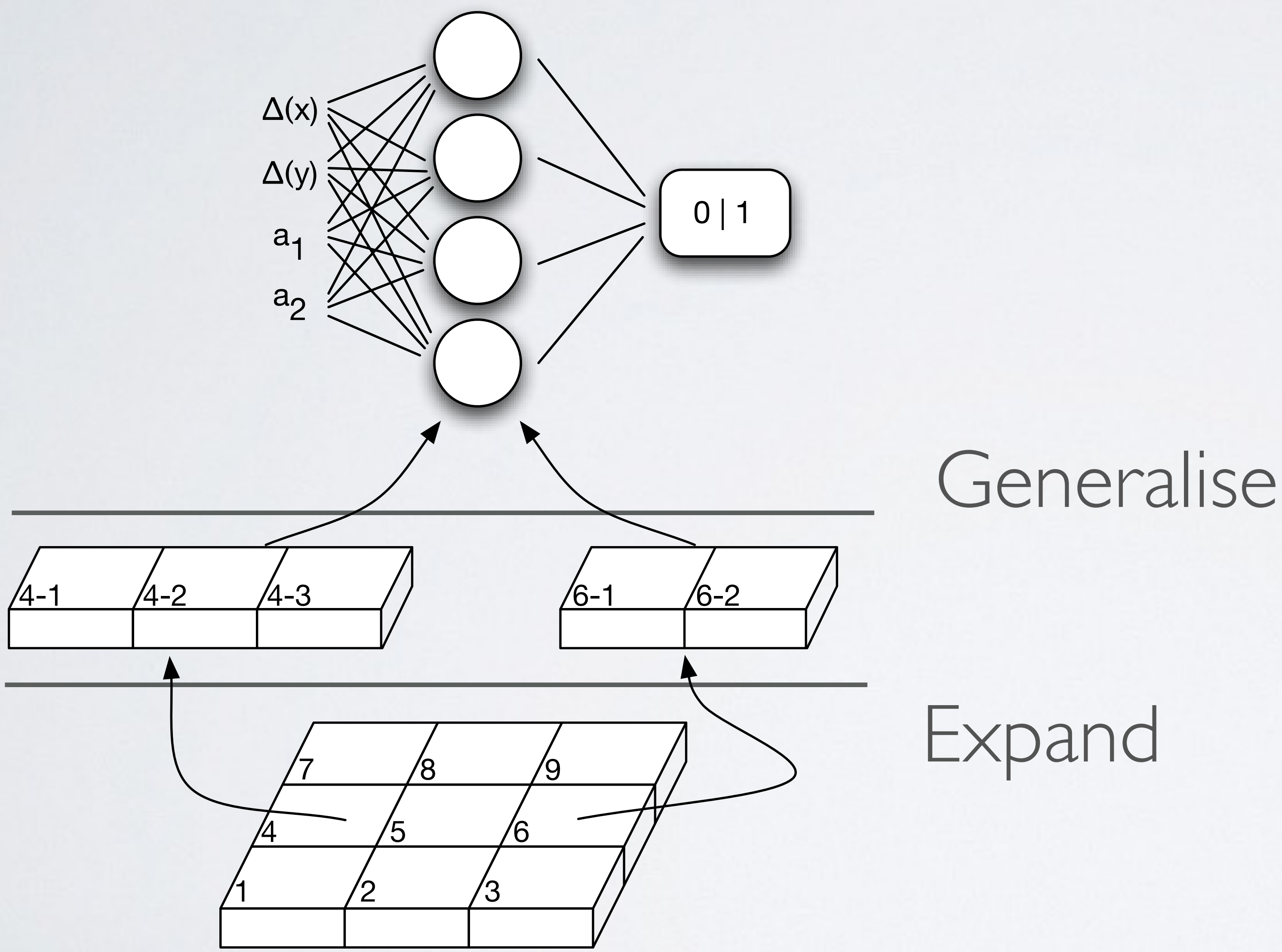




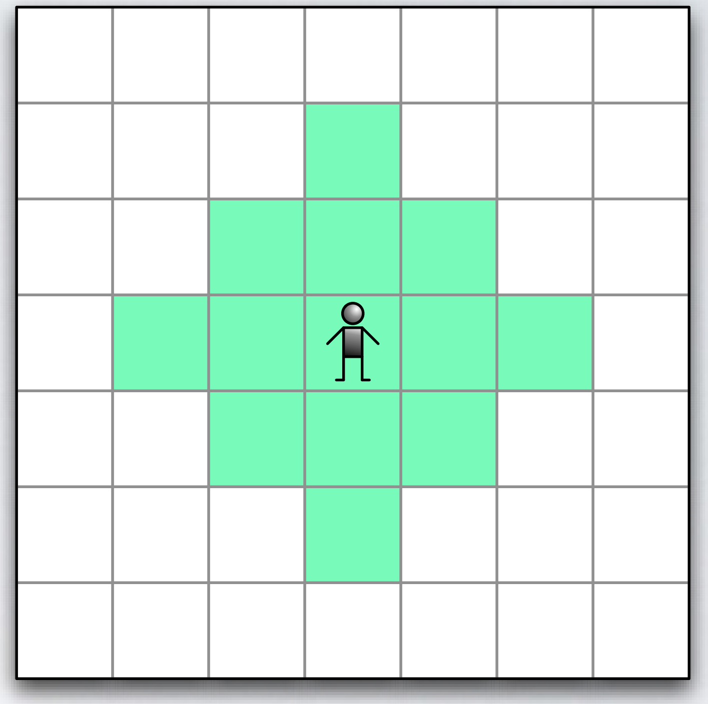


GENERALISATION WITH CQ-LEARNING

Neural network



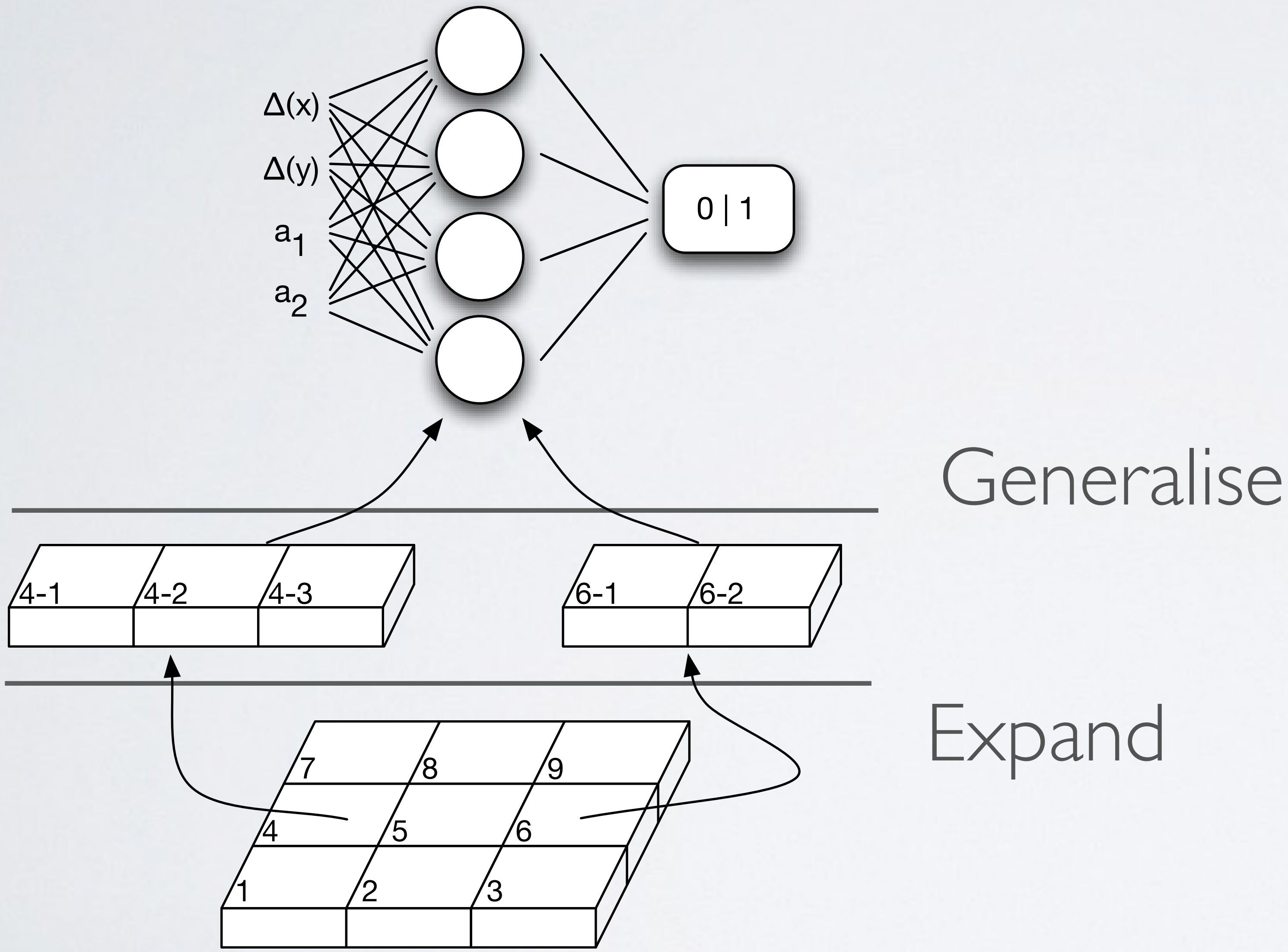
Local state space



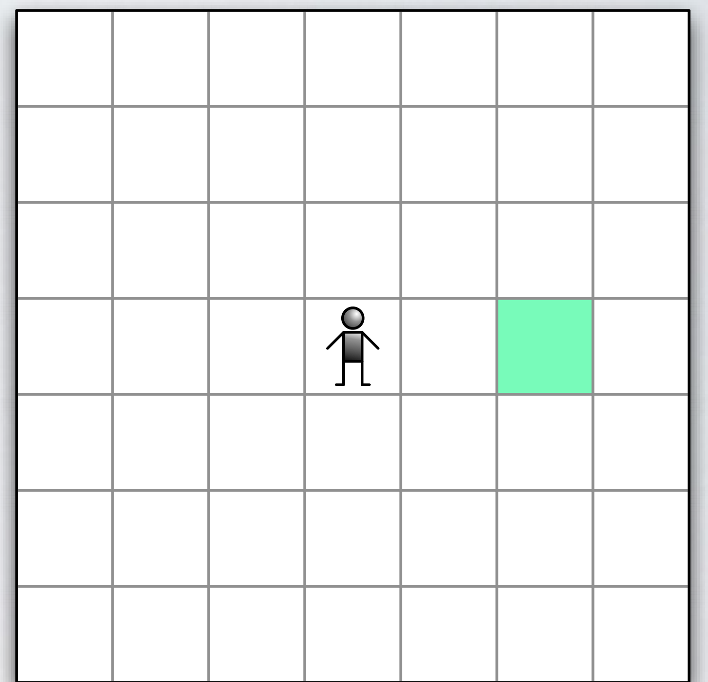
Generalisation learned with 2Observe

GENERALISATION WITH CQ-LEARNING

Neural network

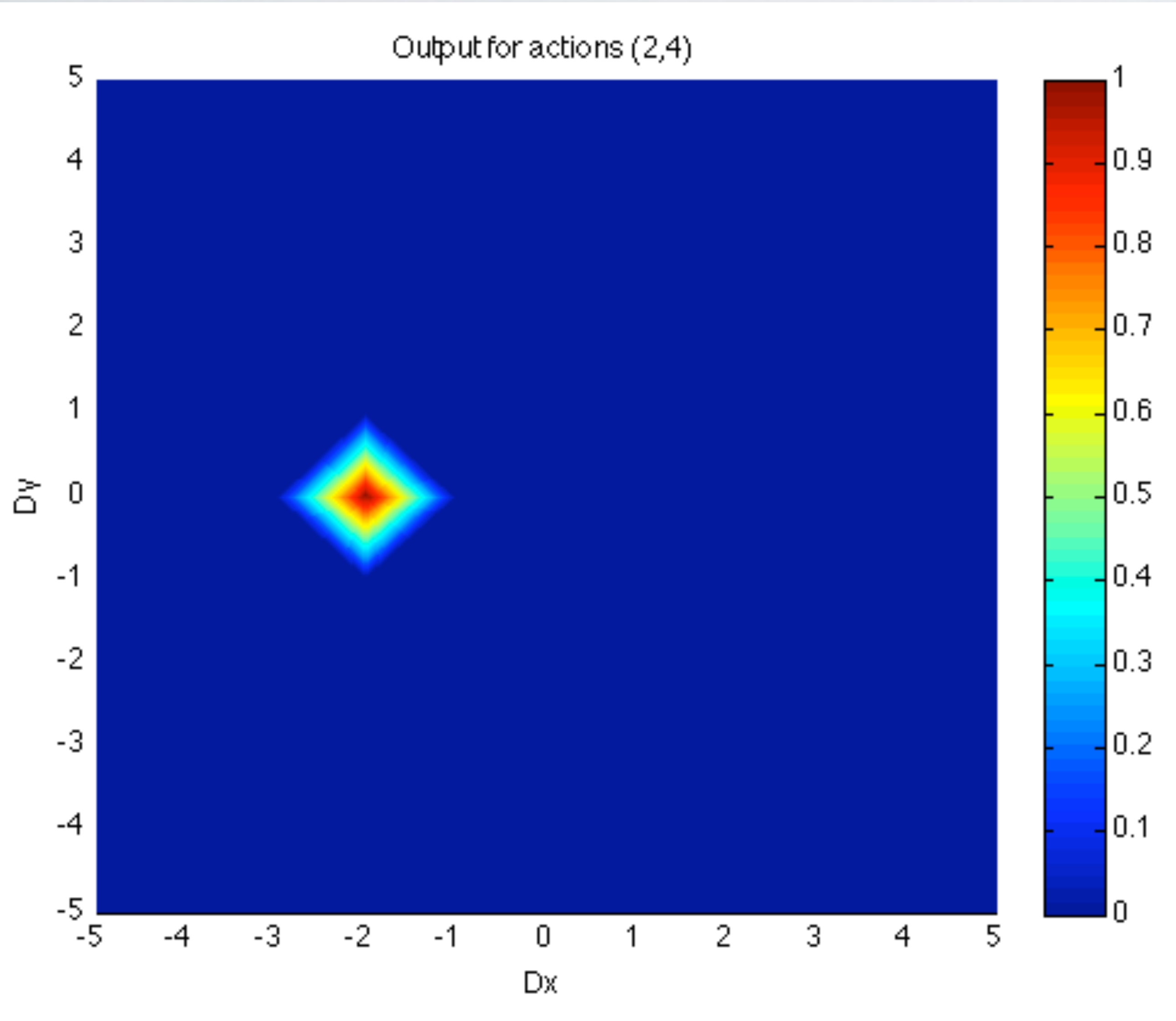


Local state space

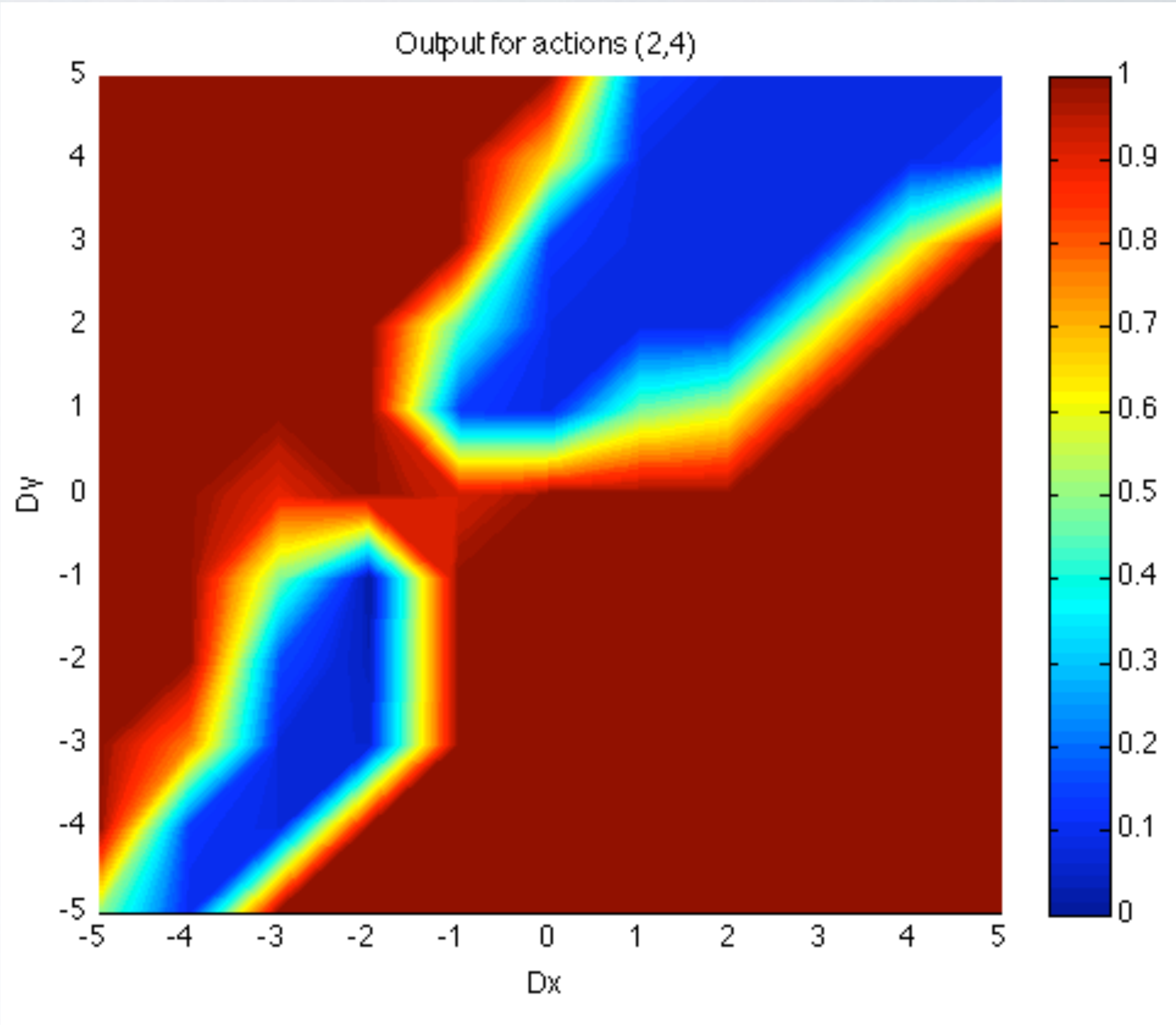


Generalisation learned with CQ-learning

GENERALISATION WITH CQ-LEARNING (2)

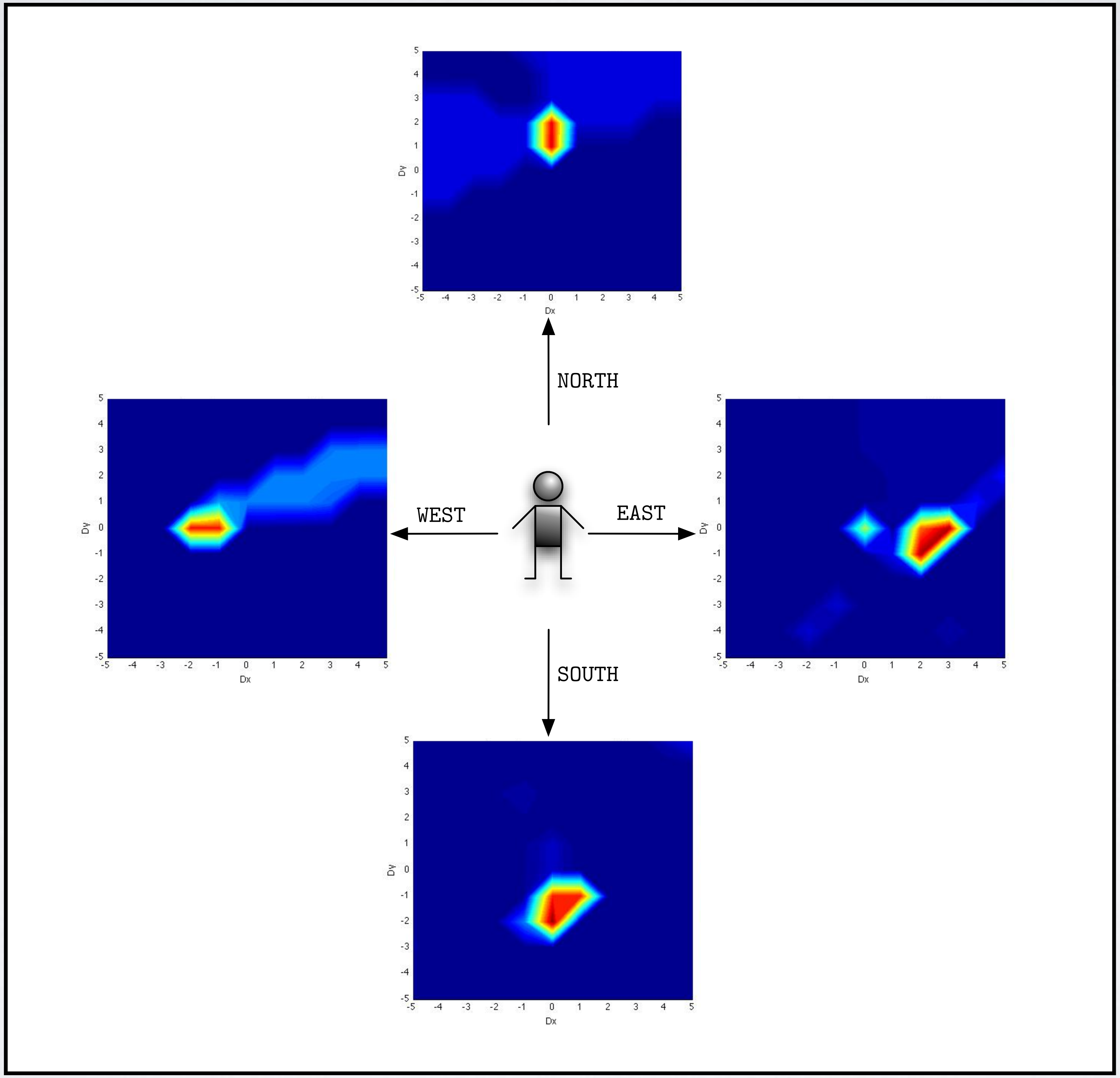


Safe initialisation

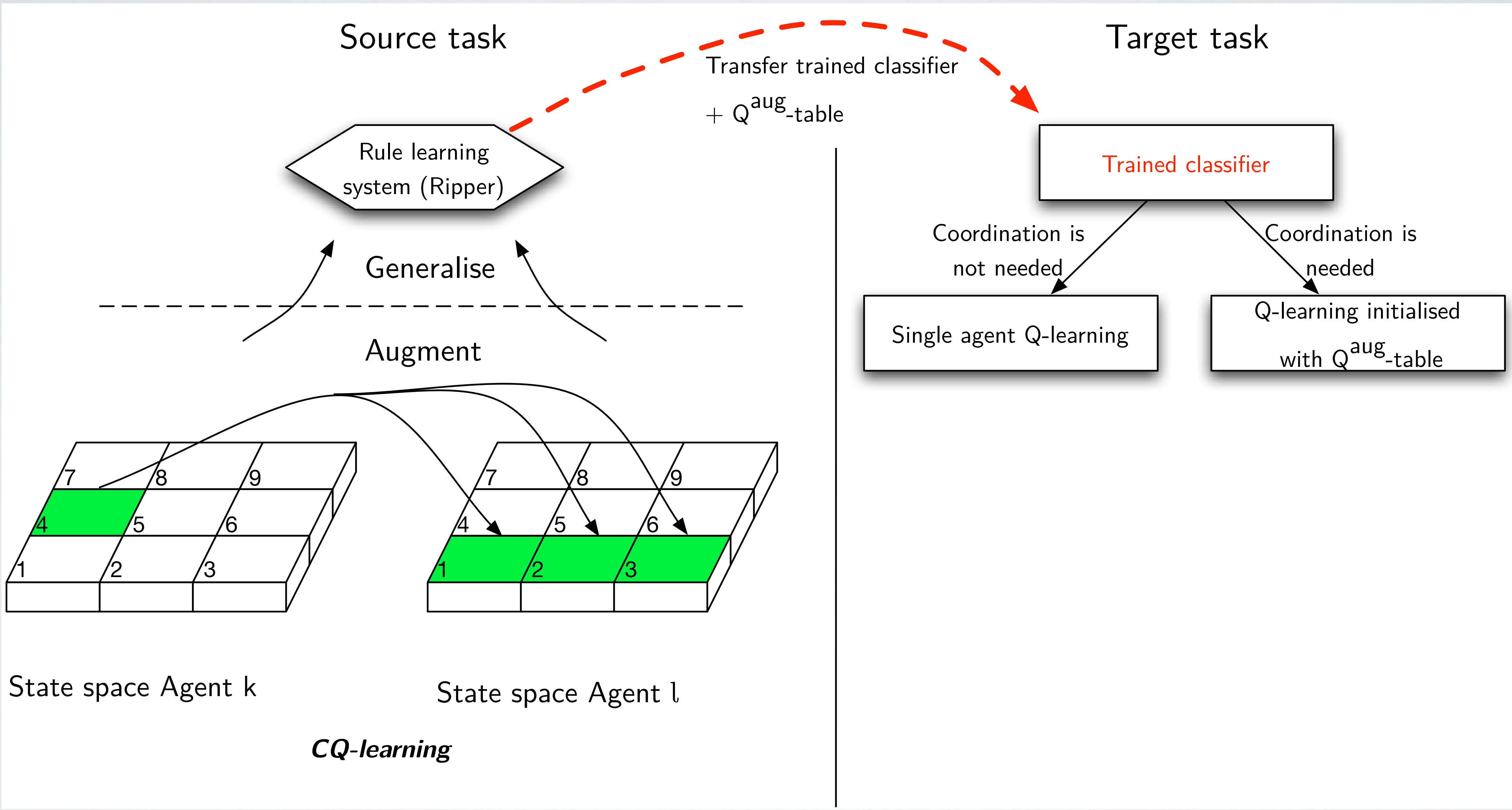


Danger initialisation

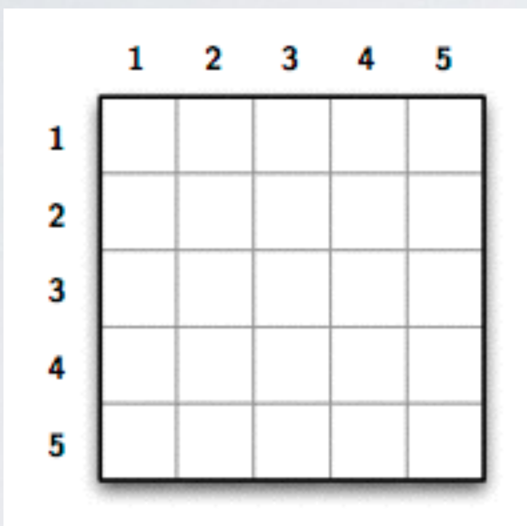
GENERALISATION WITH CQ-LEARNING (2)

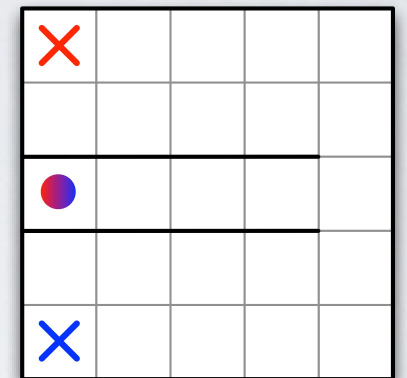


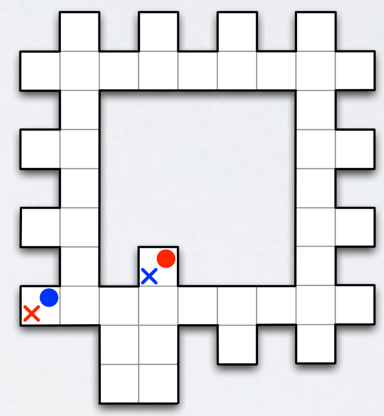
TRANSFER LEARNING WITH CQ-LEARNING

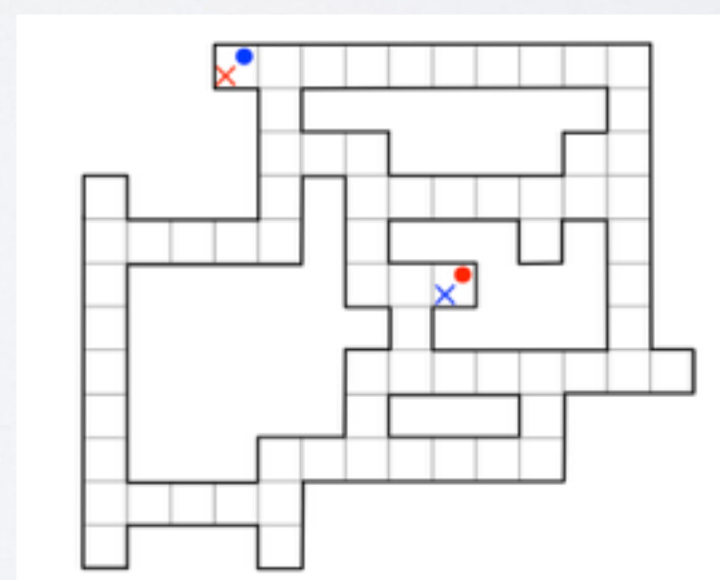


TRANSFER LEARNING WITH CQ-LEARNING (2)

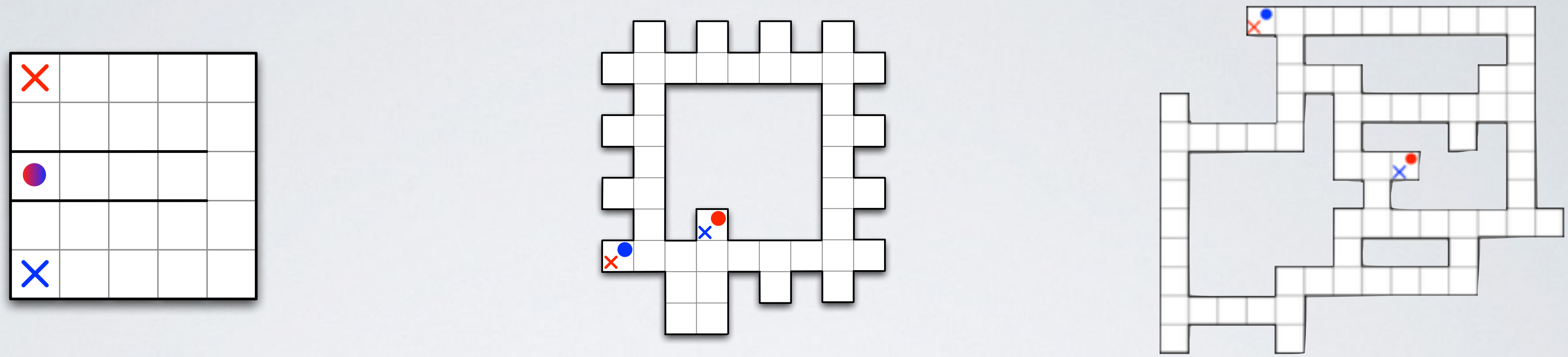


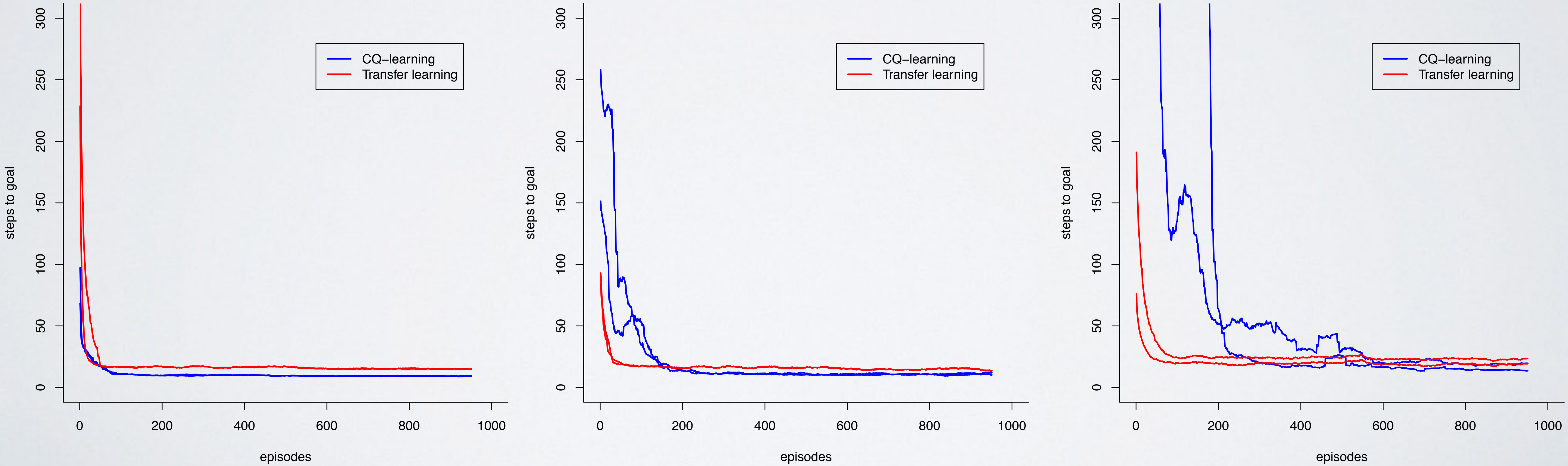






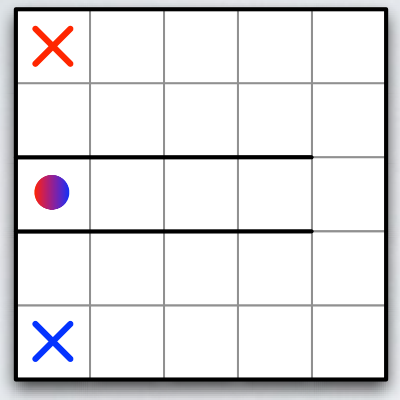
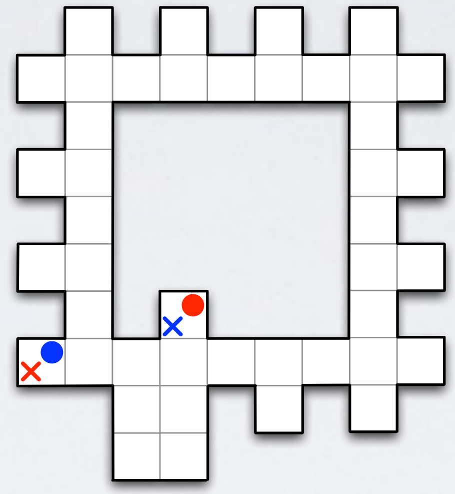
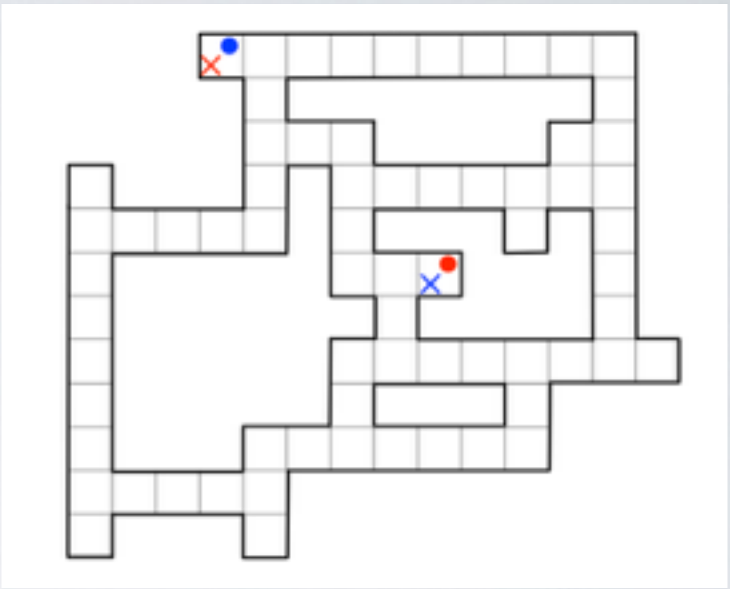
RESULTS

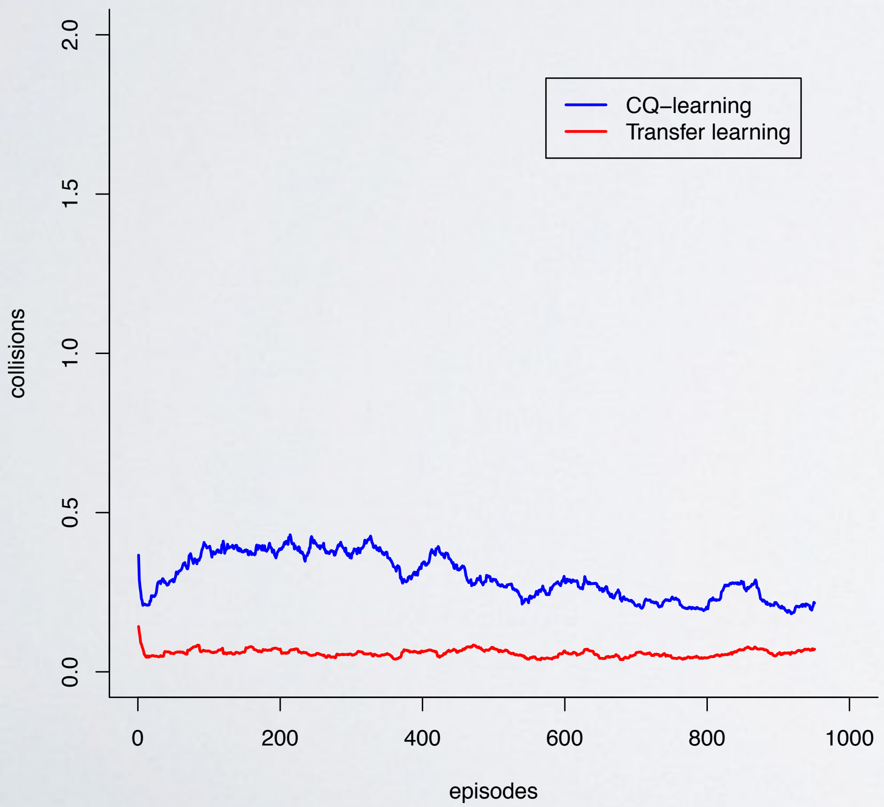




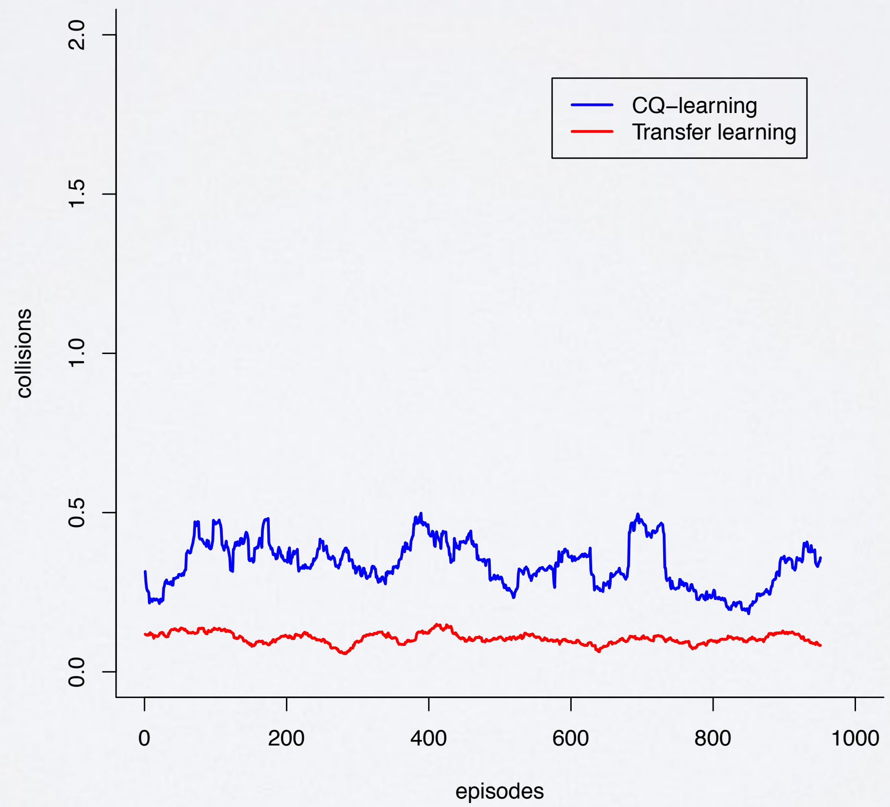
steps to goal
episodes
CQ–learning
Transfer learning
0
50
100
150
200
250
300
0
200
400
600
800
1000

RESULTS (2)

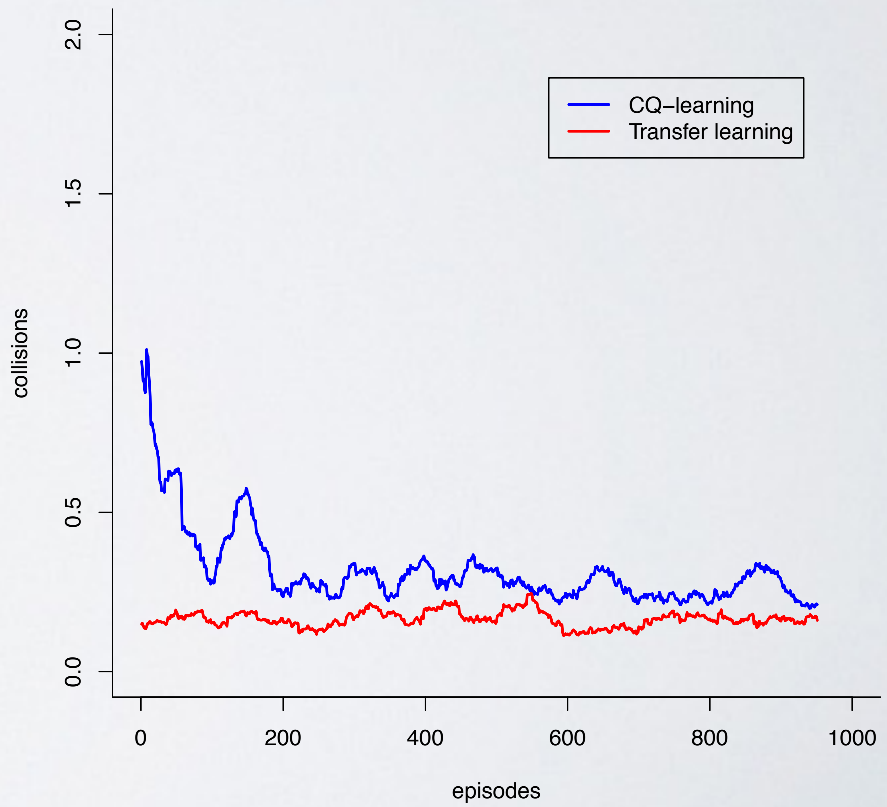






CQ–learning
Transfer learning
collisions
episodes
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000



CQ–learning
Transfer learning
collisions
episodes
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000



CQ–learning
Transfer learning
collisions
episodes
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000

CONCLUSIONS

- In multi-agent environments with sparse interactions, learning these interaction states improves the learning process
- Interaction states can be learned through increased penalties for miscoordination
- GLA can approximate interaction areas relative to the agent
- Interaction states can be identified using statistical tests on the reward signal (immediate + future)
- Information about interaction states can be generalized and transferred between agents and environments