#### Multilayered Skill Learning and Movement Coordination for Autonomous Robotic Agents

#### Patrick MacAlpine

Department of Computer Science, The University of Texas at Austin

July 21, 2017



#### **Robots Becoming More Prevalent**

- Robots have increasing capabilities and are becoming cheaper to manufacture
- "About 1.2 million additional advanced robots are expected to be deployed in the U.S. by 2025" - Boston Consulting Group



#### **Need to Learn Skills**

- Desire robots to be autonomous—with more robots less likely to have a person trained to directly control them
- Have robots perform complex tasks consisting of multiple skills



(video from DARPA)

#### **Need Robots to Work Together**

- Have multiple robots in the same environment
- Want robots to coordinate and work together



# Video

(video from https://youtu.be/Ew\_Ih779FwY)

### Thesis Question(s)

Combining a desire for robots to be able to act autonomously and coordinate their movement ...

#### **Skill Learning**

How to use machine learning to optimize multiple skills or behaviors for mobile robots that work well together (such as walking and pushing an object)?

#### **Movement Coordination**

Once we have optimized these skills, how can we get robots to coordinate their movement and complete tasks, ideally in as little time as possible, without running into each other?

# Outline

RoboCup 3D Simulation Domain Description

Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work

# Outline

RoboCup 3D Simulation Domain Description

Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work

#### RoboCup 3D Simulation Domain

- Teams of 11 vs 11 autonomous robots play soccer
- Realistic physics using Open Dynamics Engine (ODE)
- Simulated robots modeled after Nao robot
- Robot receives noisy visual information about environment
- Robots can communicate with each other over limited bandwidth channel





#### Example of RoboCup 3D Simulation Game



RoboCup 2016 Final

#### UT Austin Villa RoboCup 3D Simulation Team 2007-2009



No games won; no goals scored

#### UT Austin Villa RoboCup 3D Simulation Team 2010



Record of 4 wins, 6 losses, 1 tie; 11 goals scored and 17 conceded

RoboCup	2010	2011-2016
Goals For:	11	
Goals Against:	17	
Record (W-L-T):	4-6-1	
Place:	Outside Top-8	

RoboCup	2010	2011-2016
Goals For:	11	469
Goals Against:	17	
Record (W-L-T):	4-6-1	
Place:	Outside Top-8	

RoboCup	2010	2011-2016
Goals For:	11	469
Goals Against:	17	7
Record (W-L-T):	4-6-1	
Place:	Outside Top-8	

RoboCup	2010	2011-2016
Goals For:	11	469
Goals Against:	17	7
Record (W-L-T):	4-6-1	101-3-6
Place:	Outside Top-8	

RoboCup	2010	2011-2016
Goals For:	11	469
Goals Against:	17	7
Record (W-L-T):	4-6-1	101-3-6
Place:	Outside Top-8	1st 5X, 2nd 1X

# **BIG IMPROVEMENT!**

\* RoboCup (2011-2016) \* Titles: 5 championships, 1 second place Record (W - L - T): 101 - 3 - 6 Goals (For - Against): 469 - 7

# Outline

RoboCup 3D Simulation Domain Description

• Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work

# Outline

RoboCup 3D Simulation Domain Description

• Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work

# **Overlapping Layered Learning**

UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning Patrick MacAlpine, Mike Depinet, and Peter Stone AAAI 2015

#### **Skills Do Not Always Work Together**



# Hard to get kick skill to work well with walk skill

### **Reinforcement Learning Direct Policy Search**



- Learn a parameterized policy that determine an agent's behavior (what actions an agent chooses based on state of environment)
- Optimization algorithm produces candidate parameters to evaluate on optimization task
- Optimization task evaluates parameters and returns fitness to optimization algorithm

# CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy)



(image from wikipedia)

- Evolutionary numerical optimization method
- Candidates sampled from multidimensional Gaussian and evaluated for their fitness
- Weighted average of members with highest fitness used to update mean of distribution
- Covariance update using evolution paths controls search step sizes

Hansen, N., S.D. Müller and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES), 2003.

#### Layered Learning:

Hierarchical machine learning paradigm that enables learning of complex behaviors by incrementally learning a series of sub-behaviors. Higher layers directly depend on the learned lower layers.



Sequential Layered Learning (SLL) Concurrent Layered Learning (CLL)

Δ

### **DESCRIPTIONS:**

**Sequential Layered Learning:** Freeze parameters of layer after learning before learning of the next layer (P. Stone, 2000)

**Concurrent Layered Learning:** Keep parameters of layer open during learning of the next layer (S. Whiteson and P. Stone, 2003)

#### Layered Learning:

Hierarchical machine learning paradigm that enables learning of complex behaviors by incrementally learning a series of sub-behaviors. Higher layers directly depend on the learned lower layers.



Sequential Layered Learning (SLL) Concurrent Layered Learning (CLL)

Α

PROBLEMS:

**Sequential Layered Learning:** Can be too limiting in the joint behavior policy seach space

**Concurrent Layered Learning:** The increased dimensionality can make learning harder or intractable

#### Layered Learning:

Hierarchical machine learning paradigm that enables learning of complex behaviors by incrementally learning a series of sub-behaviors. Higher layers directly depend on the learned lower layers.



Sequential Layered Learning (SLL) Concurrent Layered Learning (CLL)

Δ

SOLUTION:

**Overlapping Layered Learning:** Tradeoff between freezing or keeping open previous learned behaviors

Optimizes "seam" or overlap between behaviors: keeps some parts of previously learned layers open during subsequent learning

Layered learning both in series and parallel, in which some but not necessairly all, parts of previously learned layers are left open during learning of subsequent layers.



MacAlpine P., Depinet M., and Stone P. UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning, AAAI 2015.

Layered learning both in series and parallel, in which some but not necessairly all, parts of previously learned layers are left open during learning of subsequent layers.



**Combining Independently Learned Behaviors:** Two or more behaviors learned independently and then combined by relearning subset of behaviors' parameters

Layered learning both in series and parallel, in which some but not necessairly all, parts of previously learned layers are left open during learning of subsequent layers.



**Combining Independently Learned Behaviors:** Two or more behaviors learned independently and then combined by relearning subset of behaviors' parameters **Partial Concurrent Layered Learning:** Only part, but not all, of a previously learned layer's behaviors are left open

Layered learning both in series and parallel, in which some but not necessairly all, parts of previously learned layers are left open during learning of subsequent layers.



Combining Independently Learned Behaviors: Two or more behaviors learned independently and then combined by relearning subset of behaviors' parameters Partial Concurrent Layered Learning: Only part, but not all, of a previously learned layer's behaviors are left open Previous Learned Layer Refinement: After a layer is learned and frozen, and then a subsequent layer is learned, part of all of the previous layer is unfrozen

PhD Defense

# Dribbling and Kicking the Ball in the Goal

- Learn different walk parameter sets for different subtasks
  - Going to a target
  - Sprinting forward (+/- 15° of current heading)
  - Positioning around the ball when dribbling
  - Approaching the ball to kick it



• Learn fixed kick



# Combine kick with walk through overlapping behavior layer



#### **Initial Walk Parameters**

- 25 different parameters
- Designed and hand-tuned to work on the actual Nao robot
- Provides a slow and stable walk



# Sequential Layered Learning of Walk Behaviors



# Video

Red 'T' = GoToTarget parameters, yellow 'S' = Sprint parameters

- Optimizing parameters for omnidirectional walk engine (step height, frequency, balance, etc.)
- Agent rewarded for distance traveled toward magenta target
- First GoToTarget layer optimized and frozen, then Sprint layer learned through sequential layered learning





#### Without Layered Learning



# Video

Attempt to transition between *Dribble* walk parameters (red 'D') *Fast* walk parameters (yellow 'F')

 Unstable when not using layered learning to learn transition between walks

Patrick MacAlpine (UT Austin)

# Walk\_PositionToDribble Optimization



# Video

 $\frac{\text{Red 'T'} = GoToTarget}{\text{cyan 'P'} = Positioning \text{ parameters}}$ 

- Dribble ball toward goal for 15 seconds from multiple starting points around ball
- Reward for distance ball dribbled toward goal



### **Kick Learning**



- Kicks represented as series of fixed poses
- Poses are parameterized by joint angles

# Kick\_Long\_Primitive Optimization



# Video

- Optimize joint positions that make up a series of fixed frame poses for executing kicking motion
- Kick ball from fixed standing position
- Reward for kick distance and accuracy

Kick\_Long\_Primitive (73)
#### Kick\_Long\_Behavior Optimization



# Video

- Approach ball and kick it
- Reward for kick distance and accuracy
- Relearning overlap kick parameters for positioning and stability with walk (combining independently learned behaviors)







Learning the Kick\_Fast\_Behavior

Concurrent Layered Learning struggles learning kick and approach at same time



Learning the Kick\_Fast\_Behavior

- Concurrent Layered Learning struggles learning kick and approach at same time
- Sequential Layered Learning difficulty learning kick in presence of walk approach



Learning the Kick\_Fast\_Behavior

- Concurrent Layered Learning struggles learning kick and approach at same time
- Sequential Layered Learning difficulty learning kick in presence of walk approach
- Overlapping Layered Learning (CILB), where walk approach and kick are learned independently in isolation and then combined, performs the best

Patrick MacAlpine (UT Austin)

#### **Dribbling and Kicking the Ball**



Red 'T' = GoToTarget parameters, yellow 'S' = Sprint parameters, cyan 'P' = Positioning parameters, orange 'A' = Approach parameters

#### Learned Layers (2014)



- 19 learned behaviors for standing up, walking, and kicking (more than 3X behaviors of previous layered learning systems)
  - ► CILB, PCLL, PLLR
- Over 500 parameters optimized during the course of learning using CMA-ES algorithm
  - frozen, open

#### Learned Layers (2014)



#### Computation

 $\approx$  700k parameter sets evaluated

pprox 1.5 years compute time (pprox 5 days on distributed computing cluster)

#### Learned Layers (2014)



#### Entering 2017 Competition

25 new kicks added since 2014 (variable distance, directional, fast walk) 69 total learned behaviors

Patrick MacAlpine	(UT Austin)
-------------------	-------------

#### **Backwards Kick**



RoboCup 2016 Game Action

#### Scoring on a Kickoff



# Video

- Kickoffs are indirect (can't score with a single kick)
- Learn touch and fixed kick behaviors independently
- Combining touch and kick by relearning positioning parameters (combining independently learned behaviors) and also learning new timing parameter (partial concurrent layered learning)



Patrick MacAlpine (UT Austin)

#### **Kickoff Fail**



Robots interfer with each other when trying to learn a kick with a touch

- Introduced three paradigms for Overlapping Layered Learning
  - Combining Independently Learned Behaviors, Partial Concurrent Layered Learning, Previous Learned Layer Refinement
- Showed effectiveness of Overlapping Layered Learning for learning complex behaviors in the RoboCup 3D simulation domain
  - Learned 19 behaviors while optimizing over 500 parameters
- Demonstrated generality of Overlapping Layered Learning to multiple robot models
  - Able to successfully learn behaviors for 5 different robot types
- Overlapping Layered Learning is a paradigm, not an algorithm
  - Automating the selection of overlapping parameters and behaviors is future work

- Introduced three paradigms for Overlapping Layered Learning
  - Combining Independently Learned Behaviors, Partial Concurrent Layered Learning, Previous Learned Layer Refinement
- Showed effectiveness of Overlapping Layered Learning for learning complex behaviors in the RoboCup 3D simulation domain
  - Learned 19 behaviors while optimizing over 500 parameters
- Demonstrated generality of Overlapping Layered Learning to multiple robot models
  - Able to successfully learn behaviors for 5 different robot types
- Overlapping Layered Learning is a paradigm, not an algorithm
  - Automating the selection of overlapping parameters and behaviors is future work

- Introduced three paradigms for Overlapping Layered Learning
  - Combining Independently Learned Behaviors, Partial Concurrent Layered Learning, Previous Learned Layer Refinement
- Showed effectiveness of Overlapping Layered Learning for learning complex behaviors in the RoboCup 3D simulation domain
  - Learned 19 behaviors while optimizing over 500 parameters
- Demonstrated generality of Overlapping Layered Learning to multiple robot models
  - Able to successfully learn behaviors for 5 different robot types
- Overlapping Layered Learning is a paradigm, not an algorithm
  - Automating the selection of overlapping parameters and behaviors is future work

- Introduced three paradigms for Overlapping Layered Learning
  - Combining Independently Learned Behaviors, Partial Concurrent Layered Learning, Previous Learned Layer Refinement
- Showed effectiveness of Overlapping Layered Learning for learning complex behaviors in the RoboCup 3D simulation domain
  - Learned 19 behaviors while optimizing over 500 parameters
- Demonstrated generality of Overlapping Layered Learning to multiple robot models
  - Able to successfully learn behaviors for 5 different robot types
- Overlapping Layered Learning is a paradigm, not an algorithm
  - Automating the selection of overlapping parameters and behaviors is future work

- Introduced three paradigms for Overlapping Layered Learning
  - Combining Independently Learned Behaviors, Partial Concurrent Layered Learning, Previous Learned Layer Refinement
- Showed effectiveness of Overlapping Layered Learning for learning complex behaviors in the RoboCup 3D simulation domain
  - Learned 19 behaviors while optimizing over 500 parameters
- Demonstrated generality of Overlapping Layered Learning to multiple robot models
  - Able to successfully learn behaviors for 5 different robot types
- Overlapping Layered Learning is a paradigm, not an algorithm
  - Automating the selection of overlapping parameters and behaviors is future work

- MAXQ algorithm (Dietterich, 2000)
  - Learns at all levels of the heirarchy simultaneously learned subtasks are locally optional not hierarchically optimal
  - Unlike MAXQ layered learning allows for using different learning algorithms at each level of the hierarchy
- Options (Sutton et al., 1999)
  - Learned policies with initiation and termination conditions used to complete subtasks



Options: want to transition from one sub-behavior to another

- MAXQ algorithm (Dietterich, 2000)
  - Learns at all levels of the heirarchy simultaneously learned subtasks are locally optional not hierarchically optimal
  - Unlike MAXQ layered learning allows for using different learning algorithms at each level of the hierarchy
- Options (Sutton et al., 1999)
  - Learned policies with initiation and termination conditions used to complete subtasks



- MAXQ algorithm (Dietterich, 2000)
  - Learns at all levels of the heirarchy simultaneously learned subtasks are locally optional not hierarchically optimal
  - Unlike MAXQ layered learning allows for using different learning algorithms at each level of the hierarchy
- Options (Sutton et al., 1999)
  - Learned policies with initiation and termination conditions used to complete subtasks



Overlapping layered learning: combine sub-behaviors into a new behavior

- MAXQ algorithm (Dietterich, 2000)
  - Learns at all levels of the heirarchy simultaneously learned subtasks are locally optional not hierarchically optimal
  - Unlike MAXQ layered learning allows for using different learning algorithms at each level of the hierarchy
- Options (Sutton et al., 1999)
  - Learned policies with initiation and termination conditions used to complete subtasks



Overlapping layered learning: combine sub-behaviors into a new behavior

## Overlapping layered learning—learns efficient behaviors that can transition between and work well with each other

#### Outline

RoboCup 3D Simulation Domain Description

• Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

• Summary and Future Work

#### Outline

RoboCup 3D Simulation Domain Description

Overlapping Layered Learning

• SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work



### SCRAM: Scaleable Collision-avoiding Role Assignment with Minimal-makespan

SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan for Formational Positioning Patrick MacAlpine, Eric Price, and Peter Stone AAAI 2015

#### **Robots Do Not Work Well With Each Other**



#### Robots collide instead of coordinating where they are moving



#### Problem:

How to assign *n* interchangeable robots to *n* targets in a one-to-one mapping so that the makespan is minimized and collisions are avoided.

Makespan = time for all robots to reach their assigned target positions (equivalent to the time for the the robot with the longest distance to travel to reach its assigned target position)



#### Problem:

How to assign *n* interchangeable robots to *n* targets in a one-to-one mapping so that the makespan is minimized and collisions are avoided.

#### **ASSUMPTIONS:**

- No two robots or targets occupy the same position
- Robots are treated as zero width point masses
- Robots move at same constant speed along straight line paths to targets



Required properties of a role assignment function to be *CM Valid* (Collision-avoiding with Minimal-makespan):

- 1. *Minimizing makespan* it minimizes the maximum distance from a robot to target, with respect to all possible mappings
- 2. Avoiding collisions robots do not collide with each other



Required properties of a role assignment function to be *CM Valid* (Collision-avoiding with Minimal-makespan):

- 1. *Minimizing makespan* it minimizes the maximum distance from a robot to target, with respect to all possible mappings
- 2. Avoiding collisions robots do not collide with each other

Desirable but not necessary to be CM Valid:

3. *Dynamically consistent* - role assignments don't change or switch as robots move toward target positions





Bipartite Graph Perfect Matching n! possible mappings

Patrick Mac Al	nine (	(IIT Auet	in)
I utilion muorai		OI Aust	

PhD Defense



Required properties of a role assignment function to be *CM Valid* (Collision-avoiding with Minimal-makespan):

- 1. *Minimizing makespan* it minimizes the maximum distance from a robot to target, with respect to all possible mappings
- 2. Avoiding collisions robots do not collide with each other

Not include  $a_2 \rightarrow p_5$  (longest possible distance), instead  $a_1 \rightarrow p_3$  (minimal longest distance)

 $a_1 \rightarrow p_1$  and  $a_2 \rightarrow p_2$  would cause a collision between  $a_1$  and  $a_2$ 

#### **Motivation**



- Scenarios for which the **bottleneck** is the time it takes for the last robot to get to its target (e.g. robots procuring items for an order to be shipped)
- Tasks requiring robots be synchronized when they start jobs at their target positions (e.g. robots on an assembly line)

### Minimum Maximal Distance Recursive (MMDR) Role Assignment Function



Lowest lexicographical cost (shown with arrows) to highest cost ordering of mappings from agents (A1,A2,A3) to role positions (P1,P2,P3). Each row represents the cost of a single mapping.

- Mapping cost = vector of distances sorted in decreasing order
- Optimal mapping = lexicographically sorted lowest cost mapping

#### **Hungarian Algorithm**



- Finds a maximum/minimum weight (sum of weights) perfect matching in a bipartite graph (solves the *assignment problem*)
- Runs in O(n<sup>3</sup>) time

#### **Hungarian Algorithm**



- Finds a maximum/minimum weight (sum of weights) perfect matching in a bipartite graph (solves the assignment problem)
- Runs in O(n<sup>3</sup>) time

#### Can we transform MMDR into the assignment problem?

#### **Hungarian Algorithm**



- Finds a maximum/minimum weight (sum of weights) perfect matching in a bipartite graph (solves the *assignment problem*)
- Runs in O(n<sup>3</sup>) time

#### Can we transform MMDR into the assignment problem? YES!

- 1. Convert edge distances to ordered bit vector weights
- 2. Run Hungarian algorithm with modified weights returns MMDR Time:  $O(n^2)$  bit vector weights X  $O(n^3)$  Hungarian algorithm =  $O(n^5)$
# **Hungarian Algorithm**



- Finds a maximum/minimum weight (sum of weights) perfect matching in a bipartite graph (solves the *assignment problem*)
- Runs in O(n<sup>3</sup>) time

#### Can we transform MMDR into the assignment problem? YES!

1. Convert edge distances to ordered bit vector weights 2. Run Hungarian algorithm with modified weights returns MMDR Time:  $O(n^2)$  bit vector weights X  $O(n^3)$  Hungarian algorithm =  $O(n^5)$ 

Scales to 100s of robots

# **Hungarian Algorithm**



- Finds a maximum/minimum weight (sum of weights) perfect matching in a bipartite graph (solves the *assignment problem*)
- Runs in O(n<sup>3</sup>) time

### Can we transform MMDR into the assignment problem? YES!

- 1. Convert edge distances to ordered bit vector weights
- 2. Run Hungarian algorithm with modified weights returns MMDR Time:  $O(n^2)$  bit vector weights X  $O(n^3)$  Hungarian algorithm =  $O(n^5)$ Scales to 100s of robots MMDR *is* dynamically consistent

Patrick MacAlpine (UT Austin)

- Find a perfect matching *M* that:
- 1. Has a minimum-maximal edge

2. Minimizes the sum of distances squared

$$M'' := \{ X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty}) \}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_2^2)$$
(2)

Find a perfect matching *M* that:

1. Has a minimum-maximal edge Find minimal-maximum edge with weight *w* using Ford-Fulkerson

2. Minimizes the sum of distances squared

$$M'' := \{X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty})\}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_2^2)$$
(2)

Find a perfect matching *M* that:

Has a minimum-maximal edge
Find minimal-maximum edge with weight w using Ford-Fulkerson
Remove all edges with weight greater than w

2. Minimizes the sum of distances squared

$$M'' := \{X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty})\}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_{2}^{2})$$
(2)

Find a perfect matching *M* that:

1. Has a minimum-maximal edge Find minimal-maximum edge with weight *w* using Ford-Fulkerson Remove all edges with weight greater than *w* 

2. Minimizes the sum of distances squared Run Hungarian Algorithm

$$M'' := \{X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty})\}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_{2}^{2})$$
(2)

Find a perfect matching *M* that:

1. Has a minimum-maximal edge Find minimal-maximum edge with weight *w* using Ford-Fulkerson Remove all edges with weight greater than *w* 

2. Minimizes the sum of distances squared Run Hungarian Algorithm

$$M'' := \{X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty})\}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_{2}^{2})$$
(2)

Time:  $O(n^3)$  Ford-Fulkerson Alg. +  $O(n^3)$  Hung. Alg. =  $O(n^3)$ Scales to 1000s of robots

Find a perfect matching *M* that:

1. Has a minimum-maximal edge Find minimal-maximum edge with weight *w* using Ford-Fulkerson Remove all edges with weight greater than *w* 

2. Minimizes the sum of distances squared Run Hungarian Algorithm

$$M'' := \{X \in \mathbb{M} \mid \|X\|_{\infty} = \min_{M \in \mathbb{M}} (\|M\|_{\infty})\}$$
(1)  
$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}} (\|M\|_2^2)$$
(2)

Time:  $O(n^3)$  Ford-Fulkerson Alg. +  $O(n^3)$  Hung. Alg. =  $O(n^3)$ Scales to 1000s of robots MMD+MSD<sup>2</sup> is not dynamically consistent

#### **Role Assignment Function Properties**

Function	Min. Makespan	No Collisions	Dyn. Consistent	
MMD+MSD <sup>2</sup>	Yes	Yes	No	
MMDR	Yes	Yes	Yes	
MSD <sup>2</sup>	No	Yes	No	
MSD	No	No	No	
Random	No	No	No	
Greedy	No	No	No	

**Function Properties** 

MSD: Minimize sum of distances between robots and targets.
MSD<sup>2</sup>: Minimize sum of distances<sup>2</sup> between robots and targets.
Greedy: Assign robots to targets in order of shortest distances.
Random: Random assignment of robots to targets.

#### **Role Assignment Functions Video**



# Video

- Yellow robots moving to green targets turn red if they collide
- Robot paths turn light blue if robot switches targets (not dynamically consistent)
- Background turns green when all robots have reached targets (makespan completed)

Patrick MacAlpine (UT Austin)

#### RoboCup 3D Positioning Video



Each position is shown as a color-coded number corresponding to the robots's uniform number assigned to that position. Robots update their role assignments and move to new positions as the ball or a robot is beamed (moved) to a new location.

#### RoboCup 2D



Yellow team (SCRAM) vs red team (static)

#### Marking Against Setplays



#### Marking Against Setplays



# What went wrong?

#### SCRAM Prioritized Role Assignment



Large priority value P added to the costs of reaching high priority targets (H) for any agents outside the priority distance of H (purple circle). MacAlpine P. and Stone P., Prioritized Role Assignment for Marking, RoboCup Symposium, 2016.

#### Marking with SCRAM Prioritized Role Assignment



# Video

#### Goals Against Across 1000 Games

Opponent	No Marking	Marking
FCPortugal	160	21
FUT-K	288	63
UTAustinVilla	667	290

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

- SCRAM minimizes the makespan or longest distance any robot has to travel
- SCRAM avoids collisions between robots
- SCRAM role assignment algorithms run in polynomial time and scales to 1000s of robots
- SCRAM is effective in complex RoboCup domains
- Fast C++ implementation: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/scram.html

SCRAM Compared to Related Work in Role Assignment

 Other work has considered topics of collision avoidance, path planning, minimizing sum of distances, and minimizing the makespan

• SCRAM focuses on simultaneous combination of minimizing the makespan and avoiding collisions

SCRAM first to consider dynamic consistency

#### Outline

RoboCup 3D Simulation Domain Description

Overlapping Layered Learning

• SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

Summary and Future Work

#### Outline

RoboCup 3D Simulation Domain Description

Overlapping Layered Learning

 SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan

• Summary and Future Work

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Oevelopment and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Oevelopment and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Development and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Development and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Development and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

- Methodologies for learning complex robot skills that work well together: Overlapping Layered Learning
- Oevelopment and analysis of multirobot role assignment functions: SCRAM role assignment functions
- Novel algorithms for multirobot movement coordination: SCRAM roles assignment algorithms
- Empirical evaluation of learning methodologies and coordination algorithms: Evaluation in RoboCup 3D simulation domain
- Complete autonomous robot soccer playing agent: UT Austin Villa RoboCup 3D simulation team agent and public base code release

#### **Code Release**

#### Code Release URL: https://github.com/LARG/utaustinvilla3d



#### Demo Behavior Second place HARTING Open Source Prize, RoboCup 2016

MacAlpine P. and Stone P., UT Austin Villa RoboCup 3D Simulation Base Code Release, RoboCup Symp., 2016.

Patrick MacAlpine (UT Austin)

PhD Defense

#### **Future Work**

#### • Automated Selection of Overlapping Parameters and Layers

- Segment motion with Hidden Markov Models (HMMs) Niekum et al., Learning and generalization of complex tasks from unstructured demonstrations, 2012.
- Learning options Daniel et al, Probabilistic inference for determining options in reinforcement learning, 2016.
- Relaxation of SCRAM Point Mass Approximation
  - Turpin et al., CAPT: Concurrent assignment and planning of trajectories for multiple robots, 2014.
- Simultaneous Learning and Reshaping of Surrogate Optimization Tasks
  - P. MacAlpine, E. Liebman, and P. Stone, Simultaneous learning and reshaping of an approximated optimization task, 2013.
  - P. MacAlpine, E. Liebman, and P. Stone, Adaptation of surrogate tasks for bipedal walk optimization, 2016.

#### Future Work — Other Domain Applications

- Overlapping Layered Learning
  - Video games
  - Genetic Programming



# SCRAM Role Assignment

- Vehicle routing problems
  - \* Hanna et al., Minimum Cost Matching for Autonomous Carsharing, 2016.
- Warehouse operations
- Quadrotor formation control

#### **Moving Pixel Drones**



(video from https://youtu.be/POtyYqDt\_fA)

#### RoboCup Goal

Have a team of fully autonomous humanoid robot soccer players beat the human World Cup champions by 2050


#### UT Austin Villa RoboCup 3D Simulation Team Members

- Frank Barrera Undergraduate Student (2011)
- Samuel Barrett Graduate Student (2011-2014)
- Yinon Bentor Graduate Student (2009-2010)
- Min Bi Undergraduate Student (2016)
- Nick Collins Undergraduate Student (2011-2012)
- Mike Depinet Undergraduate Student (2013-2014)
- Josiah Hanna Graduate Student (2015)
- Shivaram Kalyanakrishnan Graduate Student (2007-2011)
- Jason Liang Graduate Student (2014-2015)
- Adrian Lopez-Mobilia Undergraduate Student (2011-2012)
- Patrick MacAlpine Graduate Student (2010-present)
- Mahmut Tarik Ozkaya Undergraduate Student (2016)
- Michael Quinlan Research Scientist (2011)
- Art Richards Undergraduate Student (2011)
- Andrew Sharp Undergraduate Student (2013)
- Nicu Stiurca Undergraduate Student (2011)
- Peter Stone Professor (2007-present)
- Jordan Torres Undergraduate Student (2016)
- Matt Union Undergraduate Student (2016)
- Daniel Urieli Graduate Student (2010-2011)
- Victor Vu Undergraduate Student (2011)
- Xinyi Wang Undergraduate Student (2016)

#### More Information

## RoboCup 3D Simulation Homepage: http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/ (Google "UT Austin Villa 3D Simulation")



# Video

## RoboCup 2016 Highlights