Scalable and Robust Multi-Agent Reinforcement Learning

Chris Amato
Northeastern University


Shayegan Omidshafiei


Yuchen Xiao

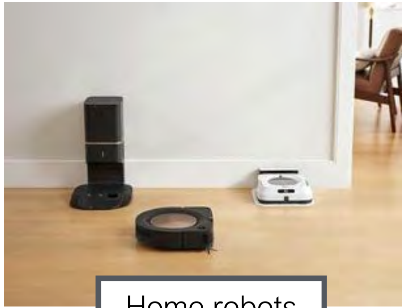
Northeastern University

Multi-agent systems are (going to be) everywhere

Drone delivery


UAV surveillance


Autonomous cars


Home robots


Cyber security


Smart energy grids

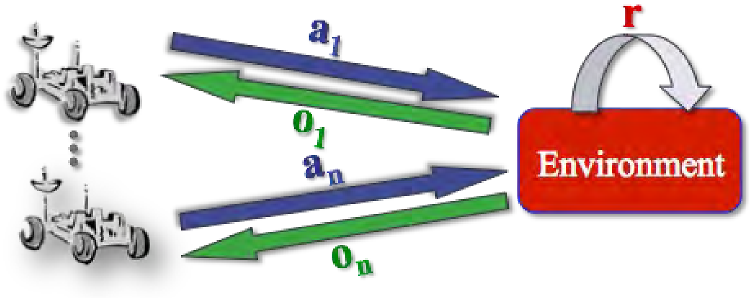
Northeastern University

Uncertainties

- These real-world problems have several forms of uncertainty:
  - Outcome uncertainty
  - Sensor uncertainty
  - Communication uncertainty

Multiple cooperating agents

- Decentralized partially observable Markov decision process (Dec-POMDP)
  Bernstein et al., 02
    - Extension of the single agent MDP and POMDP models
    - Multiagent sequential decision-making under uncertainty
- At each stage, each agent takes an action and receives:
    - A local observation
    - A joint immediate reward



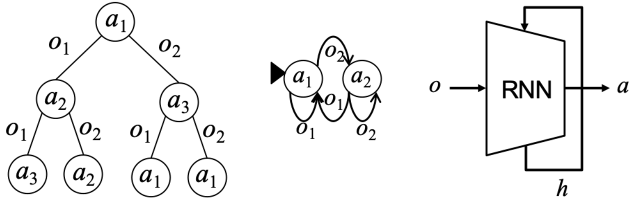
Dec-POMDP model

- A Dec-POMDP can be defined with the tuple: $<I, S, \{A_i\}, T, R, \{\Omega_i\}, O>$
  - I, a finite set of agents
  - S, a finite set of states with designated initial state distribution b^0
  - A_i, each agent's finite set of actions
  - T, the state transition model: $P(s' \mid s, \vec{a})$
  - R, the reward model: $R(s, \vec{a})$
  - Ω_i, each agent's finite set of observations
  - O, the observation model: $P(\vec{o} \mid s', \vec{a})$
  - h, horizon or discount γ

Note: Functions depend on all agents

Dec-POMDP solutions

- A *local policy* for each agent, $H \to A$, maps its observation sequences to actions
  - State is unknown, so beneficial to remember history

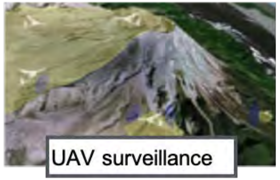


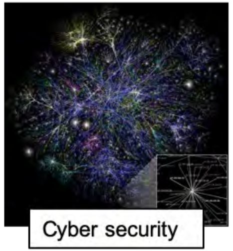
- Policy representations:
  - *one for each agent*
  - Some examples: Policy trees, finite-state controllers, recurrent networks
- Evaluation: $V(q,s) = R(s,a_q) + \gamma \sum_{s',o} \Pr(s'|s,a_q) \Pr(o|s',a) V(q_o,s')$
  - Starting from a set of nodes q, taking the associated actions and transitions
- Goal: maximize expected cumulative reward over a finite or infinite horizon (use discount factor, γ, in infinite case)

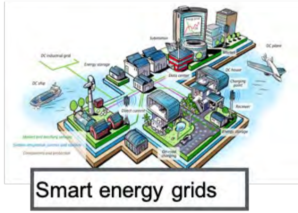
Northeastern University

Dec-POMDPs are general

- Any cooperative problem with outcome, sensor and communication uncertainty
- A common framework for multi-agent RL (MARL)







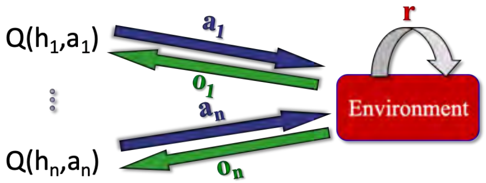
- The only more general framework is the partially observable stochastic game
- This generality means the solution must consider partial observability and other agents

Overview

- How do we learn solutions to Dec-POMDPs?
- Scalable to large domains?
  - How do we integrate deep RL methods into MARL?
- How can we scale to large horizons?

Decentralized learning

- Many MARL methods are *centralized learning for decentralized execution* (e.g., use full state information, centralized value function)
- Can also do *decentralized learning for decentralized execution*
  - More scalable
  - Can apply to online learning
  - Can directly apply single-agent RL to each agent (e.g., independent learning)
  - Problem is now non-stationary from the perspective of each agent



Decentralized Hysteretic DQN (Dec-HDRQN)

Omidshafiei, Pazis, Amato, How and Vian - ICML 17

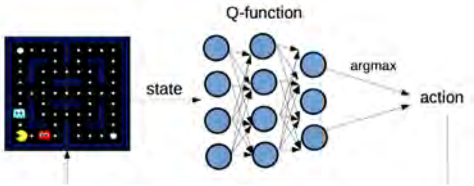
- Traditional Q-learning: estimate Q-value with (*x* can be state, observation or history)

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

- Hysteresis (Matignon et al., IROS 07): two learning rates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (with $\boldsymbol{\beta} < \boldsymbol{\alpha}$) — Helps with nonstationarity

$$Q(x,a) \leftarrow \begin{cases} Q(x,a) + \beta\delta & \text{if } \delta \leq 0 \\ Q(x,a) + \alpha\delta & \text{otherwise} \end{cases}$$

- Deep Q-Networks (DQN) (Mnih et al., Nature 15) uses a neural net for function approximation — Helps with scalability

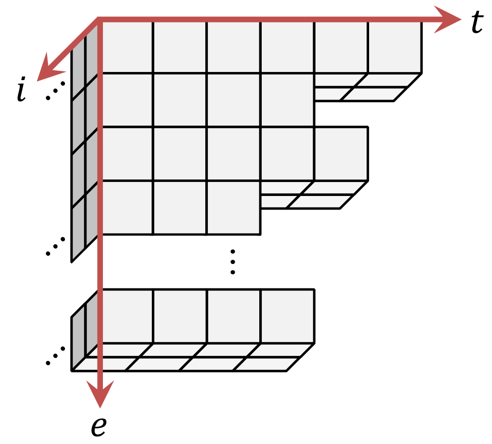


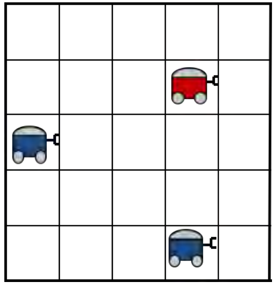
- DRQN (Hausknecht and Stone, arXiv 15) adds a recurrent layer for memory — Helps with partial observability

Northeastern University

Synchronizing samples

- Concurrent Experience Replay Trajectories (CERTs)
- Helps stabilize learning
- Can be implemented in a decentralized manner



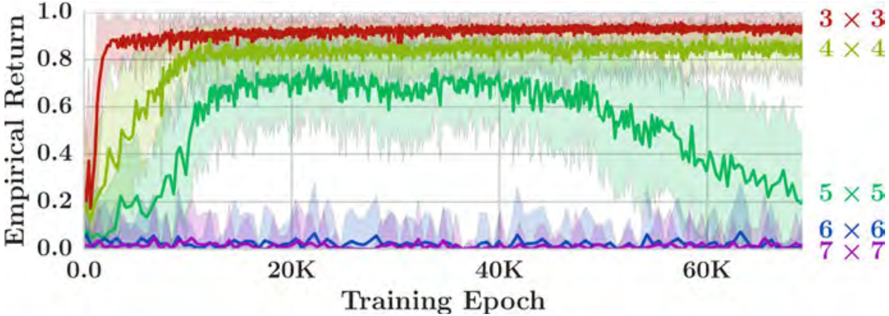


Target capture

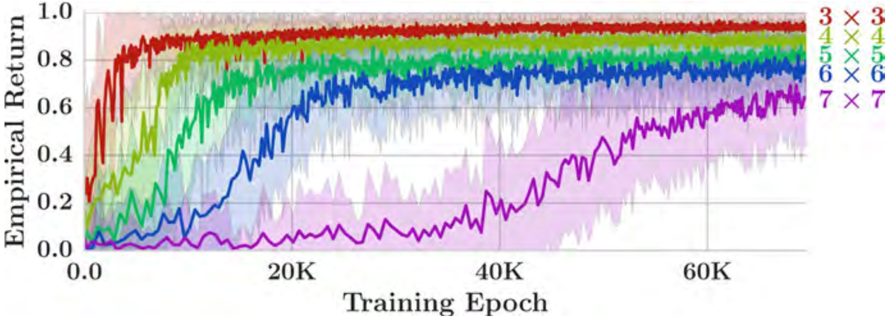
Results

Omidshafiei, Pazis, Amato, How and Vian - ICML 17

Dec-DRQN
(without hysteresis)



Dec-HDRQN
(our method)



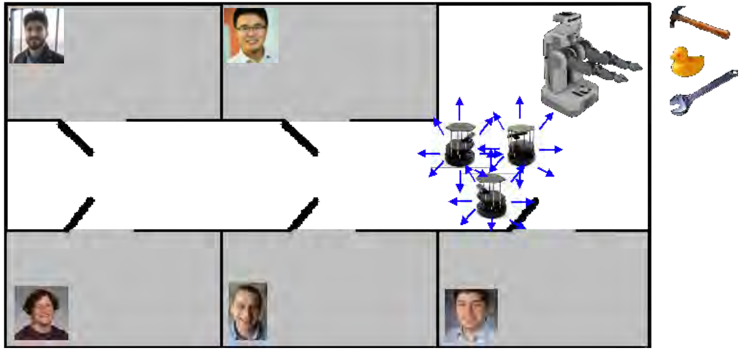
Our method is more stable and scalable

This paper also developed a multitask version of this algorithm

Scaling up: macro-actions

Amato, Konidaris and Kaelbling - AAMAS 14
Amato, Konidaris, How and Kaelbling - JAIR 19

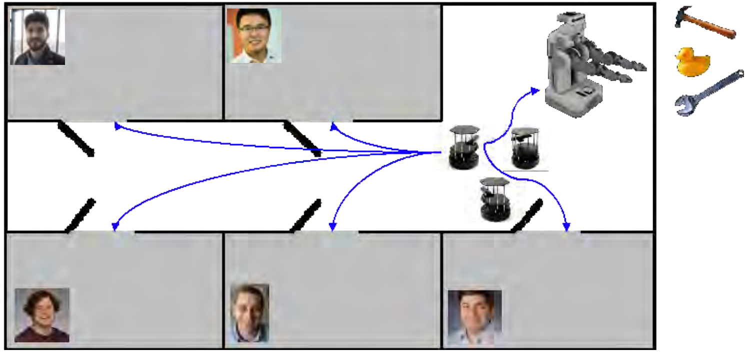
- Dec-POMDP methods model and solve at a low level (actions as control inputs)



Scaling up: macro-actions

Amato, Konidaris and Kaelbling - AAMAS 14
Amato, Konidaris, How and Kaelbling - JAIR 19

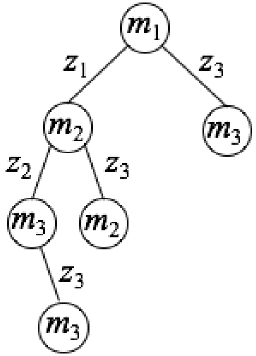
- Dec-POMDP methods model and solve at a low level (actions as control inputs)
- This is intractable (and unnecessary!) for real-world systems
- Often easy to plan for subgoals/subtasks
  - Set initial and terminal conditions (i.e., states)
  - Have expertly programmed controllers
- Allows for asynchronous decision-making
- Resulting model: MacDec-POMDP (macro-action Dec-POMDP)

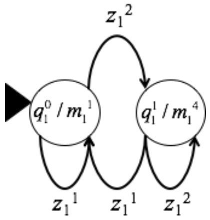


Northeastern University

Macro-action solution representations

- Can extend policy representations to macro-action case
  - m = macro-action
  - z = high-level observation
- Finite-state controllers μ for each agent i defined with node set Q_i:
  - Action selection, λ: $Q_i \rightarrow M_i$
  - Node transitions, δ: $Q_i \times Z_i \rightarrow Q_i$
  - An initial node: $q_{i^0} \in Q_i$
- But macro-actions finish at different times!
- Developed decentralized partially observable *semi*-Markov decision process (Dec-POSMDP)





Omidshafiei, Agha, Amato, Liu and How - IJRR 17

$$V^{\mu}(q,s) = R(s,\lambda(q)) + \sum_{k}^{\infty} \gamma^k \sum_{s',o} \Pr(s',k|s,\lambda(q)) \Pr(o|s',\lambda(q)) V^{\mu}(\delta(q,o),s')$$

Northeastern University

Macro-action deep MARL?

- All current deep MARL methods assume synchronized (i.e., primitive) actions
- It isn't clear how to incorporate asynchronous macro-actions into deep MARL methods

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

We assume we can get macro-action-level information every (primitive) time step

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

Example:

Time		t=1
Agent_1		z_1
		m_1
		z_1
		r_1
Agent_2		z_6
		m_4
		z_7
		r_1
Joint Reward		r_1

Northeastern University

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

Example:

Time	t=1	t=2
Agent_1	z_1	z_1
	m_1	m_1
	z_1	z_1
	r_1	$\sum_{i=1}^{2} r_i$
Agent_2	z_6	z_7
	m_4	m_5
	z_7	z_7
	r_1	r_2
Joint Reward	r_1	r_2

Northeastern University

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

Example:

Time		t=1	t=2	t=3
Agent_1		z_1	z_1	z_1
		m_1	m_1	m_1
		z_1	z_1	z_2
		r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$
Agent_2		z_6	z_7	z_7
		m_4	m_5	m_5
		z_7	z_7	z_7
		r_1	r_2	$\sum_{i=2}^{3} r_i$
Joint Reward		r_1	r_2	r_3

Northeastern University

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

Example:

Time	t=1	t=2	t=3	t=4
Agent_1	z_1	z_1	z_1	z_2
	m_1	m_1	m_1	m_2
	z_1	z_1	z_2	z_2
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4
Agent_2	z_6	z_7	z_7	z_7
	m_4	m_5	m_5	m_5
	z_7	z_7	z_7	z_8
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$
Joint Reward	r_1	r_2	r_3	r_4

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Concurrent Experience Replay Trajectories (Mac-CERTs)

- ➢ Collect the concurrent macro-action-observation experiences of agents;
- ➢ Transition tuple of each agent i is defined as $\langle z, m, z', r^c \rangle_i$, where $r^c = \sum_{t=t_m}^{\tau} r_t$

Note that, the next macro-observation z' is set as same as z if the macro-action is still under running.

Example:

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_2	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3	z_3
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8	$\sum_{i=8}^{9} r_i$	$\sum_{i=8}^{10} r_i$
Agent_2	z_6	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_5	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$	r_9	$\sum_{i=9}^{10} r_i$
Joint Reward	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}

Generating concurrent trajectories

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_2	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3	z_3
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8	$\sum_{i=8}^{9} r_i$	$\sum_{i=8}^{10} r_i$
Agent_2	z_6	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_5	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$	r_9	$\sum_{i=9}^{10} r_i$
Joint Reward	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}

Generating concurrent trajectories

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_2	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3	z_3
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8	$\sum_{i=8}^{9} r_i$	$\sum_{i=8}^{10} r_i$
Agent_2	z_6	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_5	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$	r_9	$\sum_{i=9}^{10} r_i$
Joint Reward	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}

Sample concurrent trajectories for each agent

Generating concurrent trajectories

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_2	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3	z_3
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8	$\sum_{i=8}^{9} r_i$	$\sum_{i=8}^{10} r_i$
Agent_2	z_6	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_5	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$	r_9	$\sum_{i=9}^{10} r_i$
Joint Reward	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}

Agent_1	z_1	z_2	z_2	z_2	z_2	z_3
	m_1	m_2	m_2	m_2	m_2	m_3
	z_2	z_2	z_2	z_2	z_3	z_3
	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8
Agent_2	z_7	z_7	z_8	z_9	z_9	z_9
	m_5	m_5	m_6	m_7	m_7	m_7
	z_7	z_8	z_9	z_9	z_9	z_{11}
	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$

Sample concurrent trajectories for each agent

Northeastern University

Generating concurrent trajectories

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_2	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_2	z_2	z_2	z_2	z_3	z_3	z_3	z_3
	r_1	$\sum_{i=1}^{2} r_i$	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8	$\sum_{i=8}^{9} r_i$	$\sum_{i=8}^{10} r_i$
Agent_2	z_6	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_5	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_7	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
	r_1	r_2	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$	r_9	$\sum_{i=9}^{10} r_i$
Joint Reward	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}

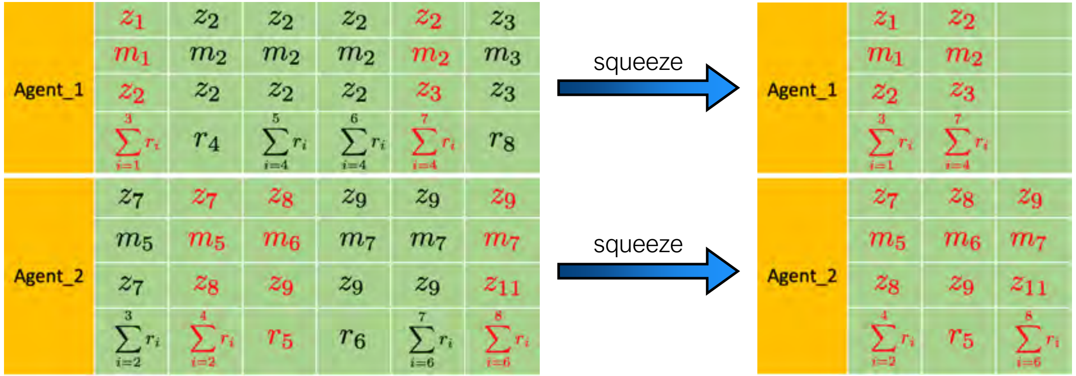
Agent_1	z_1	z_2	z_2	z_2	z_2	z_3
	m_1	m_2	m_2	m_2	m_2	m_3
	z_2	z_2	z_2	z_2	z_3	z_3
	$\sum_{i=1}^{3} r_i$	r_4	$\sum_{i=4}^{5} r_i$	$\sum_{i=4}^{6} r_i$	$\sum_{i=4}^{7} r_i$	r_8
Agent_2	z_7	z_7	z_8	z_9	z_9	z_9
	m_5	m_5	m_6	m_7	m_7	m_7
	z_7	z_8	z_9	z_9	z_9	z_{11}
	$\sum_{i=2}^{3} r_i$	$\sum_{i=2}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	$\sum_{i=6}^{8} r_i$

Identify when macro-actions change

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent

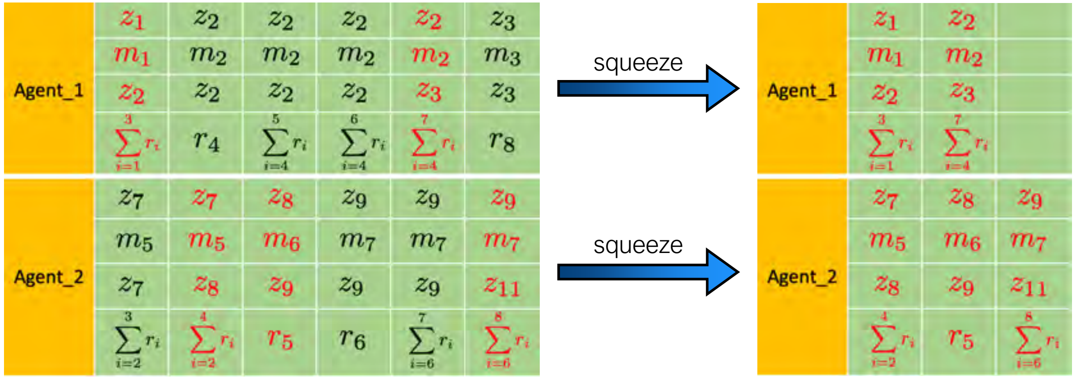


Many possibilities, but we throw away time info

Decentralized learning

Xiao, Hoffman and Amato - CoRL 19

Using Mac-CERTs to learn macro-action-value function for each agent



Can now use Decentralized Hysteretic DRQN (Dec-HDRQN) to learn each agent's **macro-action-value function** $Q_{\theta_i}(h, m)$, using **squeezed sequential experiences**, by minimizing the loss:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{<z,m,r^c,z'>_i \sim \mathcal{D}}\Big[\big(y_i - Q_{\theta_i}(h,m)\big)^2\Big], \text{ where } y_i = r^c + \gamma Q_{\theta_i^-}\big(h', \arg\max_{m'} Q_{\theta_i}(h', m')\big)$$

The double Q version

Centralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Joint Experience Replay Trajectories (Mac-JERTs)

- ➢ Collect the joint macro-action-observation experiences of agents;
- ➢ Joint transition tuple is defined as $\langle \vec{z}, \vec{m}, \vec{z}', \vec{r}^c \rangle$, where $\vec{r}^c = \sum_{t=t_{\vec{m}}}^{\vec{\tau}} r_t$

Example:

Time	t=1
Agent_1	z_1
	m_1
	z_1
Agent_2	z_6
	m_4
	z_6
Joint Reward	r_1

Centralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Joint Experience Replay Trajectories (Mac-JERTs)

- ➢ Collect the joint macro-action-observation experiences of agents;
- ➢ Joint transition tuple is defined as $\langle \vec{z}, \vec{m}, \vec{z}', \vec{r}^c \rangle$, where $\vec{r}^c = \sum_{t=t_{\vec{m}}}^{\vec{\tau}} r_t$

Example:

Time		t=1	t=2
Agent_1		z_1	z_1
		m_1	m_1
		z_1	z_1
Agent_2		z_6	z_6
		m_4	m_4
		z_6	z_7
Joint Reward		r_1	$\sum_{i=1}^{2} r_i$

Centralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Joint Experience Replay Trajectories (Mac-JERTs)

- Collect the joint macro-action-observation experiences of agents;
- Joint transition tuple is defined as $\langle \vec{z}, \vec{m}, \vec{z}', \vec{r}^c \rangle$, where $\vec{r}^c = \sum_{t=t_{\vec{m}}}^{\vec{\tau}} r_t$

Example:

Time		t=1	t=2	t=3
Agent_1		z_1	z_1	z_1
		m_1	m_1	m_1
		z_1	z_1	z_1
Agent_2		z_6	z_6	z_7
		m_4	m_4	m_5
		z_6	z_7	z_7
Joint Reward		r_1	$\sum_{i=1}^{2} r_i$	r_3

Centralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Joint Experience Replay Trajectories (Mac-JERTs)

- ➢ Collect the joint macro-action-observation experiences of agents;
- ➢ Joint transition tuple is defined as $\langle \vec{z}, \vec{m}, \vec{z}', \vec{r}^c \rangle$, where $\vec{r}^c = \sum_{t=t_{\vec{m}}}^{\vec{\tau}} r_t$

Example:

Time	t=1	t=2	t=3	t=4
Agent_1	z_1	z_1	z_1	z_1
	m_1	m_1	m_1	m_1
	z_1	z_1	z_1	z_2
Agent_2	z_6	z_6	z_7	z_7
	m_4	m_4	m_5	m_5
	z_6	z_7	z_7	z_8
Joint Reward	r_1	$\sum_{i=1}^{2} r_i$	r_3	$\sum_{i=3}^{4} r_i$

Centralized learning

Xiao, Hoffman and Amato - CoRL 19

Macro-Action Joint Experience Replay Trajectories (Mac-JERTs)

- ➢ Collect the joint macro-action-observation experiences of agents;
- ➢ Joint transition tuple is defined as $\langle \vec{z}, \vec{m}, \vec{z}\,', \vec{r}^{\,c} \rangle$, where $\vec{r}^{\,c} = \sum_{t=t_{\vec{m}}}^{\vec{\tau}} r_t$

Example:

Time		t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1		z_1	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3
		m_1	m_1	m_1	m_1	m_2	m_2	m_2	m_3	m_3	m_3
		z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3	z_3
Agent_2		z_6	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
		m_4	m_4	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
		z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
Joint Reward		r_1	$\sum_{i=1}^{2} r_i$	r_3	$\sum_{i=3}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	r_8	r_9	$\sum_{i=9}^{10} r_i$

Northeastern University

Centralized learning

Using Mac-JERTs to learn a joint macro-action-value function

Xiao, Hoffman and Amato - CoRL 19

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_1	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3	z_3
Agent_2	z_6	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_4	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
Joint Reward	r_1	$\sum_{i=1}^{2} r_i$	r_3	$\sum_{i=3}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	r_8	r_9	$\sum_{i=9}^{10} r_i$

Identify when ***any*** agent's macro-action terminates

Northeastern University

Centralized learning

Using Mac-JERTs to learn a joint macro-action-value function

Xiao, Hoffman and Amato - CoRL 19

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_1	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3	z_3
Agent_2	z_6	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_4	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
Joint Reward	r_1	$\sum_{i=1}^{2} r_i$	r_3	$\sum_{i=3}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	r_8	r_9	$\sum_{i=9}^{10} r_i$

Sample a joint trajectory

squeeze →

Agent_1	z_1	z_2	z_2	z_3
	m_1	m_2	m_2	m_3
	z_2	z_2	z_3	z_3
Agent_2	z_7	z_8	z_9	z_9
	m_5	m_6	m_7	m_7
	z_8	z_9	z_9	z_{11}
Joint Reward	$\sum_{i=3}^{4} r_i$	r_5	$\sum_{i=6}^{7} r_i$	r_8

Remove the time info

Centralized learning

Using Mac-JERTs to learn a joint macro-action-value function

Xiao, Hoffman and Amato - CoRL 19

Time	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Agent_1	z_1	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3
	m_1	m_1	m_1	m_1	m_2	m_2	m_2	m_3	m_3	m_3
	z_1	z_1	z_1	z_2	z_2	z_2	z_3	z_3	z_3	z_3
Agent_2	z_6	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}
	m_4	m_4	m_5	m_5	m_6	m_7	m_7	m_7	m_8	m_8
	z_6	z_7	z_7	z_8	z_9	z_9	z_9	z_{11}	z_{11}	z_{12}
Joint Reward	r_1	$\sum_{i=1}^{2} r_i$	r_3	$\sum_{i=3}^{4} r_i$	r_5	r_6	$\sum_{i=6}^{7} r_i$	r_8	r_9	$\sum_{i=9}^{10} r_i$

squeeze →

Agent_1	z_1	z_2	z_2	z_3
	m_1	m_2	m_2	m_3
	z_2	z_2	z_3	z_3
Agent_2	z_7	z_8	z_9	z_9
	m_5	m_6	m_7	m_7
	z_8	z_9	z_9	z_{11}
Joint Reward	$\sum_{i=3}^{4} r_i$	r_5	$\sum_{i=6}^{7} r_i$	r_8

We apply Double-DRQN to learn the **joint macro-action-value function** $Q_\phi(\vec{h}, \vec{m})$ (referred to as **Cen-DDRQN**) using **squeezed joint sequential experiences**, by minimizing the loss:

$$\mathcal{L}(\phi) = \mathbb{E}_{<\vec{z},\vec{m},\vec{z}\,',\vec{r}\,^c>\sim\mathcal{D}}\Big[\big(y - Q_\phi(\vec{h},\vec{m})\big)^2\Big], \text{ where } y = \vec{r}\,^c + \gamma Q_{\phi^-}\big(\vec{h}\,', \arg\max_{\vec{m}'} Q_\phi(\vec{h}\,',\vec{m}')\big)$$

Considering the asynchronous macro-action executions over agents, we propose ***conditional target-value prediction:***

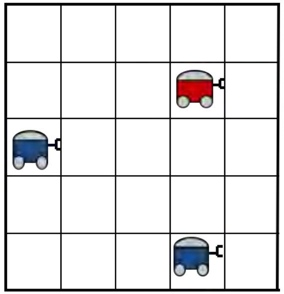
$$y = \vec{r}\,^c + \gamma Q_{\phi^-}\big(\vec{h}\,', \arg\max_{\vec{m}'} Q_\phi(\vec{h}\,', \vec{m}' \mid \vec{m}^{\text{undone}})\big)$$

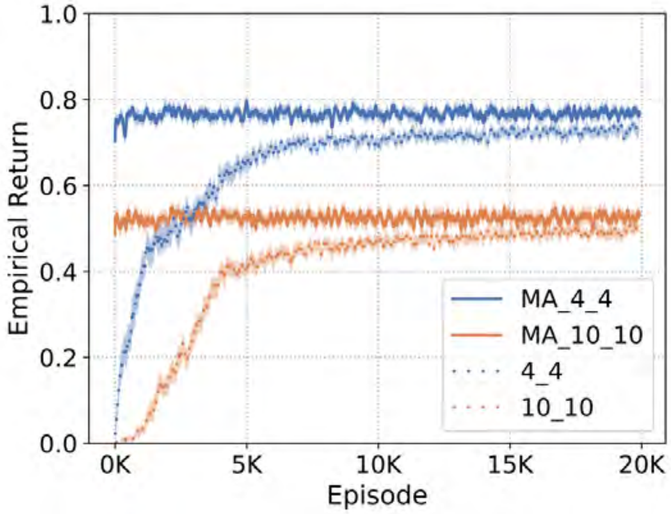
where, $\vec{m}^{\text{undone}}$ is the joint macro-action over the agents who **have not terminated** their macro-actions

Northeastern University

Results: Target capture

Xiao, Hoffman and Amato - CoRL 19





Primitive-observations: *agent's location* (fully observable), *target's location* (partially observable with a flickering probability 0.3)
Macro-observations: same as the primitive

Primitive-actions: *up*, *down*, *left*, *right*, and *stay*;
Macro-actions: ***Move-to-Target*** (terminates when the agent reaches the latest observed target's position) and ***Stay***

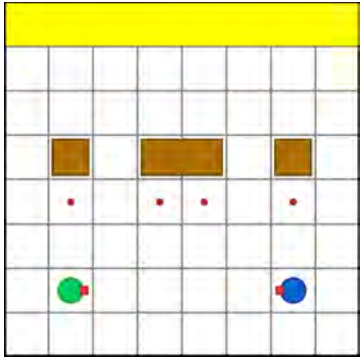
Our decentralized macro-action approach (MA) vs primitive-action version for various grid sizes

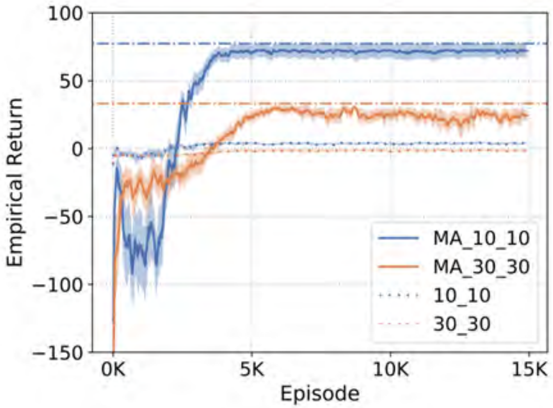
The macro-action method can learn much faster and converge to the same solution in this simple problem

Northeastern University

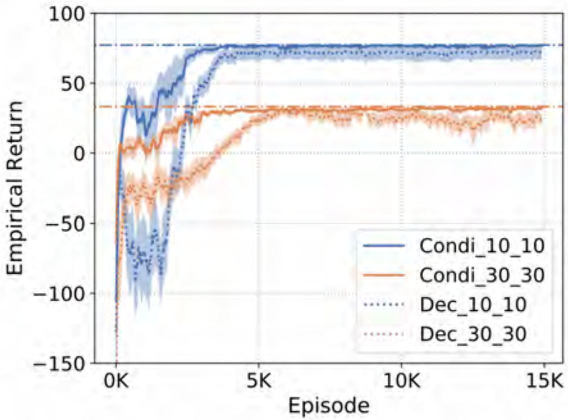
Results: Box pushing

Xiao, Hoffman and Amato - CoRL 19





Decentralized learning with macro-actions (MA) vs primitive-actions

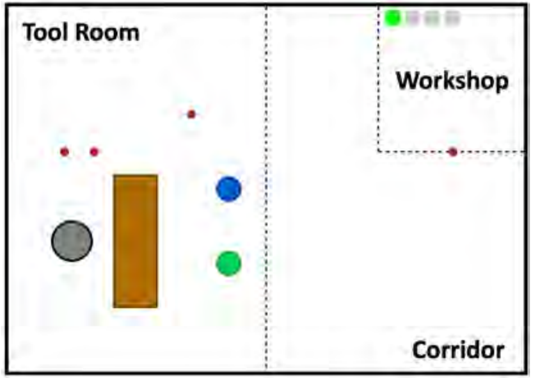


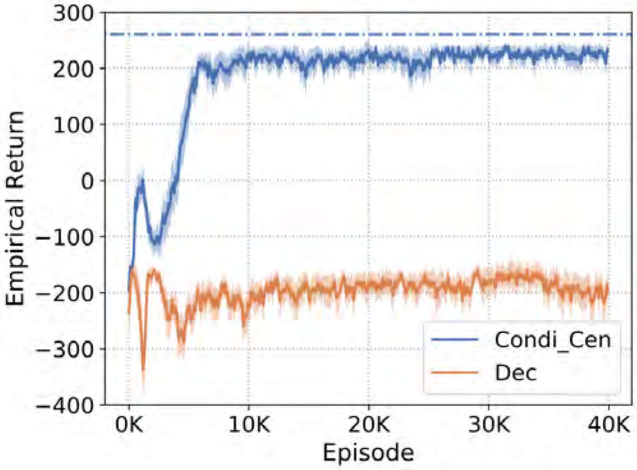
Conditional centralized learning vs decentralized learning with macro-actions

The primitive method can't learn well in this problem, while the decentralized and centralized methods perform well

Results: Warehouse tool delivery

Xiao, Hoffman and Amato - CoRL 19





conditional centralized learning vs decentralized learning

The centralized learner achieves near-optimal performance

The decentralized learner performs poorly due to the difficulty of learning from only local experiences

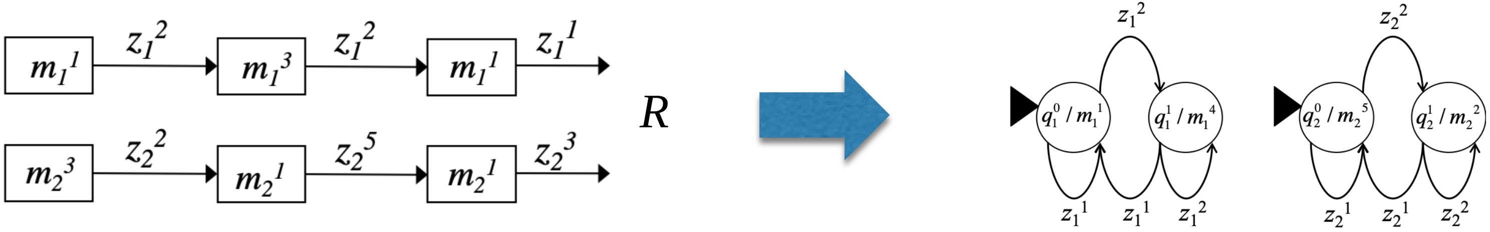
Warehouse robot results

Xiao, Hoffman, Xia and Amato – under submission

Learning controllers

Liu, Amato, Anesta, Griffith and How - AAAI 16

- Want to learn solutions directly from a limited number of demonstrations
- Demonstration trajectories create possible controllers which are optimized to produce a high-valued set of finite-state controllers
- Scalable to large state, macro-action and observation sets

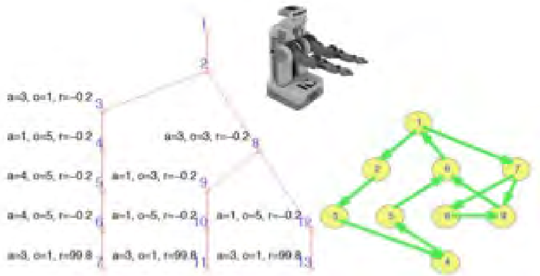


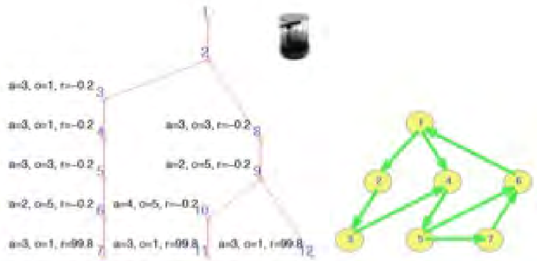
Northeastern University

Learning controllers

Liu, Amato, Anesta, Griffith and How - AAAI 16

- Trajectories give possible sequences and values
- Since don't have model use Monte Carlo-based EM to optimize controller parameters
- Return a controller with parameters learned from the data



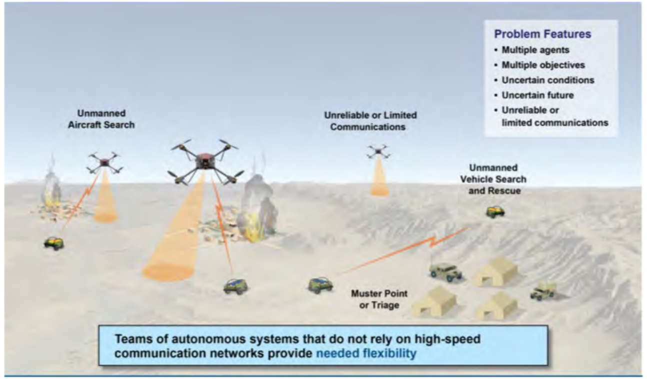


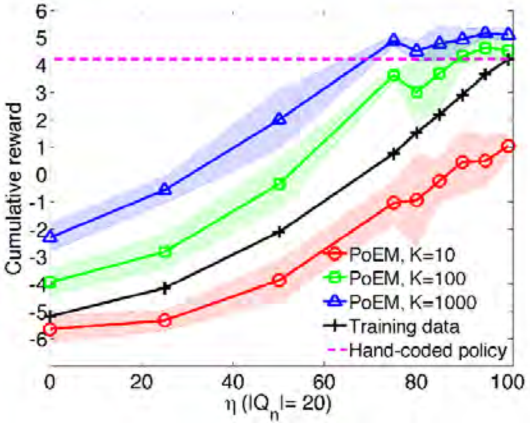
Northeastern University

Search and rescue in simulation

Liu, Amato, Anesta, Griffith and How - AAAI 16

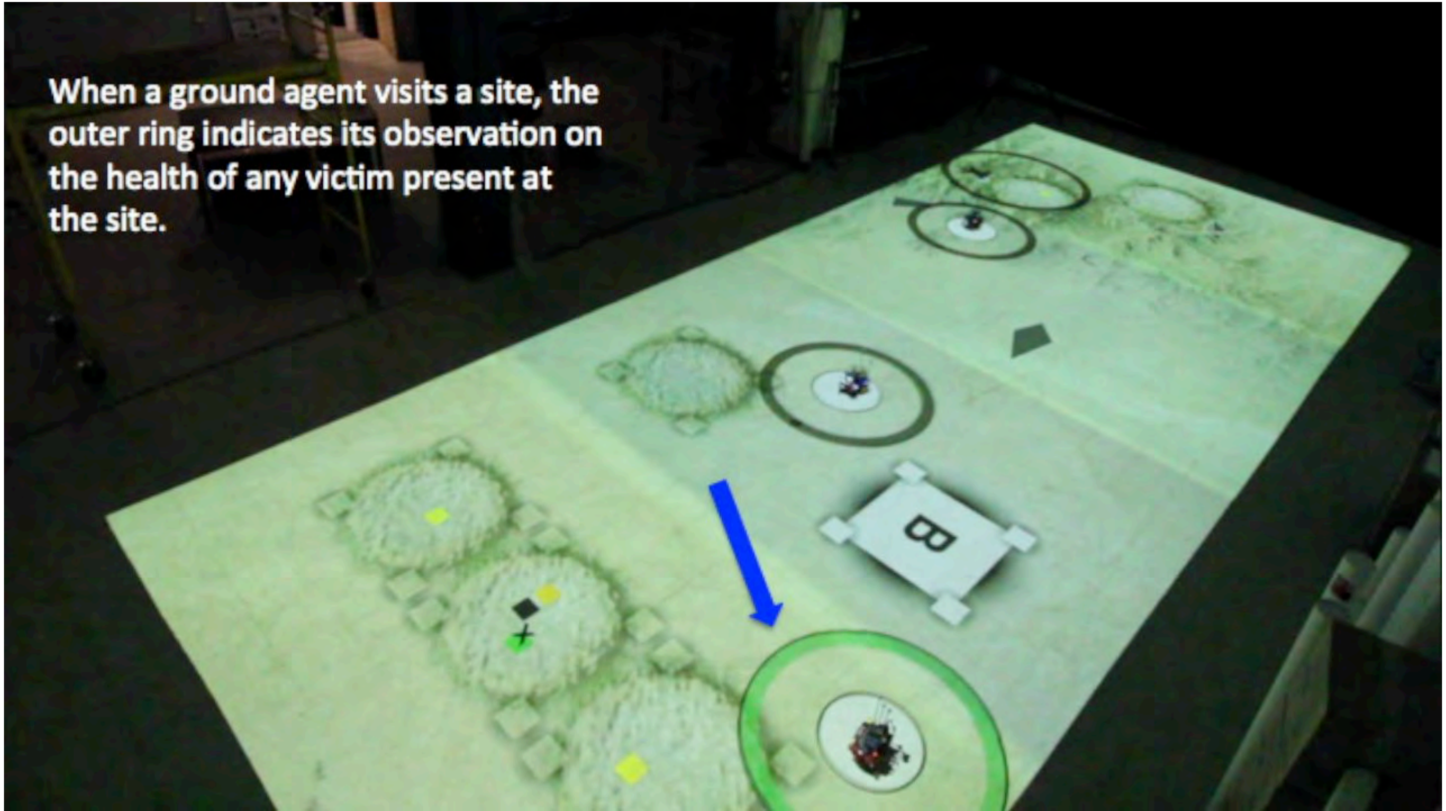
- Used demonstrations from a hand coded 'expert' solution (from MIT-Lincoln lab)
- Tested in a simulator
- Outperforms hand coded 'expert' solutions, even with a small amount of noisy data
- Can also learn optimal or near optimal policies in benchmarks





Search and rescue in hardware

Liu, Sivakumar, Omidshafiei, Amato and How - IROS 17



Why can't we just use deep RL?

- Using deep RL for Dec-POMDPs has become a hot topic (e.g., Omidshafiei, Pazis, Amato, How and Vian, ICML 17, Foerster, Assael, de Freitas, and Whiteson, NIPS 16, Gupta, Egorov, Kochenderfer ICML 17, and many others)
- Helps scale to large state/observation spaces, but doesn't solve other multi-agent learning problems
  - Centralized vs. decentralized learning
  - Sample efficiency/online learning
  - Dealing with nonstationarity
  - Dealing with partial observability

Conclusions

- Dec-POMDPs represent a powerful probabilistic multi-agent framework
  - Considers outcome, sensor and communication uncertainty in a single framework
  - Can model any multi-agent coordination problem
- Macro-actions provide an abstraction to improve scalability
- Learning methods can remove the need to generate a detailed multi-agent model
- Methods also apply when less uncertainty
- Begun demonstrating scalability and quality in a number of domains, but a lot of great open questions to solve!

Northeastern University