Hierarchical Winning Approaches to RoboCup Soccer Simulation Challenge

Aijun Bai

Nov 4, 2016

UC Berkeley

Outline

Overview

- 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 Monte-Carlo tree search
  - NIPS (Bai et al., 2013a)
  - ICAPS (Bai et al., 2014; Zhang et al., 2015)
- State and action abstractions for MDPs
  - IJCAI (Bai et al., 2016)

Introduction to RoboCup 2D

RoboCup Soccer Simulation 2D

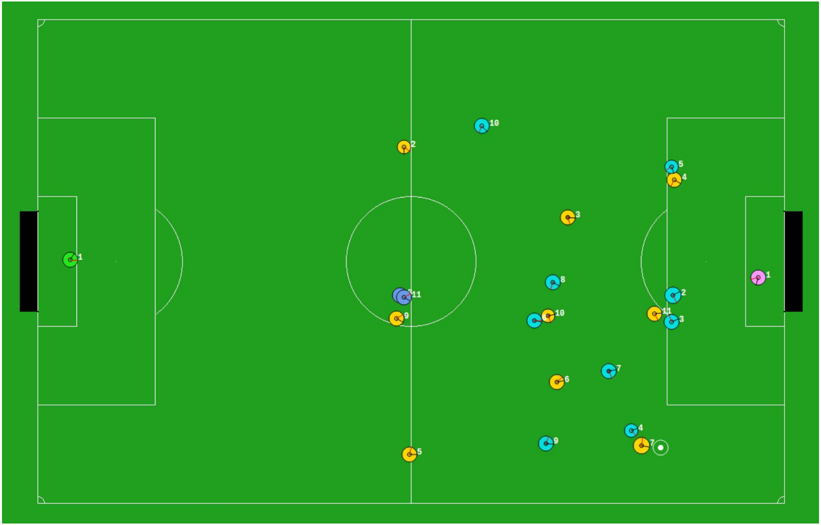
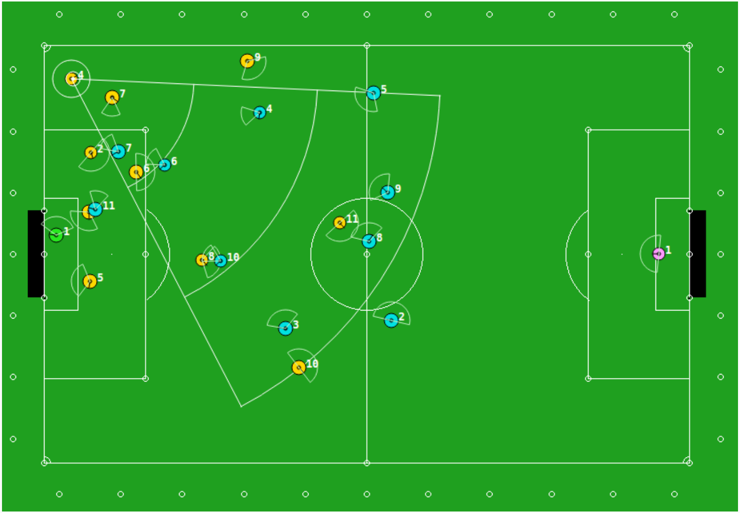


Figure 1: WrightEagle (my team) v.s. Helios

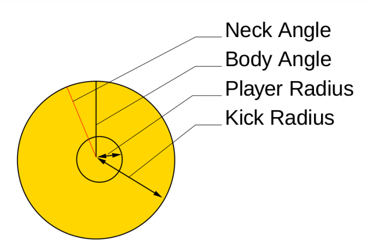
What Makes RoboCup 2D Interesting/Challenging?

- Key Features:
  - Abstractions made by the simulator
  - High-level planning and learning
  - No need to handle robot hardware issues
- Key Challenges:
  - Fully distributed multi-agent stochastic system
  - Continuous state, action and observation spaces
  - Exact solutions simply do not exist!

The View Model

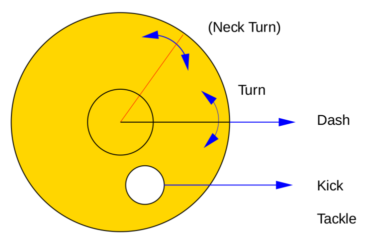


Ball and Player States



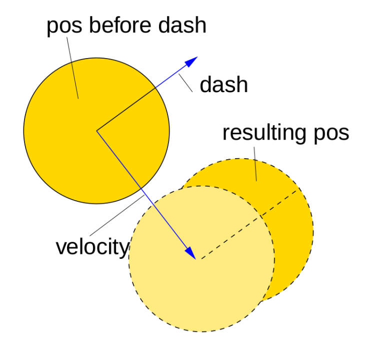
- Ball
  - Position, Velocity
- Player
  - Position, Velocity, Body Angle, Neck Angle, Stamina, . . .
  - Maximal Speed, Kickable Area, Stamina Recovery, . . .

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

RoboCup 2D in Action

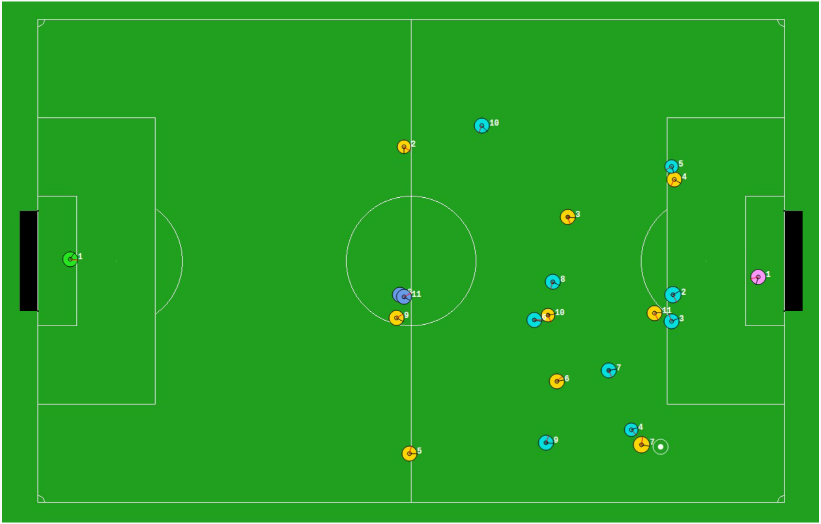


Figure 2: WrightEagle (my team) v.s. Helios

An Intractable Joint Formulation

A partially observable stochastic game (POSG) formulation:

- Agents: $1, 2, \ldots, 22$
- Joint state: $\mathbf{s} = [s_0, s_1, s_2, \ldots, s_{22}]$
- Joint action: $\mathbf{a} = [a_1, a_2, \ldots, a_{22}]$
- Transition function: $T(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \in [0, 1]$
- Observation: $\mathbf{o} = [landmarks, ball, players, \ldots]$
- Observation function: $\Omega_i(\mathbf{o} \mid \mathbf{a}, \mathbf{s}) \in [0, 1]$
- Reward function: $R_i(\mathbf{s}, \mathbf{a})$

which is mathematically intractable!

A Single-Agent Formulation

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

A Single-Agent Formulation (Cont'd)

- Approximately use belief state to predict their future state
  - Belief state: a distribution over possible states
- An MDP formulation:
  - Joint belief: $\mathbf{b} = [b_0, b_1, b_2, \ldots, b_{22}]$
  - Action: a
  - Transition function: $T(\mathbf{b}' \mid \mathbf{b}, a) \in [0, 1]$
  - Reward function: $R(\mathbf{b}, a)$

Future Belief Prediction

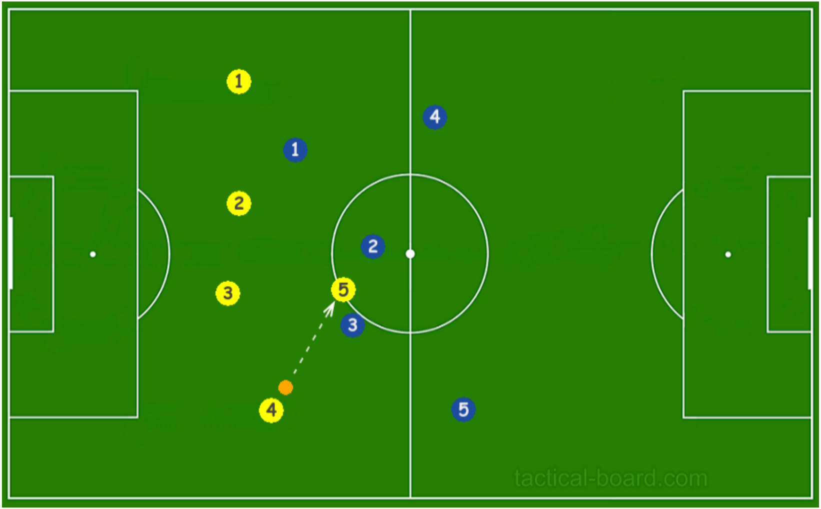


Figure 3: Player 4 planning a 10-cycle pass to player 5 at t_1

Future Belief Prediction (Cont'd)

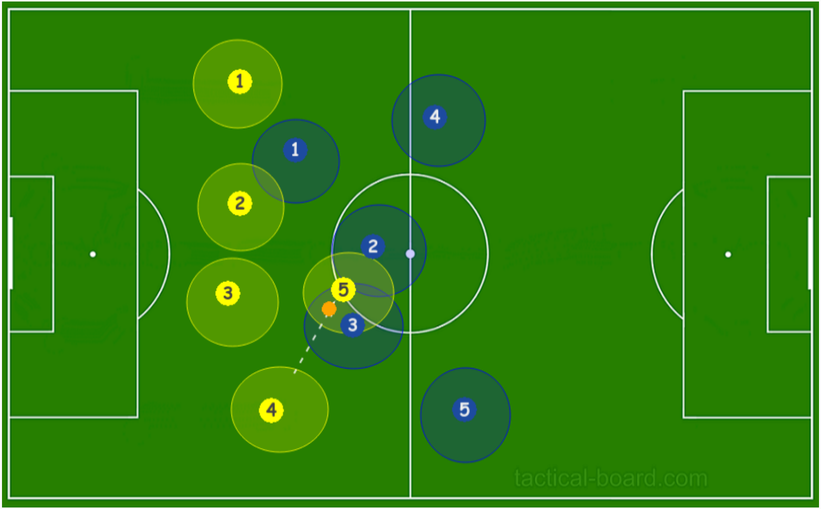


Figure 4: Predicted future state at t_{10}

Particle Filtering based Belief Update

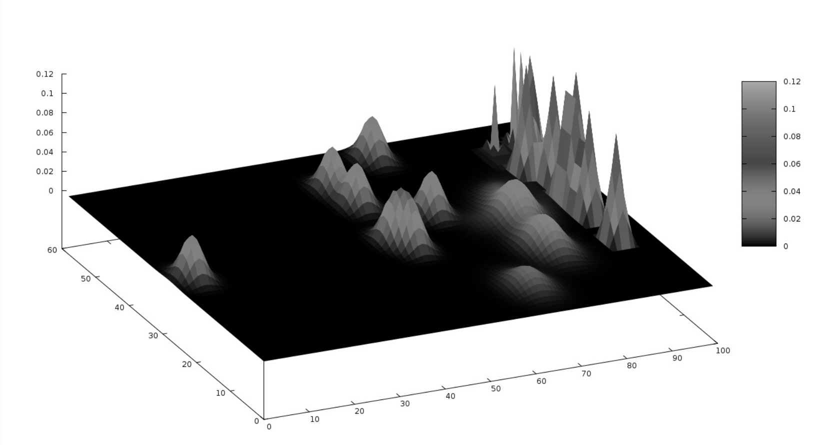


Figure 5: Belief visualization

Solving the Single-Agent MDP in Realtime

Exact solution is still intractable:

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

Approximate Solutions

My Proposals:

- MAXQ based hierarchical online planning
  - Perform a tree search from the current state consistent with a task structure
- HAM based concurrent hierarchical reinforcement learning
  - Learn an optimal completion for partial programs
  - An ongoing project

A Hierarchical Planning Approach

Hierarchical Decomposition

- Exploit hierarchical structure of a task
- Introduce macro actions/options/skills
  - In contrast with primitive actions
  - People are good at this!
- 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 set 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 $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$

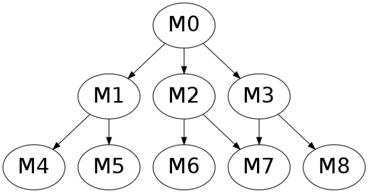


Figure 6: MAXQ hierarchy

MAXQ Value Function Decomposition

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

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

$$Q^*(i,s,a) = V^*(a,s) + C^*(i,s,a) \tag{2}$$

$$C^*(i,s,a) = \sum_{s',N} \Pr(s',N \mid s,a) V^*(i,s') \tag{3}$$

- Contradiction: estimating $\Pr(s',N \mid s,a)$ online = solving offline

MAXQ based Online Planning: MAXQ-OP

- Approximating $\Pr(s' \mid s, a) = \sum_N \Pr(s', N \mid s, a)$ is easy
- For non-primitive subtasks

$$V^*(i,s) \approx \max_{a \in A_i} \left\{ V^*(a,s) + \sum_{s'} \Pr(s' \mid s, a) V^*(i, s') \right\} \quad (4)$$

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

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geq D[i] \\ \max_{a \in A_i} \{ V(a,s,d) + & \\ \sum_{s'} \Pr(s' \mid s, a) V(i, s', d[i] \leftarrow d[i]+1) \} & \text{otherwise} \end{cases} \quad (5)$$

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

Comparing to Non-Hierarchical Online Planning

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

$$[s_1 \to s_2 \to s_3 \to \cdots \to s_H] \rightsquigarrow g \tag{6}$$

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

$$\begin{aligned}[s_1 \to \cdots \to s_{H_1}] \rightsquigarrow [g_1/s'_1 \to \cdots \to s'_{H_2}] \rightsquigarrow \\ [g_2/s''_1 \to \cdots \to s''_{H3}] \cdots \rightsquigarrow g\end{aligned} \tag{7}$$

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

MAXQ Decomposition for RoboCup 2D

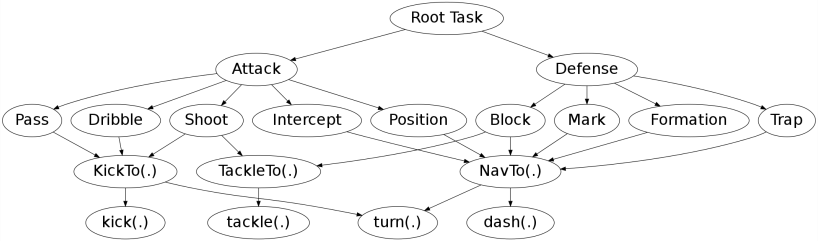


Figure 7: MAXQ based hierarchical decomposition in WrightEagle

Hierarchical Online Planning in WrightEagle

- Rule-based system:

  PlanAttack() {

  ...

  if *should_shoot* **then**
      return PlanShoot()

  else if *should_pass* **then**
      return PlanPass()

  else
      return PlanDrrible()

  ...

  }

- Hierarchical planning:

  PlanAttack() {

  ...

  shoot $\leftarrow$ PlanShoot()

  pass $\leftarrow$ PlanPass()

  dribble $\leftarrow$ PlanDrrible()

  ...

  return $\max${shoot, pass, dribble, ...}

  }

The Idea of Virtual Agent

- The agent controlling the ball is assumed to be the leader
- Tree search is done from the perspective of the leader
  - Agent 5 is making decision
    - A search tree with agent 5 at the root node
    - Agent 5 passes the ball to agent 7
    - A virtual agent 7 will be doing the tree search after receiving the ball
  - Agent 5 selects an action to execute based on resulting Q values

Value Function Decomposition in WrightEagle

$$Q^*(\text{Root}, \boldsymbol{s}, \text{Attack}) = V^*(\text{Attack}, \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{Attack}) V^*(\text{Root}, \boldsymbol{s}'), \tag{8}$$

$$V^*(\text{Root}, \boldsymbol{s}) = \max\{Q^*(\text{Root}, \boldsymbol{s}, \text{Attack}), Q^*(\text{Root}, \boldsymbol{s}, \text{Defense})\}, \tag{9}$$

$$\begin{aligned} V^*(\text{Attack}, \boldsymbol{s}) = \max\{&Q^*(\text{Attack}, \boldsymbol{s}, \text{Pass}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Dribble}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Shoot}), \\ &Q^*(\text{Attack}, \boldsymbol{s}, \text{Intercept}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Position})\}, \end{aligned} \tag{10}$$

$$Q^*(\text{Attack}, \boldsymbol{s}, \text{Pass}) = V^*(\text{Pass}, \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{Pass}) V^*(\text{Attack}, \boldsymbol{s}'), \tag{11}$$

$$Q^*(\text{Attack}, \boldsymbol{s}, \text{Intercept}) = V^*(\text{Intercept}, \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{Intercept}) V^*(\text{Attack}, \boldsymbol{s}'), \tag{12}$$

$$V^*(\text{Pass}, \boldsymbol{s}) = \max_{\text{position } p} Q^*(\text{Pass}, \boldsymbol{s}, \text{KickTo}(p)), \tag{13}$$

$$V^*(\text{Intercept}, \boldsymbol{s}) = \max_{\text{position } p} Q^*(\text{Intercept}, \boldsymbol{s}, \text{NavTo}(p)), \tag{14}$$

$$Q^*(\text{Pass}, \boldsymbol{s}, \text{KickTo}(p)) = V^*(\text{KickTo}(p), \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{KickTo}(p)) V^*(\text{Pass}, \boldsymbol{s}'), \tag{15}$$

$$Q^*(\text{Intercept}, \boldsymbol{s}, \text{NavTo}(p)) = V^*(\text{NavTo}(p), \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{NavTo}(p)) V^*(\text{Intercept}, \boldsymbol{s}'), \tag{16}$$

$$V^*(\text{KickTo}(p), \boldsymbol{s}) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\text{KickTo}(p), \boldsymbol{s}, \text{kick}(a, \theta)), \tag{17}$$

$$V^*(\text{NavTo}(p), \boldsymbol{s}) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\text{NavTo}(p), \boldsymbol{s}, \text{dash}(a, \theta)), \tag{18}$$

$$Q^*(\text{KickTo}(p), \boldsymbol{s}, \text{kick}(a, \theta)) = R(s, \text{kick}(a, \theta)) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{kick}(a, \theta)) V^*(\text{KickTo}(p), \boldsymbol{s}'), \tag{19}$$

MAXQ-OP in WrightEagle

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

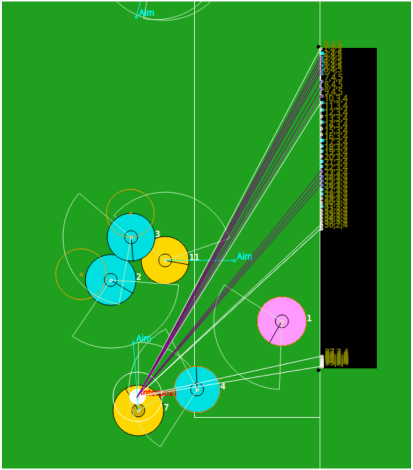


Figure 8: Search in shoot

Hierarchical Planning for Pass Behavior

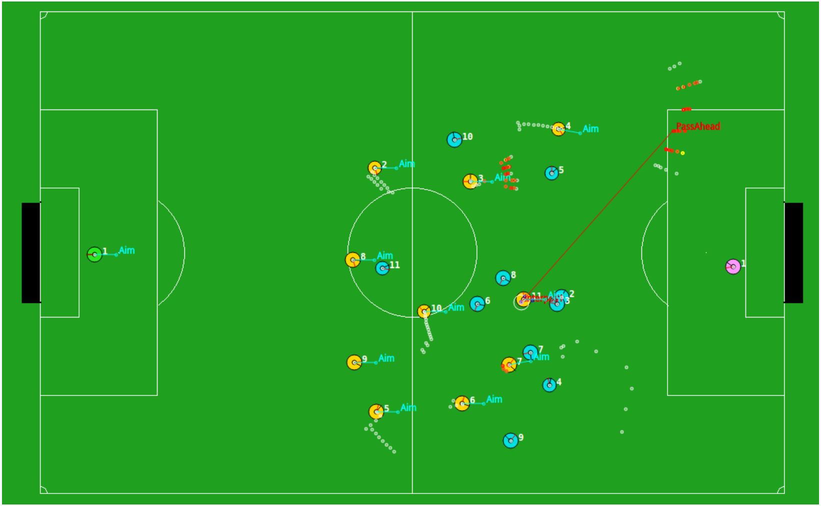


Figure 9: Hierarchical planning for pass behavior

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: $NavTo$, $KickTo$, $\ldots$

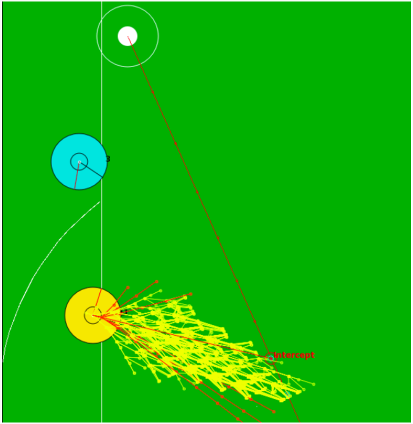


Figure 10: Search tree in *NavTo*

Terminating Distribution Estimation - An Example

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

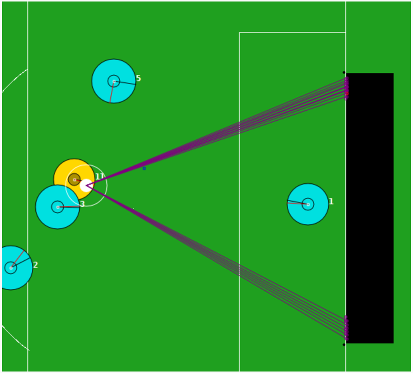


Figure 11: A shoot scenario

Terminating Distribution Estimation - An Example (Cont'd)

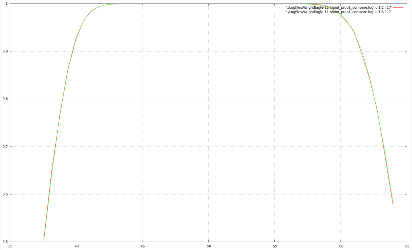


Figure 12: $\Pr(s_{in} \mid s, KickTo)$

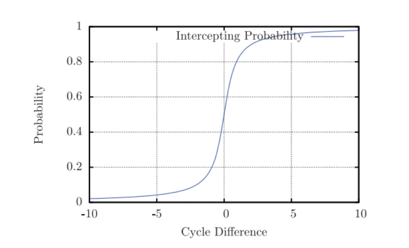


Figure 13: $\Pr(s_{caught} \mid s, KickTo)$

Heuristic Evaluation

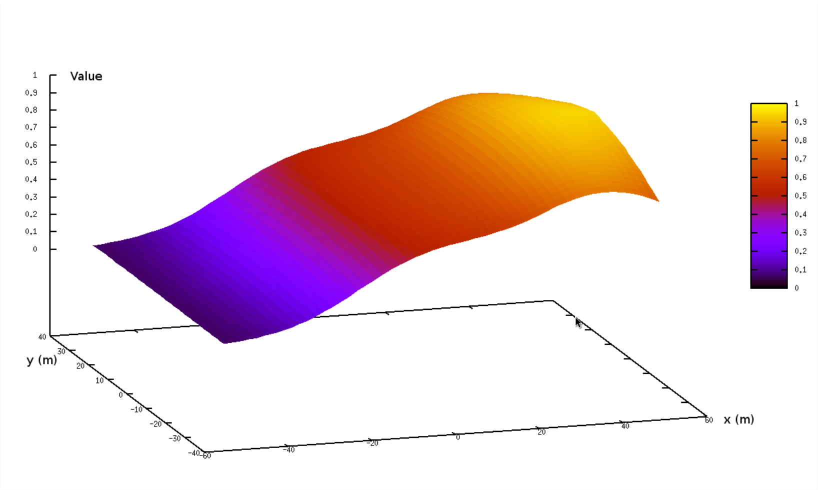


Figure 14: A heuristic function based on ball position

Achievements

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

- Started work on RoboCup 2D since 2006
- Became the main contributor since 2009
- 5 world champions: 2009, 2011, 2013, 2014 and 2015
- More information: `http://www.wrighteagle.org/2d/`

Benchmark: The Taxi Domain

- 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 of 0.8 of success
- Probability of 0.2 of failure

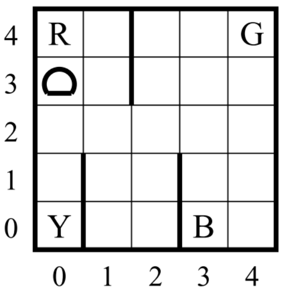


Figure 15: Taxi domain

Empirical Results in the Taxi Domain

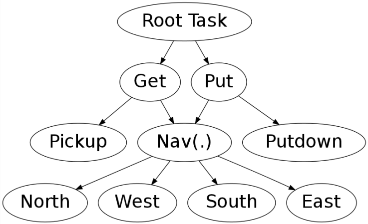


Figure 16: Task graph for Taxi

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.

A Hierarchical Learning Approach

Hierarchical Reinforcement Learning

- Hierarchical planning:

  PlanAttack() $\{$

  $\ldots$

  shoot $\leftarrow$ PlanShoot()

  pass $\leftarrow$ PlanPass()

  dribble $\leftarrow$ PlanDrrible()

  $\ldots$

  return $\color{red}{\max}$$\{$shoot, pass,
  dribble, $\ldots\}$

  $\}$

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

- Write a partial program for an agent
- Leave some unspecified 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

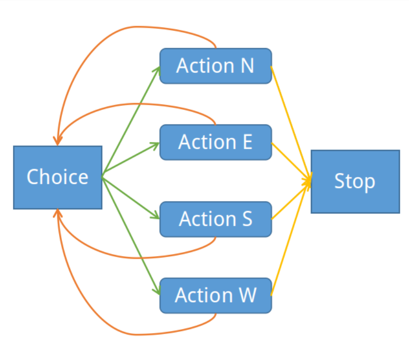


Figure 17: An FSM

Navigate(State s) {

while *!atDestination(s)* **do**

  a ← Choose({N, E, S, W})

  Action(a)

  s ← GetState()

}

HAM learning

An agent executing a partial program:

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

Concurrent HAM learning

Concurrent HAM learning:

- Each agent has its own partial program
- Running concurrently
- Making joint choices as much as possible
- Learning with shared machine state and value functions
  - $Q(\mathbf{s}, \mathbf{m}, \mathbf{c}) \leftarrow R(\mathbf{s}, \mathbf{m}, \mathbf{c}, \mathbf{s}', \mathbf{m}', \tau) + \gamma^{\tau} Q(\mathbf{s}', \mathbf{m}', \mathbf{c}')$

RoboCup Keepaway Task

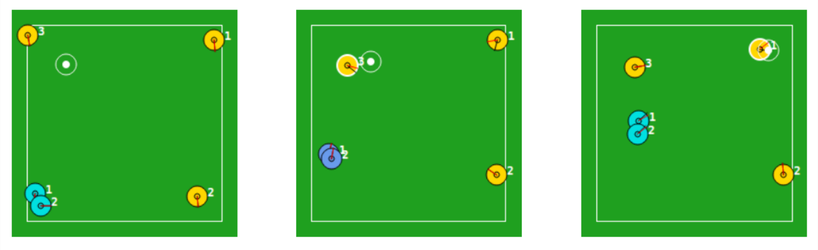


Figure 18: A 3vs2 RoboCup keepaway task

- 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 and Future Work

Summary

- RoboCup soccer simulation 2D
  - Single-agent MDP formulation
- Hierarchical planning approach
  - MAXQ-OP online planning
- 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

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., 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)*.

References ii

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).

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. (2014). 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).