UC, Berkeley

Planning under Uncertainty in RoboCup Soccer Simulation 2D

Aijun Bai

http://home.ustc.edu.cn/~baj/

University of Science and Technology of China

September 12, 2014

Outline

Outline

The RoboCup 2D domain - introduction

- Simulated soccer game
- Two teams of 11 players
- Independently controlled
- In each cycle (100ms)
  - Receive perception
  - Make decision
  - Send action(s)
- Normally 6,000 cycles

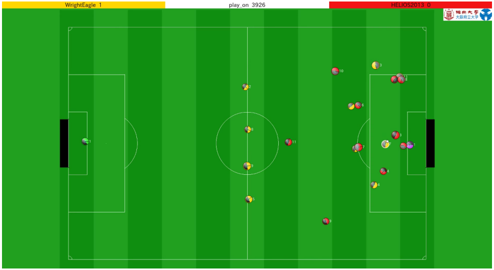


Figure 1 : RoboCup 2D.

The RoboCup 2D domain - model

- State:
  - Ball state, player states and game information
- Observation:
  - Visual information (within field of view):
    - Simulated ball, landmark, and player detections
  - Aural information: msg ($|msg| \leq 10$)
- Action (with parameters):
  - $turn, dash, kick, tackle, say, [catch], \ldots$

The RoboCup 2D domain - model (cont'd)

- Transition model: game rules, physical laws with noise
- Observation model: noise and hidden information
- Key feature:
  - Abstraction made by the simulator
  - High-level planning, learning and cooperation
  - No need to handle robot hardware issues
- Key challenges:
  - Fully distributed multi-agent stochastic system
  - Continuous state, observation and action spaces
- Demonstration - before Q&A session

WrightEagle 2D soccer simulation team

- Have participated in RoboCup 2D, since 2000
- 5 world champions: 2006, 2009, 2011, 2013 and 2014
- Have been the main contributor since 2007
- Key components:
  1. Belief update via particle filtering (Bai et al., 2012a,c)
  2. Hierarchical online planning (Bai et al., 2012a,b, 2013b)
  3. Monte-Carlo planning (Bai et al., 2013a, 2014b)
  4. Multi-agent decision-making (Bai et al., 2011, 2012c)

Belief update via particle filtering

- Particle filter based self-localization and multi-object tracking (Bai et al., 2012a,c)
- Belief state is useful in:
  1. Information gathering
  2. State estimation
  3. Probability estimation
  4. Future state prediction

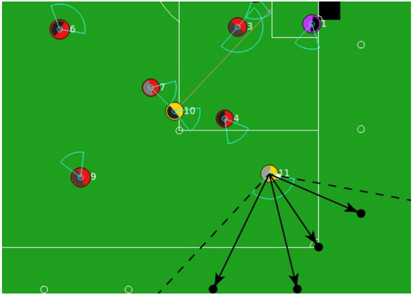


Figure 2 : Local views.

Belief state via particle filtering - example

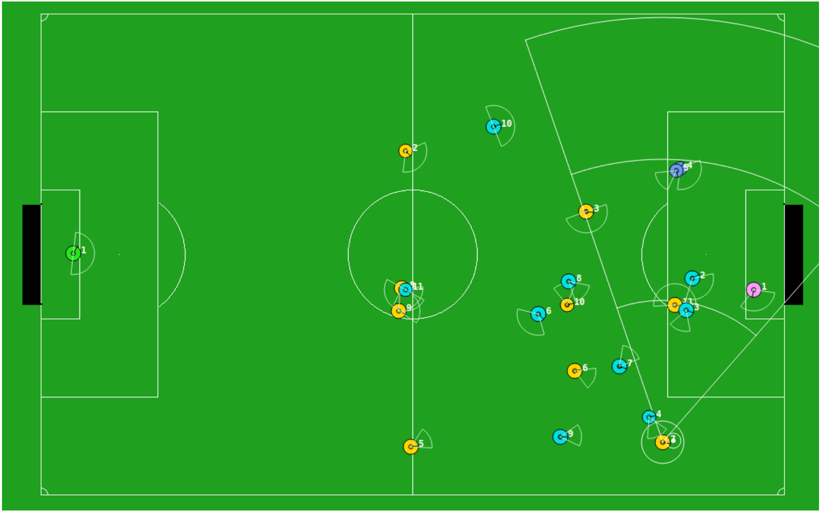


Figure 3 : Unobservable real state.

Belief state via particle filtering - example (cont'd)

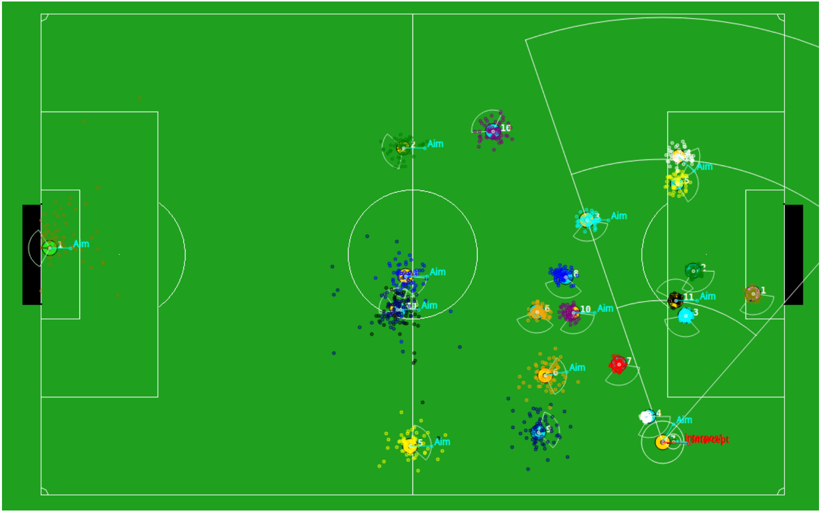


Figure 4 : Updated belief state of player #7.

Multi-object tracking in joint belief space

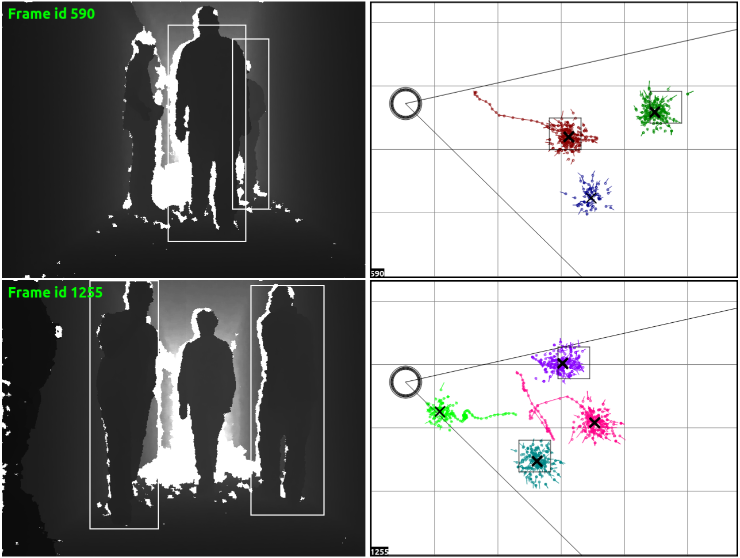


Figure 5 : Multi-object tracking in joint belief space (Bai et al., 2014a).

Hierarchical online planning

- Decision tree:

```
PlanAttack() {
...
if should_shoot then
  return PlanShoot()
else if should_pass then
  return PlanPass()
else
  return PlanDrrible()
...
}
```

- Hierarchical online planning:

```
PlanAttack() {
...
shoot ← PlanShoot()
pass ← PlanPass()
dribble ← PlanDrrible()
...
return max{shoot, pass,
           dribble, ...}
}
```

Hierarchical task graph in WrightEagle

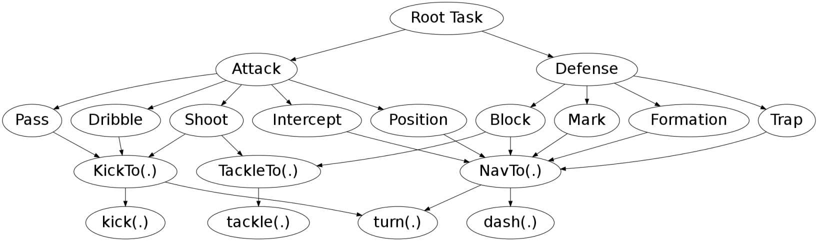


Figure 6 : Hierarchical structure in WrightEagle (Bai et al., 2012b).

Hierarchical online planning - example

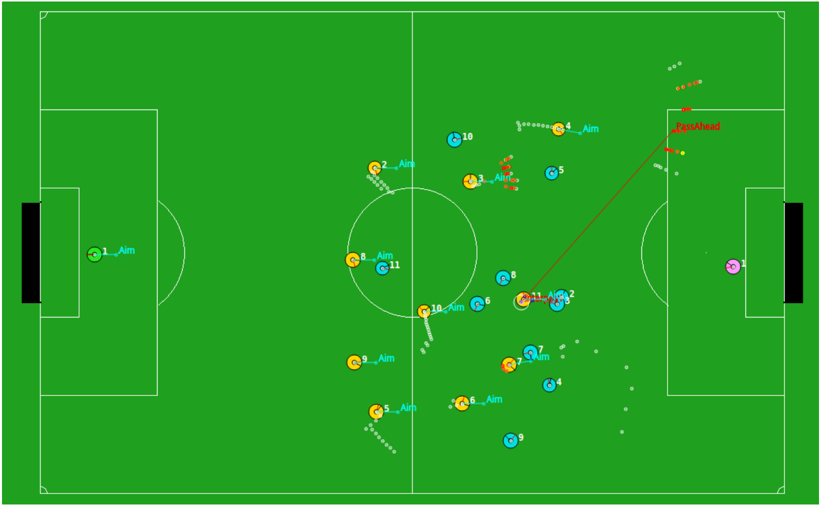


Figure 7 : Hierarchical planning of pass behavior.

Heuristic search in action space

- Efficiently search in huge (macro-)action spaces
  - Enumeration is impossible and not necessary
  - Behavior dependent: hill climbing, best-first-search, pruning, ...

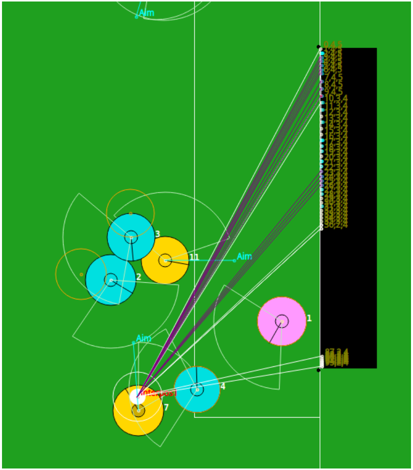


Figure 8 : Search in shoot.

Monte-Carlo planning

- Explicit transition matrix $\Pr(s' \mid s, a)$ is unavailable
- State sampling rules $s' \sim \Pr(s' \mid s, a)$ given by the simulator
- Monte-Carlo tree search (Bai et al., 2013a, 2014b)
- Low-level skills: $NavTo$, $KickTo$, ...
- Embedded in the overall hierarchical framework

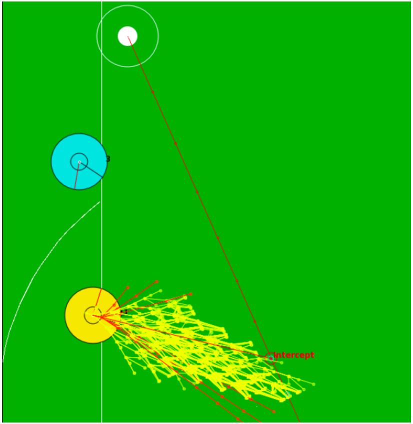


Figure 9 : Search tree in *NavTo*.

Multi-agent decision-making

- Formation and role system
  - Formation: $\Pr(x_1, y_1, \ldots, x_{22}, y_{22} \mid x_b, y_b)$
  - Role classification: forward, midfielder, defender
  - Task allocation (particularly in defense behavior)
- Plan for the team
  1. Pass the ball to teammate t
  2. Recursively plan t's future actions after receiving the ball
  3. Evaluate the pass behavior
- Communicate whenever possible
  1. Share information
  2. Propose future plans
  3. Emergence

Outline

1. Introduction to RoboCup 2D
2. Hierarchical Online Planning
3. Bayesian Monte-Carlo Planning
4. Summary

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
  - Local policy: $\pi_i : S_i \to A_i$
  - Solving M_0 solves the original MDP M
- Focus on undiscounted and goal-directed MDPs: $\gamma = 1$

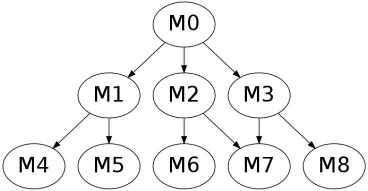


Figure 10 : MAXQ hierarchy.

MAXQ hierarchical decomposition (cont.)

- Hierarchical policy
  - $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$
    - A set of local policies for each subtask
  - Recursively optimal policy π^*
    - Each subtask is optimal given the policies of its descendants
    - Local optimality following the tree structure
  - Contribution: MAXQ-OP approximately finds π^* online (Bai et al., 2012b)

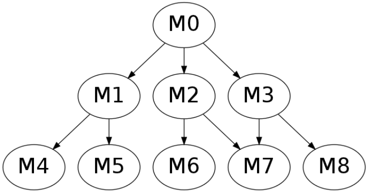


Figure 11 : MAXQ hierarchy.

Recursively optimal policy

- Value function V^* of π^* satisfies

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

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

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

- π^* satisfies

$$\pi_i^*(s) = \operatorname*{argmax}_{a \in A_i} Q^*(i,s,a) \tag{4}$$

Completion function approximation

- Computing completion function implies solving the entire problem
- Introduce terminating distribution

$$\Pr(s' \mid s, a) = \sum_{N} \Pr(s', N \mid s, a) \tag{5}$$

- Rewrite complete function as

$$C^*(i, s, a) = \sum_{s'} \Pr(s' \mid s, a) V^*(i, s') \tag{6}$$

- Approximate $\Pr(s' \mid s, a)$ either online or offline
  1. Offline: $NavTo$ terminates at its target with probability 1
  2. Online: terminating distributions of $Intercept$, $Pass$, $Shoot$, ...

Main structure of MAXQ-OP

- For non-primitive subtasks

$$V^*(i,s) \approx \max_{a \in A_i} \left\{ V^*(a,s) + \sum_{s'} \Pr(s' \mid s,a) V^*(i,s') \right\} \quad (7)$$

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

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geq D[i] \\ \max_{a \in A_i} \{ V(a,s,d) + \\ \sum_{s'} \Pr(s' \mid s,a) V(i,s',d[i] \leftarrow d[i]+1) \} & \text{otherwise} \end{cases} \quad (8)$$

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

Comparing to traditional online search algorithms

- Traditional online search algorithms
  - Search only in state space
  - Search path:

$$[s_1 \to s_2 \to s_3 \to \cdots \to s_H] \rightsquigarrow g \tag{9}$$

- MAXQ-OP algorithm
  - Search both in task hierarchy and state space
  - Search path:

$$\begin{aligned}[s_1 \to \cdots \to s_{H_1}] \rightsquigarrow [g_1/s'_1 \to \cdots \to s'_{H_2}] \rightsquigarrow \\ [g_2/s''_1 \to \cdots \to s''_{H3}] \cdots \rightsquigarrow g\end{aligned} \tag{10}$$

- MAXQ-OP can search deeper by utilizing the task hierarchy

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 to succeed
- Probability of 0.2 to fail

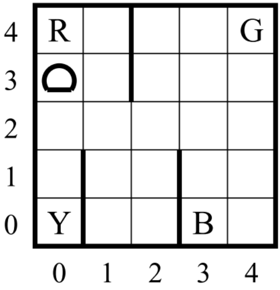


Figure 12 : Taxi domain.

Empirical results in the Taxi domain

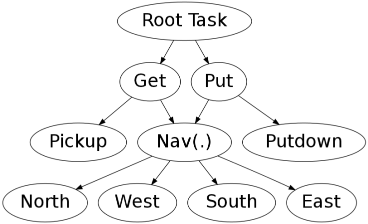


Figure 13 : Task graph for Taxi.

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.

Outline

Monte-Carlo Planning

- Impossible to give explicit transition models for large domains
  - Rules of chess (32 pieces, 64 squares, ∼100 moves)
  - 100 000 000 000 000 000 000 000 000 000 000 000 000 pages (Russell, 2013)
- Easy to have a Bayesian network
  - Sampling rules $s' \sim T(s' \mid s, a)$
  - A simulator for planning problem
- Monte-Carlo tree search (MCTS)
  - Online planning algorithm
  - Build a best-first search tree
  - Take advantage of Monte-Carlo simulations
- Contributions: Posterior sampling approaches to MCTS for MDPs and POMDPs (Bai et al., 2013a, 2014b)

MCTS procedure

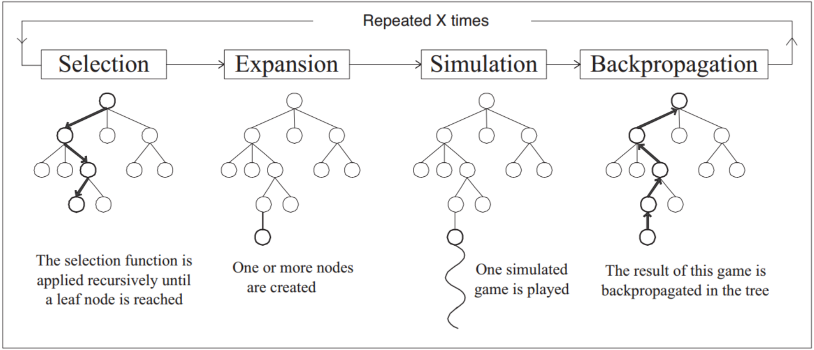


Figure 14 : Outline of Monte-Carlo tree search (Chaslot et al., 2008).

- Rollout policy + Tree policy → Constantly improving policy

Resulting asymmetric search tree

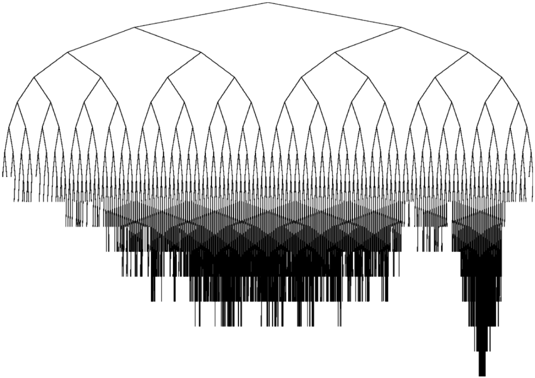


Figure 15 : Asymmetric search tree (Coquelin & Munos, 2007).

Multi-armed bandit problem

- Multi-armed bandit:
  - N slot machines
  - Stochastic rewards
  - Unknown distributions
- Cumulative regret (CR):

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T}(X_{a^*} - X_{a_t})\right] \quad (11)$$

- Policy
  - Action-reward history $\rightarrow$ Action
  - Optimal policy: minimize CR

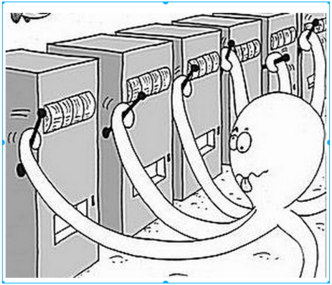


Figure 16 : MAB.

The *exploration vs. exploitation* dilemma

- A fundamental problem in MAB (also in MCTS):
  1. Must not only select the action that currently seems best
  2. Should also keep exploring for possible higher future outcomes
- Upper confidence bound (UCB) heuristic:

$$\mathrm{UCB}(a) = \bar{R}(a) + c\sqrt{\frac{\log T}{N(a)}}, \tag{12}$$

  where:
  1. $\bar{R}(a)$ is the mean reward of applying action a
  2. T is the total times of acting so far
  3. $N(a)$ is the times of performing action a
  4. c is the exploration constant
- Asymptotic optimality in MABs

UCT algorithm

- Upper confidence over trees (UCT) (Kocsis & Szepesvári, 2006):

$$\mathrm{UCB}(s,a) = \bar{Q}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \tag{13}$$

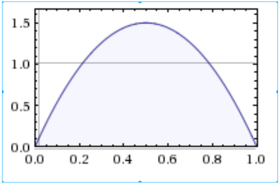
- Find the best action for root node with probability 1
- With suitable choice of c
- No principled ways to determine c

Thompson sampling

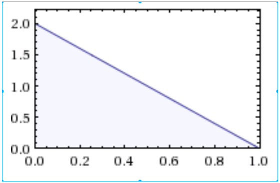
- Select an action based on its posterior probability of being optimal (Thompson, 1933)
  - 2-armed bandit: a* and b
  - Initially: 50% vs. 50%
  - Normally: 60% vs. 40%
  - Finally: 100% vs. 0%
- Can efficiently be approached by sampling method
  1. Z: the observed history
  2. θ_a: the hidden parameters of the distribution of X_a
  3. Poster distribution of θ_a: $\Pr(\theta_a \mid Z)$
  4. Sample a set of hidden parameters $\theta_a \sim \Pr(\theta_a \mid Z)$
  5. Select the action with highest expectation $\mathbb{E}\left[X_a \mid \theta_a\right]$

An example of Thompson sampling

- 2-armed bandit: a and b
- Bernoulli reward distributions
- Hidden parameters p_a and p_b
- Prior distributions:
  1. $p_a \sim Uniform(0, 1)$
  2. $p_b \sim Uniform(0, 1)$
- History: a, 1, b, 0, a, 0, ?
- Posterior distributions:
  1. $p_a \sim Beta(2, 2)$
  2. $p_b \sim Beta(1, 2)$
- Sample p_a and p_b
- Compare $\mathbb{E}[X_a \mid p_a]$ and $\mathbb{E}[X_b \mid p_b]$



(a) $Beta(2, 2)$.



(b) $Beta(1, 2)$.

Figure 17 : Posteriors.

Motivation

- Thompson sampling
  1. Theoretically achieves asymptotic optimality
  2. Empirically outperforms UCB
  3. Utilize more informative models in terms of prior knowledge
- Basic idea for our methods:
  1. Model the distribution of action reward
  2. Update the posterior reward distribution
  3. Use Thompson sampling to guide the action selection

DNG-MCTS algorithm

- DNG-MCTS: Dirichlet-NormalGamma MCTS (Bai et al., 2013a)
- $X_{s,\pi}$: the cumulative reward of following policy π starting from state s
- $X_{s,a,\pi}$: the cumulative reward of first performing action a in state s and following policy π thereafter
- By definition:

$$X_{s,a,\pi} = R(s,a) + \gamma X_{s',\pi}, \tag{14}$$

where $s' \sim T(s' \mid s, a)$

DNG-MCTS algorithm (cont'd)

- Basic assumptions:
  1. $X_{s,\pi}$ follows a Normal distribution (CLT on Markov chains)
  2. $X_{s,a,\pi}$ follows a mixture of Normal distributions
- Bayesian modelling and inference:
  1. $X_{s,\pi} \sim \mathcal{N}(\mu_s, 1/\tau_s)$;
     $(\mu_s, \tau_s) \sim NormalGamma(\mu_{s,0}, \lambda_s, \alpha_s, \beta_s)$
  2. $T(\cdot \mid s, a) \sim Dirichlet(\boldsymbol{\rho}_{s,a})$
- Action selection: Thompson sampling
- Find the best action for the root node with probability 1

DNG-MCTS - experiments

- MDP benchmark problems (cost based):
  1. Canadian traveler problem
  2. Race track problem
  3. Sailing problem
- Evaluation:
  1. Run from current state for a number of iterations
  2. Apply the best action according to the returned values
  3. Repeat the loop until termination conditions
  4. Report the total discounted cost

Canadian traveler problem

- A path finding problem
- Imperfect information
- Edges may be blocked with given prior probabilities
- Modeled as an MDP
- State space size: $n \times 3^m$

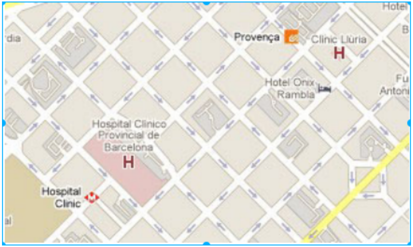


Figure 18 : CTP.

Canadian traveler problem (cont'd)

Table 2 : CTP problems with 20 nodes.

		random rollout policy		optimistic rollout policy	
ins.	#state	UCT	DNG	UCT	DNG
20-1	20×3^{49}	216.4±3	223.9±4	180.7±3	177.1±3
20-2	20×3^{49}	178.5±2	178.1±2	160.8±2	155.2±2
20-3	20×3^{51}	169.7±4	159.5±4	144.3±3	140.1±3
20-4	20×3^{49}	264.1±4	266.8±4	238.3±3	242.7±4
20-5	20×3^{52}	139.8±4	133.4±4	123.9±3	122.1±3
20-6	20×3^{49}	178.0±3	169.8±3	167.8±2	141.9±2
20-7	20×3^{50}	211.8±3	214.9±4	174.1±2	166.1±3
20-8	20×3^{51}	218.5±4	202.3±4	152.3±3	151.4±3
20-9	20×3^{50}	251.9±3	246.0±3	185.2±2	180.4±2
20-10	20×3^{49}	185.7±3	188.9±4	178.5±3	170.5±3
total		2014.4	1983.68	1705.9	1647.4

Race track problem

- A set of initial states
- Move towards the goal
- Accelerate in one of the eight directions
- Probability of 0.9 to succeed
- Probability 0.1 to fail
- State space size: 22,534

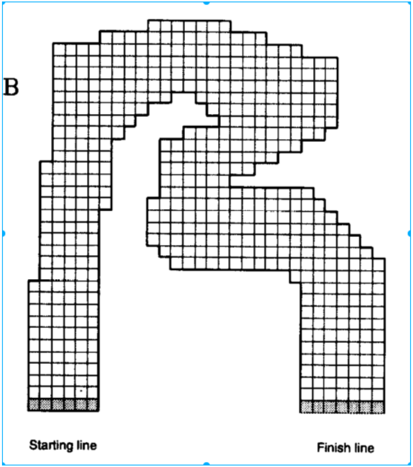


Figure 19 : Race track.

Race track problem (cont'd)

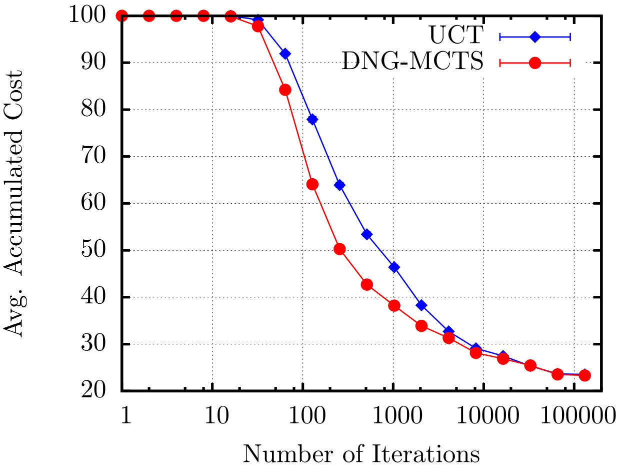


Figure 20 : Racetrack-barto-big with random policy.

Sailing problem

- A sailboat navigates in a 100×100 grid world
- Direction of the wind changes over time
- Move to a neighbor grid at each localization
- Reach the destination as fast as possible
- State space size: 80,000

Figure 21 : Sailing boat.

Sailing problem (cont'd)

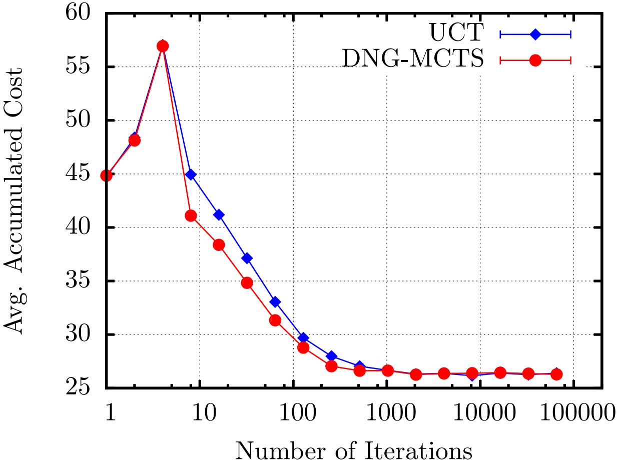


Figure 22 : Sailing-100×100 with random policy.

Extend to POMDPs

- POMCP (Silver & Veness, 2010):

$$\mathrm{UCB}(h, a) = \bar{Q}(h, a) + c\sqrt{\frac{\log N(h)}{N(h, a)}} \tag{15}$$

- Search in the tree of histories

D^2NG-POMCP algorithm

- D^2NG-POMCP: Dirichlet-Dirichlet-NormalGamma partially observable Monte-Carlo planning (Bai et al., 2014b)
- $X_{b,a}$: the immediate reward of performing action a in belief b
- $X_{s,b,\pi}$: the cumulative reward of following policy π from $\langle s, b \rangle$
- $X_{b,\pi}$: the cumulative reward of following policy π in belief b
- By definition:

$$\Pr(X_{b,a} = r) = \sum_{s \in S} \mathbf{1}[R(s,a) = r] b(s), \tag{16}$$

$$f_{X_{b,\pi}}(x) = \sum_{s \in S} b(s) f_{X_{s,b,\pi}}(x) \tag{17}$$

D^{2}NG-POMCP algorithm (cont'd)

- Basic assumptions:
  1. $X_{b,a}$ follows a Multinomial distribution
  2. $X_{s,b,\pi}$ follows a Normal distribution (CLT on Markov chains)
  3. $X_{b,\pi}$ follows a mixture of Normal distributions
- Bayesian modelling and inference:
  1. $X_{b,a} \sim Multinomial(\boldsymbol{p}_{b,a})$; $\boldsymbol{p}_{b,a} \sim Dirichlet(\boldsymbol{\psi}_{b,a})$
  2. $X_{s,b,\pi} \sim \mathcal{N}(\mu_{s,b}, 1/\tau_{s,b})$;
     $(\mu_{s,b}, \tau_{s,b}) \sim NormalGamma(\mu_{s,b,0}, \lambda_{s,b}, \alpha_{s,b}, \beta_{s,b})$
  3. $\Omega(\cdot \mid b, a) \sim Dirichlet(\boldsymbol{\rho}_{b,a})$
- Action selection: Thompson sampling
- Find the best action for the root node with probability 1

Experiments

- POMDP benchmark problems:
  1. RockSample problem
  2. PocMan problem
- Evaluation:
  1. Run the algorithms for a number of iterations for current belief
  2. Apply the best action based on the resulting action-values
  3. Repeat until termination conditions
  4. Report the total discounted reward

RockSample problem

- Rover exploration
- Navigate in a grid world
- Sample rocks
- Noisy sensors
- RockSample[7,8]
  1. 12,545 states
  2. 13 actions
  3. 2 observations

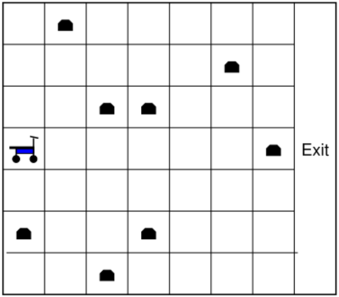


Figure 23 : RockSample[7,8].

RockSample problem (cont'd)

Table 3 : Comparison in RockSample (given exactly 1 second per action).

RockSample	[7, 8]	[11,11]	[15,15]
States $\|s\|$	12,544	247,808	7,372,800
AEMS2	21.37 ± 0.22	N/A	N/A
HSVI-BFS	21.46 ± 0.22	N/A	N/A
SARSOP	21.39 ± 0.01	21.56 ± 0.11	N/A
POMCP	20.71 ± 0.21	20.01 ± 0.23	15.32 ± 0.28
D^2NG-POMCP	20.87 ± 0.20	21.44 ± 0.21	20.20 ± 0.24

PocMan problem

- PocMan finding food
- 17×19 maze world
- 4 ghosts roaming
- Die when touching ghosts
- Size:
  1. 10^{56} states
  2. 4 actions
  3. 1,024 observations

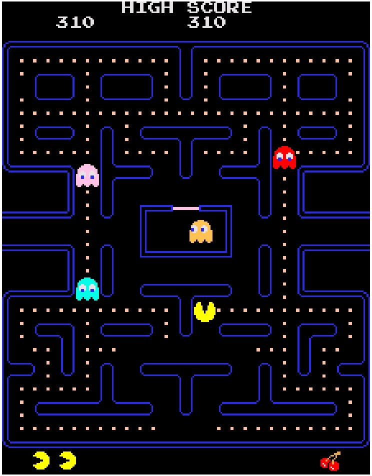
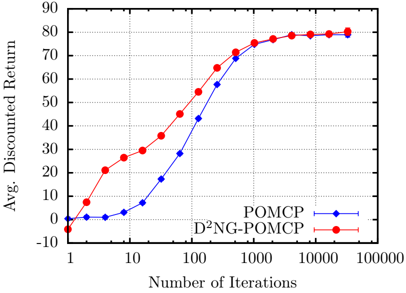
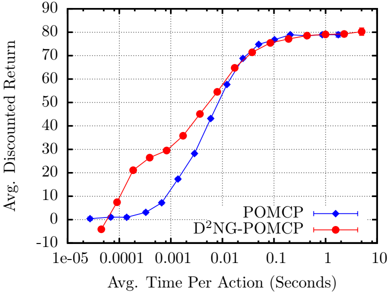


Figure 24 : PocMan.

PocMan problem (cont'd)



(a) PocMan.



(b) PocMan.

Figure 25 : Performance of D^{2}NG-POMCP in PocMan.

Outline

Summary

1. RoboCup soccer simulation 2d domain
   - Fully-distributed multi-agent stochastic system
   - Continuous state, observation and action spaces
2. WrightEagle soccer simulation team
   - Planning and sensing in belief space
   - Utilizing MAXQ hierarchical structure
   - Various heuristic and Monte-Carlo techniques
3. Hierarchical online planning — MAXQ-OP
   - Exploit MAXQ hierarchical structure online
   - Completion function approximation
4. Bayesian Monte-Carlo tree search — DNG-MCTS and D^2NG-POMCP
   - Maintain posterior distributions of action rewards
   - Select an action according to its probability of being optimal

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., Lu, G., Zhang, H., & Chen, X. (2011). WrightEagle 2D soccer simulation team description 2011. In *RoboCup Soccer Simulation 2D Competition, Istanbul, Turkey*.

Bai, A., Simmons, R., Veloso, M., & Chen, X. (2014a). Intention-aware multi-human tracking for human-robot interaction via particle filtering over sets. In *AAAI Fall Symposium: AI for Human-Robot Interaction (AI-HRI 2014)*. Arlington, United States.

Bai, A., Wu, F., & Chen, X. (2012b). Online planning for large MDPs with MAXQ decomposition. 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 (NIPS 2013)*, (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., 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.

Bai, A., Zhang, H., Lu, G., Jiang, M., & Chen, X. (2012c). WrightEagle 2D soccer simulation team description 2012. In *RoboCup Soccer Simulation 2D Competition, Mexico City, Mexico*.

Chaslot, G., Bakkes, S., Szita, I., & Spronck, P. (2008). Monte-carlo tree search: A new framework for game AI. In C. Darken, & M. Mateas (Eds.) *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, October 22-24, 2008, Stanford, California, USA*. The AAAI Press. URL http://www.aaai.org/Library/AIIDE/2008/aiide08-036.php

Coquelin, P.-A., & Munos, R. (2007). Bandit algorithms for tree search. *arXiv preprint 0703062*.

References II

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

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, (pp. 282–293).

Russell, S. (2013). Unifying logic and probability; a new dawn for ai? Presented at Colloquium Sorbonne-Universits.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, (pp. 2164–2172).

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*, 285–294.