

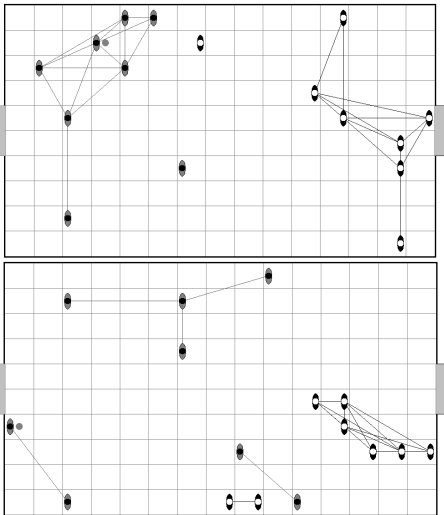
# Distributed Coordination Guidance in Multi-Agent Reinforcement Learning

Qiangfeng Peter Lau, Mong Li Lee and Wynne Hsu  
Dept. of Computer Science  
National University of Singapore

November 8, 2011

# Collaborative Multi-Agent Domains

## Dynamic Communication



- Agents work as a team towards a common global goal
- Examples:
  - Computers on grid network managing jobs
  - Group of soccer robots
  - Combat Simulator (like a Real-time Strategy Game)

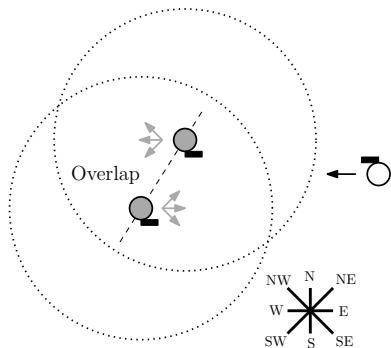


## Difficulties in Domain

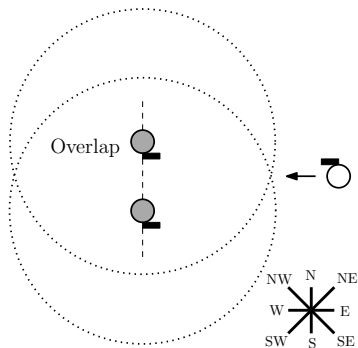
- Very large joint state-action space to explore
- Naive reinforcement learning (learning from interactions) impractical
- For online learning to be practical, quantity of interactions for exploration must be optimized
- Overcome the above by using coordination knowledge to guide exploration

# Coordination Constraints

RTS example

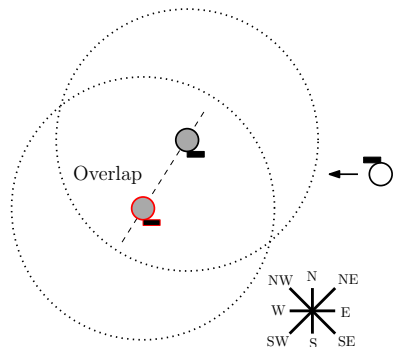


Bad Alignment



Good Alignment

## Dynamic Coordination Constraints



Not a bad alignment if one marine is wounded

- CCs may not be useful in every state
- Determining when to use CCs is part of the learning process
- RL system learns to guide itself

## Coordination in Existing Work and This Work

Existing work:

- Usually refers to coordinated action selection, i.e., which agents should select actions jointly
- Express as features involving two or more agents for use in value function approximation
- Used only as static rules, e.g., collision avoidance

In this work,

- Coordination constraints are additional high level actions that agents can use to guide exploration
- A two level system that allows coordination constraints to be dynamically utilized

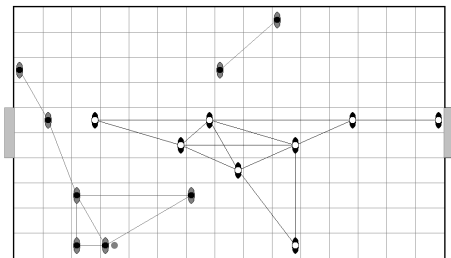
## Challenges

1. Communication between Agents' changes over time, agents may be removed, system must be distributed
  - no critical components in any one agent
2. Additional execution and exploration requirements of the top level must have positive impact to the overall learning rate
3. Learning at the top level should not detract from solving the original goal

# Decentralized Markov Decision Process (DEC-MDP)

## Basic Problem Formulation

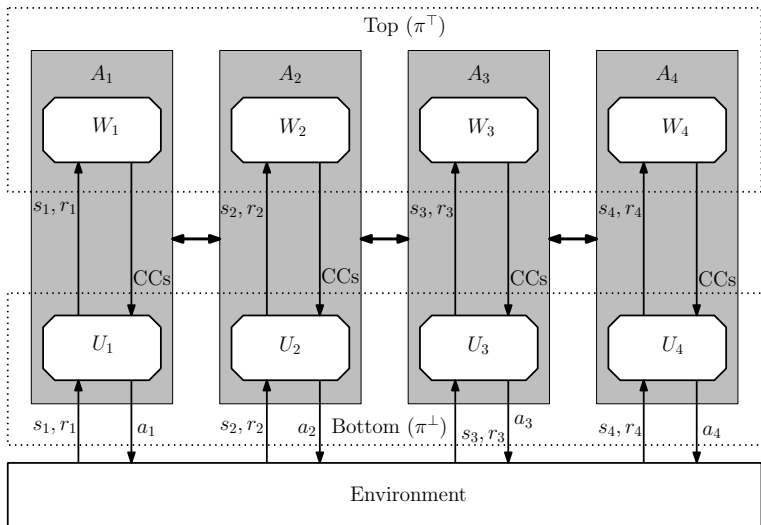
- Factored State Space,  $\mathcal{S} = \mathcal{S}_0 \times \mathcal{S}_1 \times \dots \times \mathcal{S}_N$
- Factored Action Space,  $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$
- Global Transition Probability,  $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$
- Agent decomposed rewards,  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$  and  $\mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \sum_{n=1}^N \mathcal{R}_n(\mathbf{s}, \mathbf{a}, \mathbf{s}')$
- Agents may only communicate with their neighbours defined by  $\mathbf{s} \in \mathcal{S}$



## Adding the top level for model-free RL

- Augment the DEC-MDP with extra top level actions to *(de)activate* CCs
- At every time step, pick a combination of CCs to activate, followed by primitive action constrained by the CCs
- With  $K$  CCs, top level action space is  $\mathcal{A}^0 = A_1^0 \times \dots \times A_K^0$ , a top level action  $\mathbf{b} \in \mathcal{A}^0$  is vector of zero/one values
- Global top action value function  $W(\mathbf{s}, \mathbf{b})$ , and bottom action value function  $U(\mathbf{s}, \mathbf{a})$
- $\pi^\top(\mathbf{s}) = \operatorname{argmax}_{\mathbf{b} \in \mathcal{A}^0(\mathbf{s})} W(\mathbf{s}, \mathbf{b})$ , and  
 $\pi^\perp(\mathbf{b}, \mathbf{s}) = \operatorname{argmax}_{\mathbf{a} \in \mathcal{A}(\mathbf{b}, \mathbf{s})} U(\mathbf{s}, \mathbf{a})$

# Example Overview of System for 4 Agents



## Local Value Functions & Updating

- Distribute global functions into local parts:

$$W(\mathbf{s}, \mathbf{b}) = \sum_{n=1}^N W_n(\mathbf{s}_n, \mathbf{b}_n) \text{ and } U(\mathbf{s}, \mathbf{a}) = \sum_{n=1}^N U_n(\mathbf{s}_n, \mathbf{a}_n)$$

- Linear Function Approximation:

$$W_n(\mathbf{s}_n, \mathbf{b}_n) \approx \vec{w}_{W_n} \cdot \vec{\phi}_{W, \mathbf{s}_n, \mathbf{b}_n} \text{ and } U_n(\mathbf{s}_n, \mathbf{a}_n) \approx \vec{w}_{U_n} \cdot \vec{\phi}_{U, \mathbf{s}_n, \mathbf{a}_n}$$

- Local Update Equations:

$$\vec{w}_{W_n} \leftarrow \vec{w}_{W_n} + \alpha[r_n + \gamma W_n(\mathbf{s}'_n, \mathbf{b}'_n) - W_n(\mathbf{s}_n, \mathbf{b}_n)] \vec{\phi}_{W, \mathbf{s}_n, \mathbf{b}_n}$$

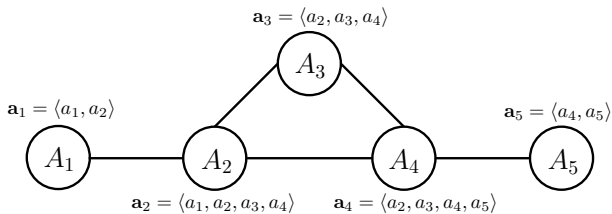
$$\vec{w}_{U_n} \leftarrow \vec{w}_{U_n} + \alpha[r_n + \gamma U_n(\mathbf{s}'_n, \mathbf{a}'_n) - U_n(\mathbf{s}_n, \mathbf{a}_n)] \vec{\phi}_{U, \mathbf{s}_n, \mathbf{a}_n}$$

- Basis functions return zero if agents involved in them have no communication
  - pairwise basis functions' weights only updated for duration that agents are neighbours

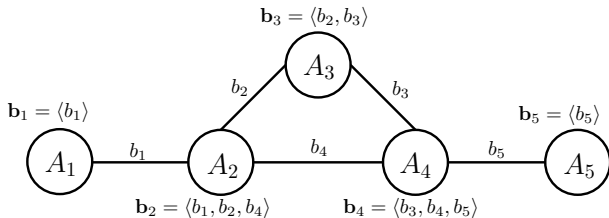
# Accessible Variables

Limited by Communication

Bottom level:



Top level:



## Basis Functions & Constraints

Predicate basis functions (PBF) return value in  $\{0, 1\}$ , e.g.,

$$\begin{aligned} \text{NotAligned}(\mathbf{s}_1, \mathbf{s}_2, a_1, a_2) &:= \text{SameNearestEnemy}(\mathbf{s}_1, \mathbf{s}_2) \\ &\wedge \text{Distance}\Delta(\mathbf{s}_1, a_1, \mathbf{s}_2, a_2) > 0 \end{aligned}$$

They can also be used as CCs, i.e.,  $c_k(\mathbf{s}, \mathbf{a}) = -\infty \cdot \phi_k(\mathbf{s}, \mathbf{a})$

Primitive action selection, a distributed COP with  $OBJ_{\mathbf{b}}$ ,

$$\sum_{n=1}^N \left[ \underbrace{\sum_j w_j \cdot \phi_j(\mathbf{s}_n, \mathbf{a}_n)}_{\text{Normal BFs \& deactivated CCs}} + \underbrace{\sum_{k \in \mathbf{b}_n} -\infty \cdot \phi_k(\mathbf{s}_n, \mathbf{a}_n)}_{\text{CCs activated in } \mathbf{b}_n} \right]$$

$U_n$ , the local function of agent  $n$

## Top Level BFs and Efficiency

- Predicates for bottom level can be reused for top level PBFs:  
 $\phi_{W1} : Wounded(s_1) \wedge Activated(NotAligned_{1,2})$   
 $\phi_{W2} : Wounded(s_2) \wedge Activated(NotAligned_{1,2})$
- Efficient top level exploration by inspecting CCs for deactivation:

$$NotAligned(\mathbf{s}_1, \mathbf{s}_2, a_1, a_2) := SameNearestEnemy(\mathbf{s}_1, \mathbf{s}_2) \\ \wedge Distance\Delta(\mathbf{s}_1, a_1, \mathbf{s}_2, a_2) > 0$$

- Deactivate CCs for agents that have no communication

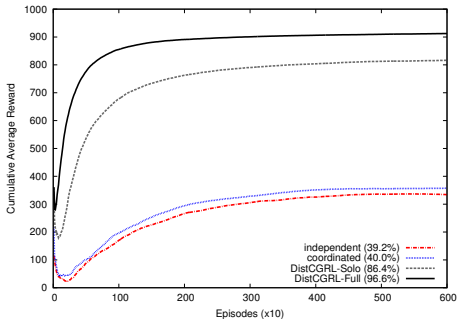
# Tactical RTS Game

## Setup

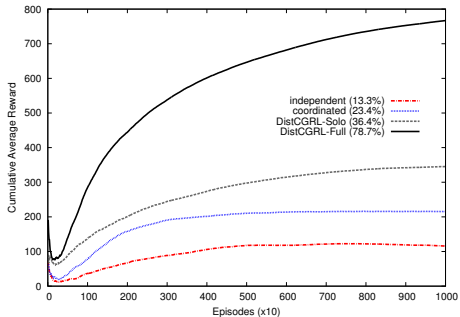
- 240 by 240 point based map, communication limited to 30 points
- 10 versus 10 marines
- Each marine has 8 compass direction movement actions, attack actions, idle
- $\epsilon$ -greedy policies used with constant step size
- 2 opponents: unpredictable and aggressive
- 4 RL players: independent, coordinated, DistCGRL-Solo, DistCGRL-Full
- CCs include *NotAligned* and others related to troop formation like protecting wounded team members

# Tactical RTS Game

## Result



Unpredictable



Aggressive

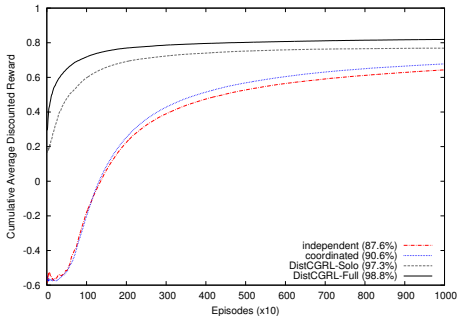
# Simplified Soccer Game

## Setup

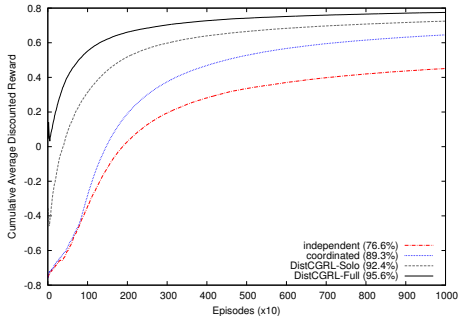
- Grid World of 15 by 10, communication limited to Manhattan distance of 5
- 8 versus 8 soccer players
- Each soccer player has 4 movement actions, pass, shoot, idle
- 2 opponents: defensive and aggressive
- decreasing step size and  $\epsilon$  over time
- CCs relating to passing to players moving to desirable positions, blocking opponents movements together, etc

# Simplified Soccer Game

## Result



Defensive



Aggressive

## Conclusion

- Presented a two level model-free RL system, for fine-grained learning of CCs to guide exploration
- CCs are related to declarative predicates used for BFs
- CCs involving coordination among multiple agents lead to better performance than those for single agents
- System is distributed, no critical components in any one agent
- Future work:
  - automatically creating CCs
  - fusing CCs with task-based methods
  - extending DistCGRL to domains with more communication restrictions, e.g. Markov games

## Q&A

Thank you for listening.