Hierarchical Winning Approaches to RoboCup Soccer Simulation Challenge

Aijun Bai

UC Berkeley

Dec 12, 2016

Outline

1. Introduction to RoboCup 2D
2. MAXQ Based Hierarchical Online Planning
3. Concurrent Hierarchical Reinforcement Learning
4. Summary and Future Work

Overview

- MAXQ based hierarchical online planning for large MDPs
  - AAMAS (Bai et al., 2012b)
  - RoboCup Symposium (Bai et al., 2012a, 2013b)
  - ACM Transactions (Bai et al., 2015)
- Thomson sampling based MCTS for MDPs and POMDPs
  - NIPS (Bai et al., 2013a)
  - ICAPS (Bai et al., 2014b; Zhang et al., 2015)
- State and action abstractions for MDPs
  - IJCAI (Bai et al., 2016)

RoboCup Soccer Simulation 2D

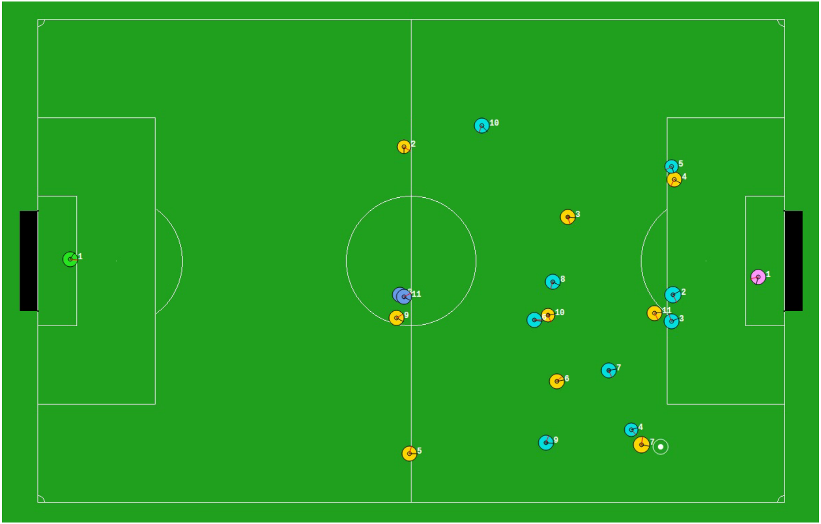
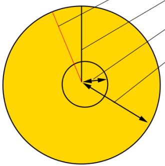


Figure 1: WrightEagle (my team) v.s. Helios

What Makes RoboCup 2D Interesting/Challenging?

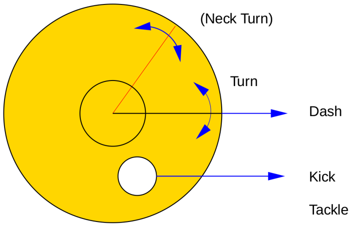
- Key Features:
  - Abstractions made by the simulator
  - High-level planning and learning
  - No need to handle low-level robotics
- Key Challenges:
  - Fully distributed multi-agent stochastic game
  - Continuous state, action and observation spaces
    * A good testbed for many real world problems

Ball and Player States



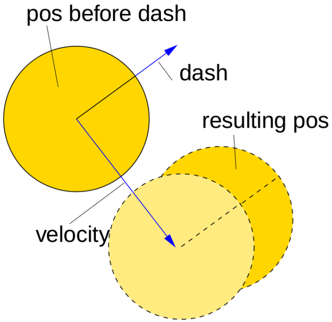
- State: $[s_b, s_1, s_2, \ldots, s_{22}]$
  - Ball State
    - Position, Velocity
  - Player State
    - Position, Velocity
    - Body Angle, Neck Angle, Stamina, ...

Primitive Actions



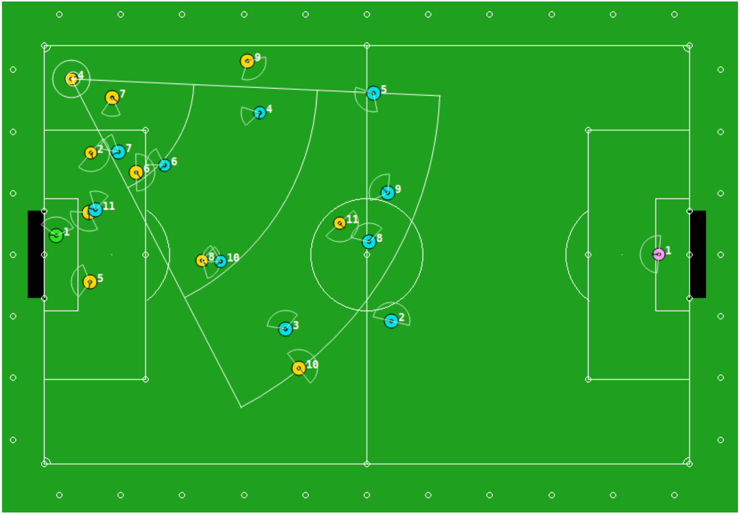
- Parameterized actions
  - $Dash(dir, power)$
  - $TurnBody(angle)$
  - $TurnNeck(angle)$
  - $Kick(dir, power)$
  - $Tackle(dir)$
  - $Catch(dir)$ [for goalie]

The Physics



- $Dash(dir, power)$
  - Moves the player
  - Exposed to noise
  - Costs some stamina
    - If stamina is too low: can not move at full speed

The Observation Model



RoboCup 2D in Action

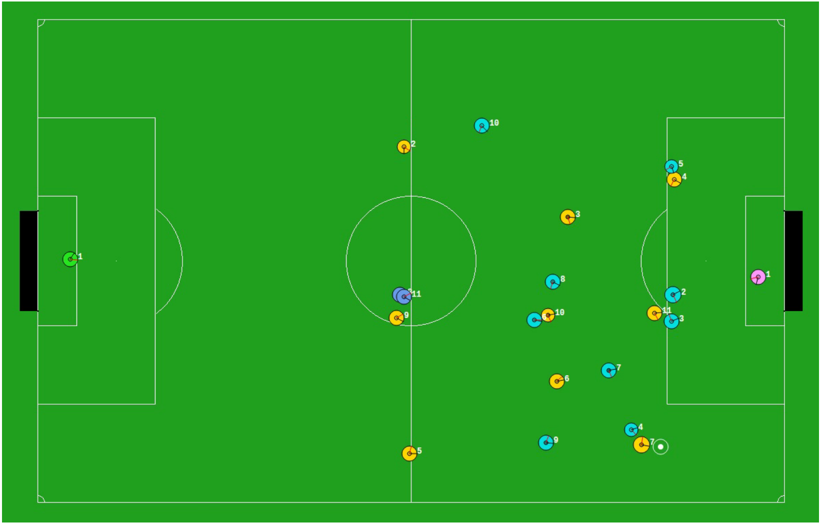


Figure 2: WrightEagle (my team) v.s. Helios

A Multi-Agent Formulation

A partially observable stochastic game (POSG) formulation:

- Agents: $1, 2, \ldots, 22$
- Joint state: $\mathbf{s} = [s_b, s_1, s_2, \ldots, s_{22}]$
- Joint action: $\mathbf{a} = [a_1, a_2, \ldots, a_{22}]$
- Observation: $\mathbf{o} = [landmarks, ball, players, \ldots]$
- Transition function: $T(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \in [0, 1]$
- Observation function: $\Omega_i(\mathbf{o} \mid \mathbf{s}) \in [0, 1]$
- Reward function: $R_i(\mathbf{s}, \mathbf{a})$

which is practically intractable!

Converting to Single-Agent Problem

- Large number of agents involved: 22
  - Belief regression
- Assume teammates/opponents are part of the environment
  - Unpredictable/adaptive/learning
  - Stochastic policies
  - Belief states

Future Belief Prediction

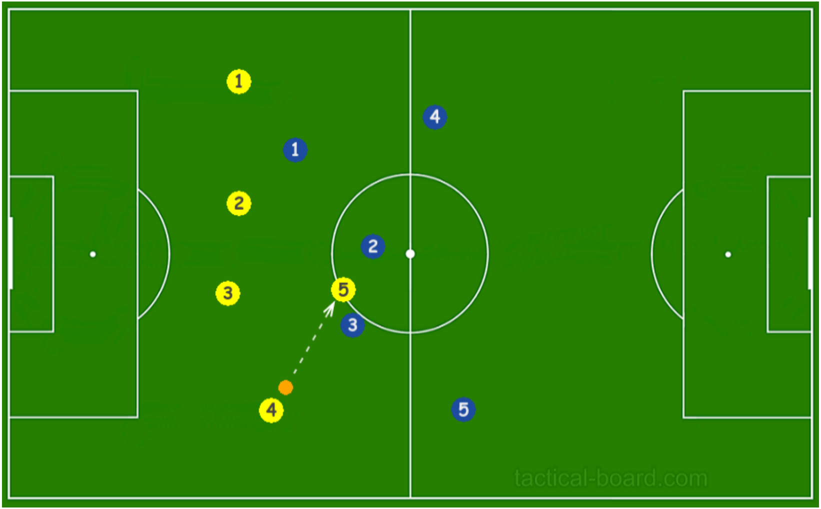


Figure 3: Player 4 planning a 10-cycle pass to player 5 at t

Future Belief Prediction (Cont'd)

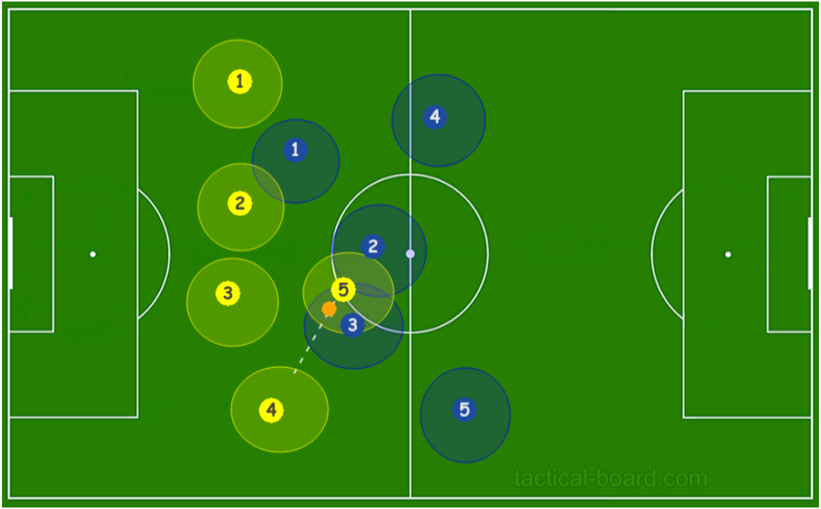


Figure 4: Predicted future state at $t + 10$

Particle Filtering based Belief Update

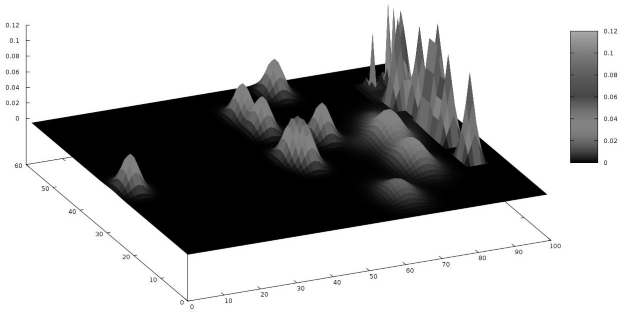


Figure 5: Belief visualization

- Particle filtering over sets for multi-object tracking
  - AAAI fall symposium (Bai et al., 2014a)

A Single-Agent Formulation

- An MDP formulation:
  - Joint belief: $\mathbf{b} = [b_b, b_1, b_2, \ldots, b_{22}]$
  - Action: $a \in \{Dash(\cdot,\cdot), Kick(\cdot,\cdot), TurnBody(\cdot), \ldots\}$
  - Transition function: $T(\mathbf{b}' \mid \mathbf{b}, a) \in [0,1]$
  - Reward function: $R(\mathbf{b}, a)$

Solving the Resulting MDP in Realtime

Exact solution is still intractable:

- The curse of dimensionality
  - Continuous high-dimensional (belief) state space
  - Continuous action space
- The curse of history
  - Long looking-ahead horizon: 6000 steps for a game
  - Sparse reward function
    - +/-1 for teammate/opponent score

Approximate Solutions

My Proposals:

- Hierarchical online planning
  - Perform a tree search from the current state within a reachable subspace consistent with a task structure
- Concurrent hierarchical reinforcement learning
  - Write a partial policy for each agent
  - Jointly learn an optimal completion for partial policies
  - An ongoing project

Hierarchical Decomposition

- Exploit hierarchical structure of a task
- Introduce macro actions/options/skills
  - In contrast with primitive actions
- Fundamental theory: Semi-MDP
  - Options: $o \in \mathcal{O}$
  - Multi-step transition function: $T(s', N \mid s, o) \in [0, 1]$
  - Hierarchical policy: $\pi(s) \in \mathcal{O}$
  - Fast planning and learning on an option basis

MAXQ Hierarchical Decomposition

- Decompose an MDP into a tree of sub-MDPs (Dietterich, 1999)
  - $M = \{M_0, M_1, \ldots, M_n\}$
  - $M_i = \langle S_i, G_i, A_i, R_i \rangle$
    1. Active states S_i
    2. Goal states G_i
    3. Available actions A_i
    4. Local reward function R_i
  - Solving M_0 solves the original MDP M
  - Hierarchical policy $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$

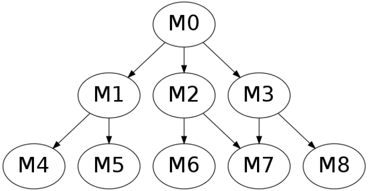


Figure 6: MAXQ hierarchy

MAXQ Hierarchical Decomposition: An Example

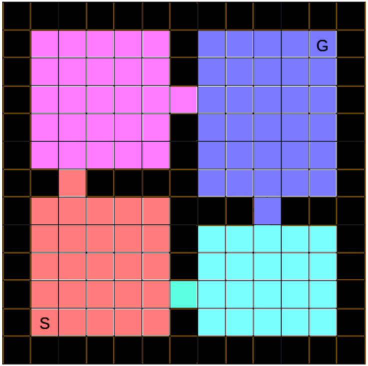


Figure 7: A 4-Room domain

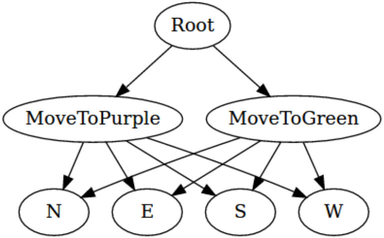


Figure 8: MAXQ task graph

MAXQ Value Function Decomposition

- Value function V^* of optimal π^* satisfies

$$V^*(i,s) = \begin{cases} R(s,i) & \text{if } M_i \text{ is primitive} \\ \max_{a \in A_i} Q^*(i,s,a) & \text{otherwise} \end{cases} \tag{1}$$

$$Q^*(i,s,a) = V^*(a,s) + \sum_{s',N} \Pr(s', N \mid s,a) V^*(i,s') \tag{2}$$

- Transition function: $\Pr(s', N \mid s, a)$
  - Marginalized: $\Pr(s' \mid s,a) = \sum_N \Pr(s', N \mid s,a)$
    - Terminating distribution

MAXQ based Online Planning: MAXQ-OP

- For non-primitive subtasks

$$V^*(i,s) \approx \max_{a \in A_i} \left\{ V^*(a,s) + \sum_{s'} \Pr(s' \mid s,a) V^*(i,s') \right\} \quad (3)$$

- Introduce search depth array d, maximal search depth array D and heuristic function $H(i,s)$

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geq D[i] \\ \max_{a \in A_i} \{V(a,s,d) + \\ \sum_{s'} \Pr(s' \mid s,a) V(i,s',d[i] \leftarrow d[i]+1)\} & \text{otherwise} \end{cases} \quad (4)$$

- Call $V(0,s,[0,0,\ldots,0])$ to find the value of s in task M_0

MAXQ-OP on Benchmark: The Taxi Problem

- States: $25 \times 5 \times 4 = 400$
  1. Taxi location: (x, y)
  2. Passenger location: R, Y, B, G and In
  3. Destination location: R, Y, B, G
- Actions: 6
  1. North, South, East, West
  2. Pickup, Putdown
- Probability 0.8 of failure

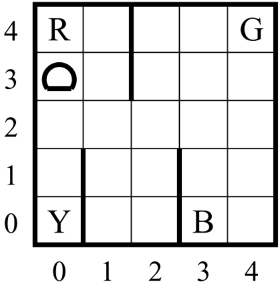


Figure 9: Taxi domain

Empirical Results in the Taxi Problem

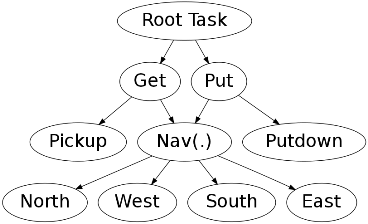


Figure 10: Task graph for Taxi

Table 1: Empirical results in Taxi

Algorithm	Avg. Reward*	Online Time (ms)
MAXQ-OP	3.93 ± 0.16	0.20 ± 0.16
LRTDP	3.71 ± 0.15	64.88 ± 3.71
AOT	3.80 ± 0.16	41.26 ± 2.37
UCT	-23.10 ± 0.84	102.20 ± 4.24

*The upper bound of Average Rewards is 4.01 ± 0.15.

Comparing to Non-Hierarchical Online Planning

- Non-hierarchical online planning algorithms
  - Use only primitive actions to grow search tree
  - Search path:

$$[s_1 \to s_2 \to s_3 \to \cdots \to s_H] \rightsquigarrow g \tag{5}$$

- MAXQ-OP algorithm
  - Use primitive and macro actions to grow search tree
  - Search path:

$$\begin{multline}[s_1 \to \cdots \to s_{H_1}] \rightsquigarrow [g_1/s'_1 \to \cdots \to s'_{H_2}] \rightsquigarrow \\ [g_2/s''_1 \to \cdots \to s''_{H3}] \cdots \rightsquigarrow g \end{multline} \tag{6}$$

- MAXQ-OP can search much deeper by exploiting the task structure

MAXQ Decomposition for RoboCup 2D

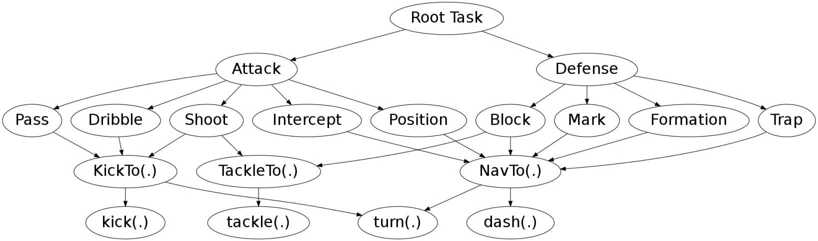


Figure 11: MAXQ based hierarchical decomposition in WrightEagle

Hierarchical Online Planning in WrightEagle

- Rule-based system:

  PlanAttack() {
  ...
  if *should_shoot* **then**
    return Shoot()
  else if *should_pass* **then**
    return Pass()
  else
    return Drrible()
  ...
  }

- Hierarchical planning:

  PlanAttack() {
  ...
  shoot $\leftarrow$ PlanShoot()
  pass $\leftarrow$ PlanPass()
  dribble $\leftarrow$ PlanDrrible()
  ...
  return $\max\{$shoot, pass, dribble, $\ldots\}$
  }

MAXQ-OP in WrightEagle

- Task hierarchy
  - Value function decomposition
- Terminating distribution approximation
  - Success and failure probabilities
- Heuristic tree search
- Local reward functions

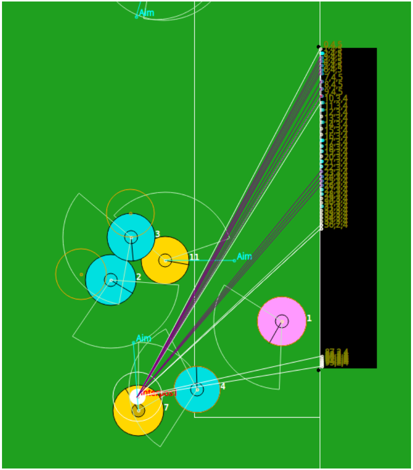


Figure 12: Search in shoot

Hierarchical Planning for Pass Behavior

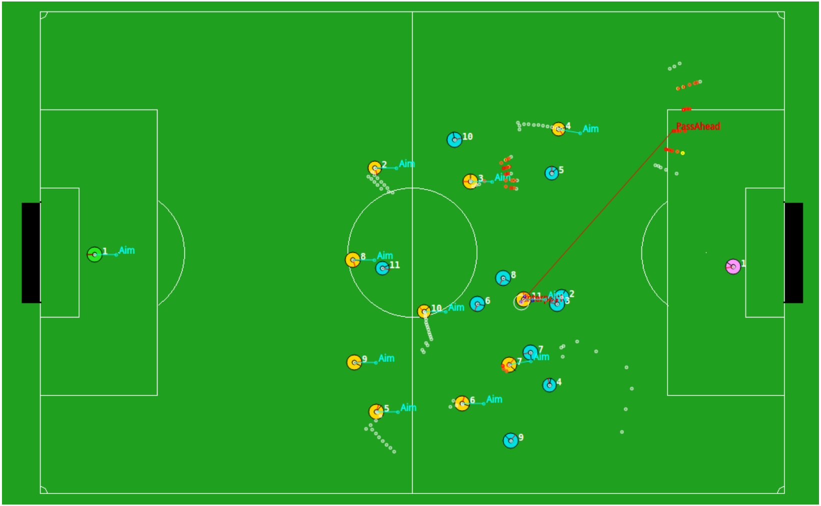


Figure 13: Hierarchical planning for pass behavior

Monte Carlo Tree Search

- Transitions as explicit distributions $\Pr(s' \mid s, a)$ are not available
- Sampling rules $s' \sim \Pr(s' \mid s, a)$ are clearly defined by the simulator
- Monte-Carlo tree search w/ state abstraction
- Low-level skills: $NavTo$, $KickTo$, ...

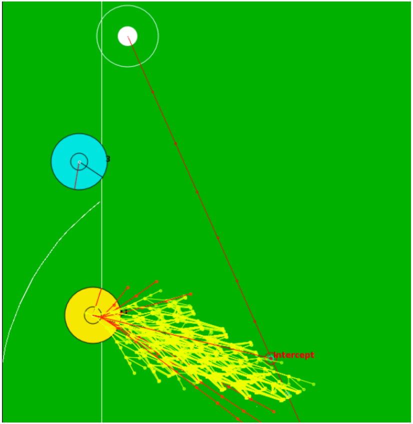


Figure 14: Search tree in *NavTo*

Terminating Distribution Estimation - An Example

- $\Pr(s_{goal} \mid s, KickTo(t)) \approx (1 - p_1) \times p_2$
  - $p_1 = \Pr(s_{caught} \mid s, KickTo(t))$
  - $p_2 = \Pr(s_{in} \mid s, KickTo(t))$

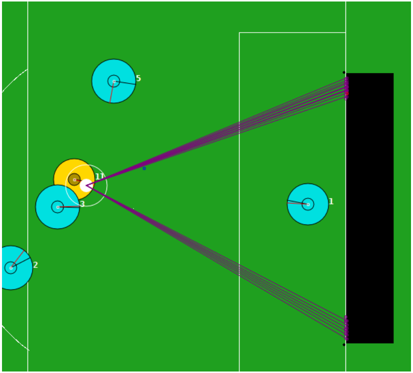


Figure 15: A shoot scenario

Terminating Distribution Estimation - An Example (Cont'd)

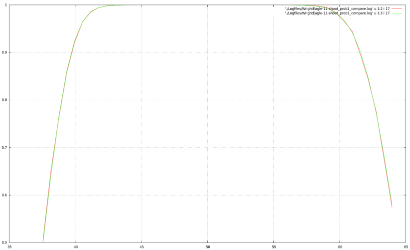


Figure 16: $\Pr(s_{in} \mid s, KickTo(t))$

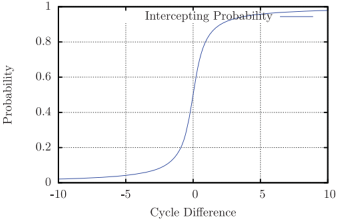


Figure 17: $\Pr(s_{caught} \mid s, KickTo(t)) \approx \max_p f(t_b(p,s) - t_g(p,s) \mid p)$

Heuristic Evaluation

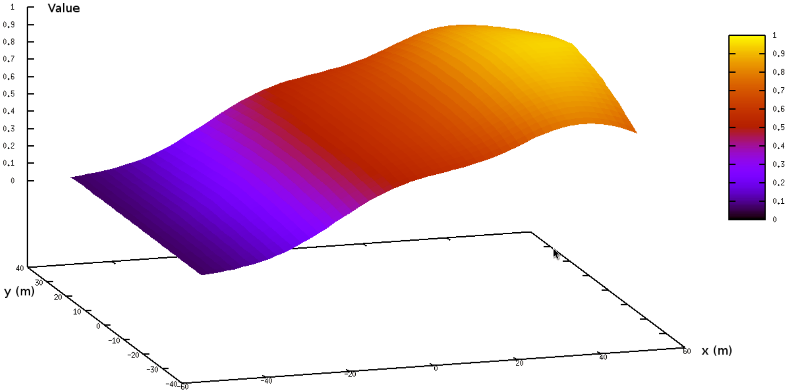


Figure 18: A heuristic function based on ball position

The Idea of Virtual Agent

- The state is augmented with a virtual agent index which is transferred among teammates
  - The teammate controlling the ball
- Search tree is grown from the perspective of the virtual agent
  - Player 5 is doing decision-making
    - A search tree with player 5 as the virtual agent at the root node
    - Player 5 passes the ball to player 7
    - Player 7 becomes the virtual agent for the subtree thereof

Achievements

WrightEagle team (from Univ. of Sci. & Tech. of China):

- Started work on RoboCup 2D since 2006
- Became the main contributor since 2009
- Won 5 world champions: 2009, 2011, 2013, 2014 and 2015
- Published on NIPS, AAMAS, ICAPS, RoboCup, etc.
- More information:
  - `https://en.wikipedia.org/wiki/RoboCup_2D_Soccer_Simulation_League`
  - `http://www.wrighteagle.org/2d/`

Hierarchical Reinforcement Learning

- Hierarchical planning:

  PlanAttack() {
  ...
  shoot $\leftarrow$ PlanShoot()
  pass $\leftarrow$ PlanPass()
  dribble $\leftarrow$ PlanDrrible()
  ...
  return $\max\{$shoot, pass,
  dribble, ...$\}$
  }

Hierarchical abstract machines (HAMs) (Parr & Russell, 1998):

- Write a partial program for an agent
- Leave some undetermined choice points
- Use reinforcement learning to learn an optimal completion

Partial Program

Partial program: an incomplete policy $\pi(s)$ with (many) unspecified choice states

- A hierarchical finite state machine with choice states
- Equivalently, a piece of code with choice macros

Partial Program - an Example

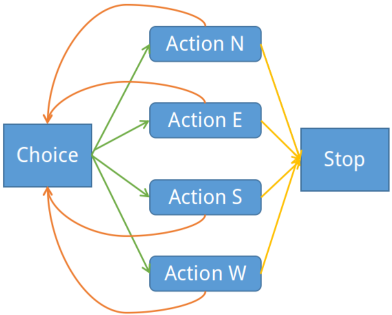


Figure 19: An FSM

Navigate(State s) {
while *!atDestination(s)* **do**
  a ← Choose({N, E, S, W})
  Action(a)
  s ← GetState()
}

Hierarchical Abstract Machines

- A hierarchy of partial programs
- High-level programs can call low-level programs
- Each program has its own:
  - Machine state
  - Call stack
  - Internal memory

M_1(State s) {
while *!terminated(M_1, s)* **do**
  m ← $Choose_1(\{M_2, M_3\})$
  Call(m)
}

M_2(State s) {
while *!terminated(M_2, s)* **do**
  a ← $Choose_2(\{a_1, a_2\})$
  Action(a); s ← GetState()
}

Hierarchical Reinforcement Learning with HAMs

An agent executing a partial program:

- An MDP over joint space of environment and machine states
- An SMDP over choice points
  - Choice point: a joint state with machine state as a choice state
  - $Q(s, m, c) \leftarrow R(s, m, c, s', m', \tau) + \gamma^{\tau} Q(s', m', c')$

Deterministic Transitions Among Choice Points

Deterministic transitions:

- $(s, [M_1, Choose_1], M_2) \rightarrow (s, [M_1, M_2, Choose_2])$
- $\tau = 0$
- $r = 0$
- $Q(s, m, c) = V(s, m') = \max_{c'} Q(s, m', c')$

M_1(State s) {
while *!terminated(M_1, s)* **do**
  m ← $Choose_1$({M_2, M_3})
  Call(m)
}

M_2(State s) {
while *!terminated(M_2, s)* **do**
  a ← $Choose_2$({a_1, a_2})
  Action(a); s ← GetState()
}

Concurrent Hierarchical Learning with HAMs

- Each agent has its own partial program
- Running and learning concurrently
  - $Q(\mathbf{s}, \mathbf{m}, \mathbf{c}) \leftarrow R(\mathbf{s}, \mathbf{m}, \mathbf{c}, \mathbf{s}', \mathbf{m}', \tau) + \gamma^\tau Q(\mathbf{s}', \mathbf{m}', \mathbf{c}')$
  - Share machine state and value functions
  - Make joint choices when available

Synchronization Semantics

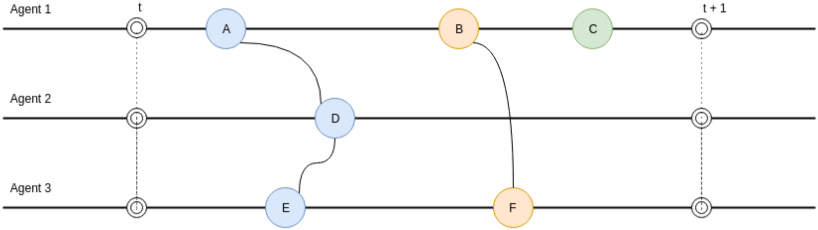


Figure 20: Synchronization for concurrent learning

RoboCup Keepaway Task

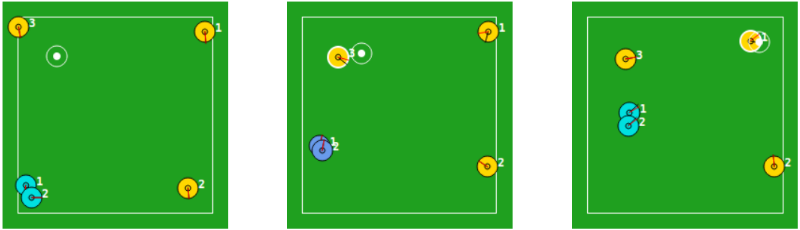


Figure 21: A 3 vs 2 RoboCup keepaway task

- Keepers: maintain possession of the ball
- Takers: gain possession of the ball
- Assume fully communication with shared memory

HAMs for RoboCup Keepaway

Partial program for keepers:

- If fastest to intercept, intercept
- If kickable, call hold or pass
- Otherwise, call stay or move

Unspecified choice states:

- Choose({Hold, Pass})
  - Choose({PassTo1, PassTo2}), Choose({PassSpeed...})
- Choose(Stay, Move)
  - Choose({MoveToDir...}), Choose({MoveSpeed...})

Function Approximation

- State: 15 dimensional feature vectors
  - $\mathbf{s} = [f_1, f_2, \ldots, f_{15}]$
  - Distances and angles
- Tile coding (encode $\mathbf{s} \times \mathbf{m}$ into a huge binary vector)
  - $\mathbf{s} \times \mathbf{m} = [1000101010..0101]_{50,000} = [t_1, t_2, \ldots, t_{480}]$
- Linear SARSA learning with binary features
  - $Q(\mathbf{s}, \mathbf{m}, \mathbf{c}) \approx w_{\mathbf{c}}[t_1] + w_{\mathbf{c}}[t_2] + \cdots + w_{\mathbf{c}}[t_{480}]$

Demonstration: Learned Policy

- Learned policy at early stage
- Converged policy

Summary

- RoboCup soccer simulation 2D
  - A belief-MDP formulation
- A hierarchical planning approach
  - MAXQ-OP online planning
- A hierarchical learning approach
  - Concurrent reinforcement learning with HAMs

Future Work

- Hierarchical Monte Carlo planning
- Reward decomposition and reward shaping
- Coordination graph for joint choices
- Game theoretical planning/learning

Thank you!

References I

Bai, A., Chen, X., MacAlpine, P., Urieli, D., Barrett, S., & Stone, P. (2012a). Wright Eagle and UT Austin Villa: RoboCup 2011 simulation league champions. In T. Roefer, N. M. Mayer, J. Savage, & U. Saranli (Eds.) *RoboCup-2011: Robot Soccer World Cup XV*, vol. 7416 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer Verlag.

Bai, A., Simmons, R., Veloso, M., & Chen, X. (2014a). Intention-aware multi-human tracking for human-robot interaction via particle filtering over sets. In *2014 AAAI Fall Symposium Series*.

Bai, A., Srivastava, S., & Russell, S. J. (2016). Markovian state and action abstractions for MDPs via hierarchical MCTS. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, (pp. 3029–3039).
URL http://www.ijcai.org/Abstract/16/430

Bai, A., Wu, F., & Chen, X. (2012b). Online planning for large MDPs with MAXQ decomposition (extended abstract). In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012)*.

Bai, A., Wu, F., & Chen, X. (2013a). Bayesian mixture modelling and inference based Thompson sampling in Monte-Carlo tree search. In *Advances in Neural Information Processing Systems 26*, (pp. 1646–1654).

References II

Bai, A., Wu, F., & Chen, X. (2013b). Towards a principled solution to simulated robot soccer. In X. Chen, P. Stone, L. E. Sucar, & T. V. der Zant (Eds.) *RoboCup-2012: Robot Soccer World Cup XVI*, vol. 7500 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer Verlag.

Bai, A., Wu, F., & Chen, X. (2015). Online planning for large markov decision processes with hierarchical decomposition. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *6*(4), 45.

Bai, A., Wu, F., Zhang, Z., & Chen, X. (2014b). Thompson sampling based Monte-Carlo planning in POMDPs. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*. Portsmouth, United States.

Dietterich, T. G. (1999). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Machine Learning Research*, *13*(1), 63.

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, (pp. 1043–1049).

Zhang, Z., Hsu, D., Lee, W. S., Lim, Z. W., & Bai, A. (2015). PLEASE: palm leaf search for POMDPs with large observation spaces. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, (pp. 249–258).