Building Intelligent Agents via Decision-Theoretic Planning

Aijun Bai

April 4, 2016

UC Berkeley

Background

Sequential Decision-Making

- A fundamental task faced by any intelligent agent
  - Agent: autonomous system — software/robot/app
- The question of "*What should I do now?*"
  - I: the automated planning/learning agent
  - now: current state/belief of the environment
  - do: one of available actions to execute
  - should: maximization of long-term rewards

Examples

- Computer Go programs
  - Where to place the next stone?
- Mobile robots
  - Localize? Navigate? Manipulate? ...
- Autonomous cars
  - Accelerate? Brake? Change lanes? ...
- Robotic soccer players
  - Position? Intercept? Pass? Dribble? Shoot? ...

Framework

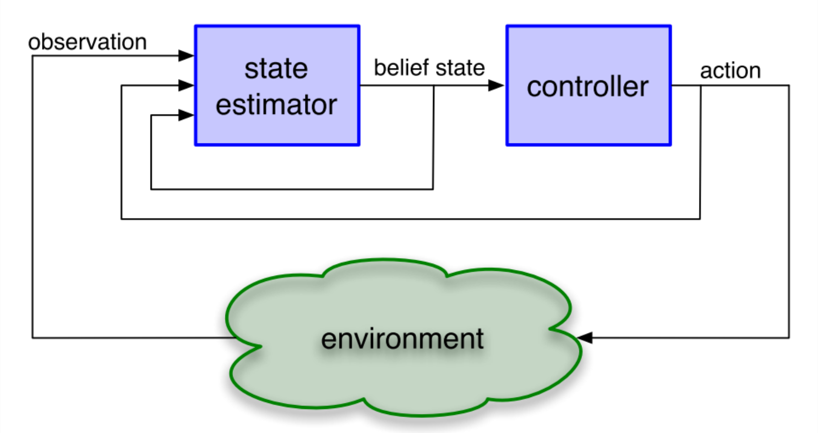


Figure 1: Agent & environment

Models

- Uncertainty
  - Transition $\Pr(s' \mid s, a)$
  - Observation $\Pr(o \mid s)$

Table 1: Sequential decision-making under uncertainty

	Not-controlled	**Controlled**	**Multi-agent**	**Game-theoretic**
Fully observable	Markov Chain	MDP	Dec-MDP	Markov Game
Partially observable	HMM	POMDP	Dec-POMDP	POSG

Markov Decision Processes

- MDP models fully observable domains:
  1. State space: $S = \left\{s_1, s_2, \ldots, s_{|S|}\right\}$
  2. Action space: $A = \left\{a_1, a_2, \ldots, a_{|A|}\right\}$
  3. Transition function: $T(s' \mid s, a) \to [0, 1]$
  4. Reward function: $R(s, a) \to \mathbb{R}$
- Policy: $\pi : S \to A$
- Value function: $V^\pi(s_0) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t R(s_i, \pi(s_i))\right]$
- Bellman optimality:

$$V^*(s) = \max_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a) V^*(s') \right\} \tag{1}$$

- Optimal policy:

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \, V^*(s) \tag{2}$$

Partially Observable MDPs

- POMDP extends MDP to partially observable domains:
    1. Observation space: $O = \left\{o_1, o_2, \ldots, o_{|O|}\right\}$
    2. Observation function: $\Omega(o \mid a, s) \rightarrow [0, 1]$
- History: $h = (a_0, o_1, a_1, o_2, \ldots a_{t-1}, o_t)$
- Belief state: $b(s) = \Pr(s \mid b_0, h)$
- Belief space: $\mathcal{B} = \{b\}$
- Policy: $\pi : \mathcal{B} \rightarrow A$

Solving POMDPs

- Belief update: $b' = \zeta(b, a, o)$, written as:

$$b'(s') = \eta\Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s) \tag{3}$$

- Bellman equation:

$$V^*(b) = \max_{a \in A} \left\{ r(b, a) + \gamma \sum_{o \in O} \Omega(o \mid b, a) V^* \left(\zeta(b, a, o)\right) \right\} \tag{4}$$

- Optimal policy:

$$\pi^*(b) = \underset{a \in A}{\operatorname{argmax}}\, V^*(b) \tag{5}$$

Approaches

- Planning
  - Having a model of the environment
  - Solve the model offline/online
    - Offline planning: dynamic programming
    - Online planning: search/Monte-Carlo simulation
  - Act as suggested by the found policy
- Reinforcement learning
  - Learn to act by interacting with the environment
    - Model-free RL/Model-based RL/Simulated RL

Case Study: Simulated Robotic Soccer

RoboCup Soccer Simulation 2D

- Simulated soccer game
- 11 players for each team
- Independently controlled
- In each cycle (100ms)
    - Receive observation
    - Make decision
    - Send action(s)
- Normally 6,000 cycles

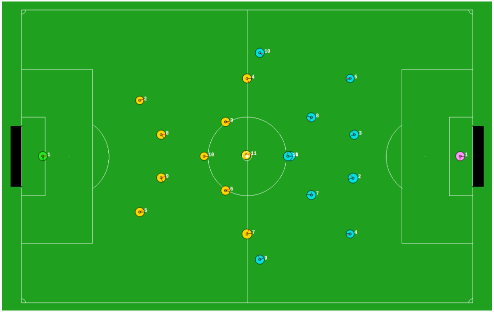
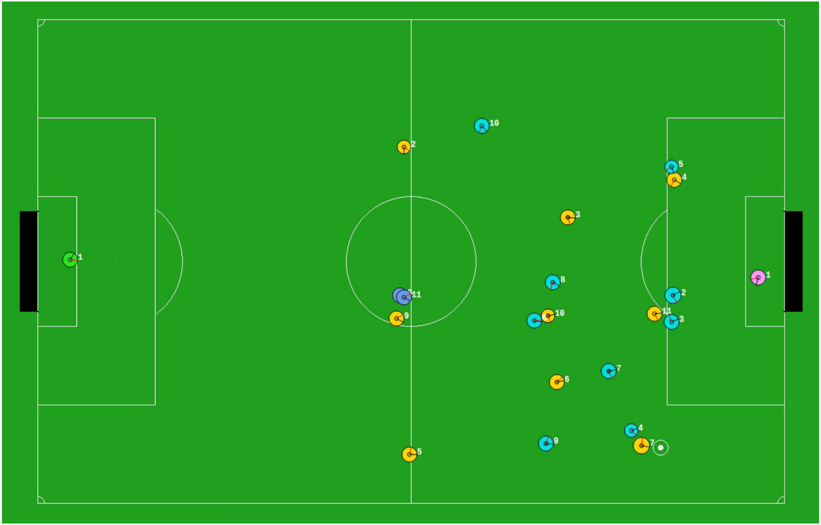


Figure 2: RoboCup 2D

RoboCup 2D in Action



The Model

- State:
  - The ball and 22 players
- Observation:
  - Noisy visual information (within field of view):
    - The ball, players, lines, corners, ...
  - Random hearing information: msg ($|msg| \leq 10$)
- Parametric actions:
  - $turn, dash, kick, tackle, say, [catch]$

The Model (cont'd)

- Transition function: game rules, simulated physical world
- Observation function: noisy perception with hidden information
- Key features:
  - 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

WrightEagle 2D Soccer Simulation Team

- A team of autonomous agents for RoboCup 2D
- Have been participating in RoboCup competitions since 2000
- Have been the main contributor from 2007 to 2014
- 6 world champions: 2006, 2009, 2011, 2013, 2014 and 2015
- Key components:
    1. Belief update via particle filtering (Bai et al., 2012a,c)
    2. Hierarchical online planning (Bai et al., 2012a,b, 2013b, 2015)
    3. Monte-Carlo planning (Bai et al., 2013a, 2014)
    4. Multi-agent decision-making (Bai et al., 2011, 2012c)

Belief Update via Particle Filtering

- Particle filter based self-localization and multi-object tracking
- Belief state is used for:
    1. State estimation
    2. Information gathering

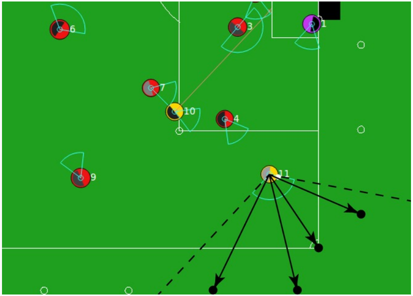


Figure 4: Localization

Belief Update via Particle Filtering - Example

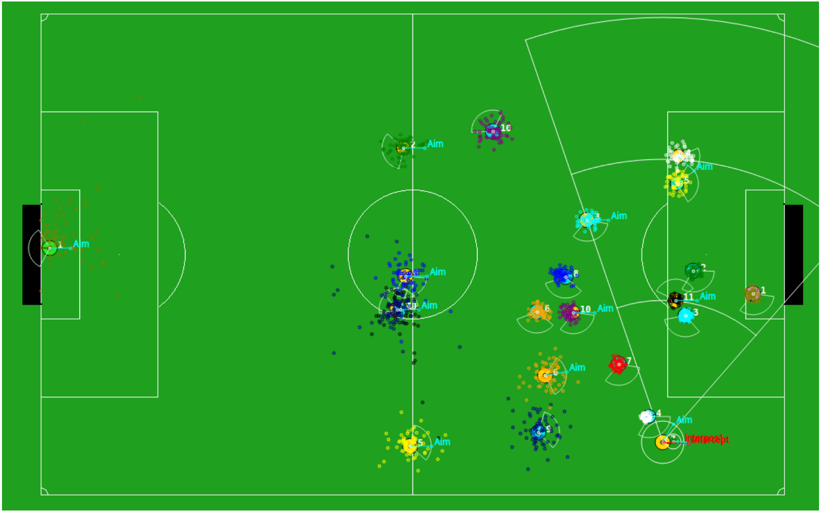


Figure 5: Updated belief state of player #7

Belief State Visualization

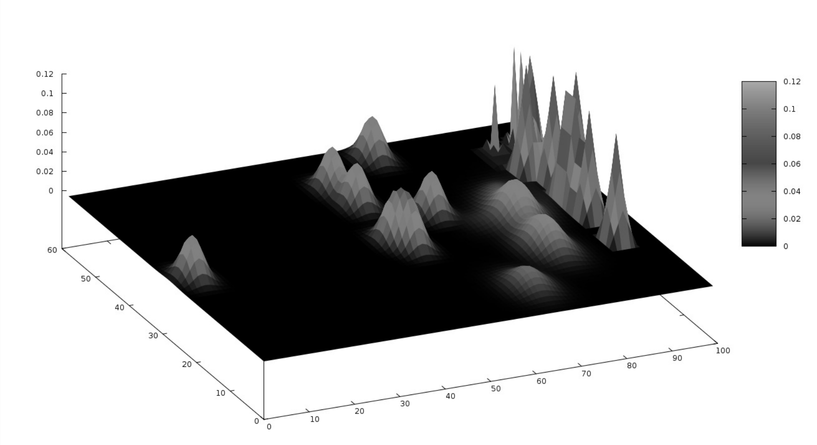


Figure 6: Belief state in terms of player position distributions

Hierarchical Online Planning

- Rule-based system:

  PlanAttack() {

  ...

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

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

  else
  - **return** PlanDrrible()

  ...

  }

- Hierarchical planning:

  PlanAttack() {

  ...

  shoot ← PlanShoot()

  pass ← PlanPass()

  dribble ← PlanDrrible()

  ...

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

  }

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 $\pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$

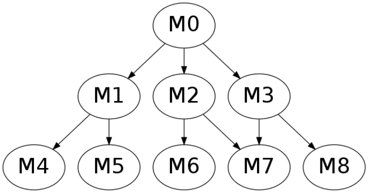


Figure 7: MAXQ hierarchy

MAXQ-based Task Graph in WrightEagle

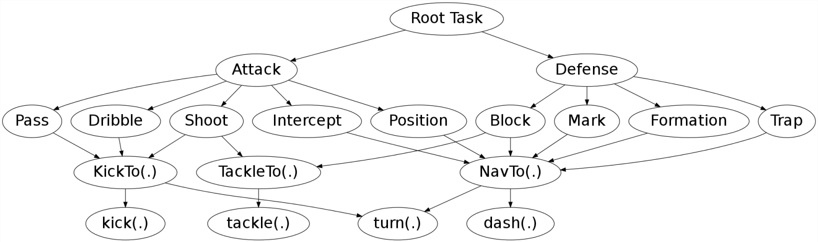


Figure 8: Hierarchical structure in WrightEagle (Bai et al., 2012b)

Hierarchical Value Function Decomposition

- Value function V^* of π^* satisfies

$$V^*(i,s) = \begin{cases} R(s,i) & \text{if } M_i \text{ is primitive} \\ \max_{a\in A_i} Q^*(i,s,a) & \text{otherwise} \end{cases} \tag{6}$$

$$Q^*(i,s,a) = V^*(a,s) + C^*(i,s,a) \tag{7}$$

$$C^*(i,s,a) = \sum_{s',N} \Pr(s',N \mid s,a) V^*(i,s') \tag{8}$$

- π^* satisfies

$$\pi_i^*(s) = \operatorname*{argmax}_{a\in A_i} Q^*(i,s,a) \tag{9}$$

MAXQ-OP (Bai et al., 2012b)

- Approximate $\Pr(s', N \mid s, a)$ either online or offline
- For non-primitive subtasks

$$V^*(i,s) \approx \max_{a \in A_i} \left\{ V^*(a,s) + \sum_{s'} \Pr(s' \mid s,a) V^*(i,s') \right\} \quad (10)$$

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

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geq D[i] \\ \max_{a \in A_i} \{ V(a,s,d) + \\ \sum_{s'} \Pr(s' \mid s,a) V(i,s',d[i] \leftarrow d[i]+1) \} & \text{otherwise} \end{cases} \quad (11)$$

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

MAXQ-OP in WrightEagle

- Task evaluation over hierarchy
    - Value function decomposition
- Terminating distribution approximation
    - Success and failure probabilities
- Search/Monte-Carlo planning
- Heuristic function

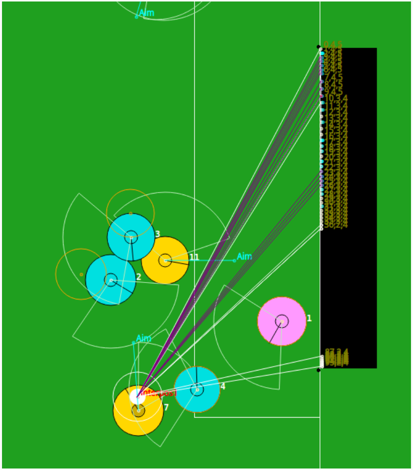


Figure 9: Search in shoot

Value Function Decomposition in WrightEagle

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

$$V^*(\text{Root}, \boldsymbol{s}) = \max\{Q^*(\text{Root}, \boldsymbol{s}, \text{Attack}), Q^*(\text{Root}, \boldsymbol{s}, \text{Defense})\}, \tag{13}$$

$$\begin{aligned} V^*(\text{Attack}, \boldsymbol{s}) = \max\{&Q^*(\text{Attack}, \boldsymbol{s}, \text{Pass}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Dribble}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Shoot}), \\ &Q^*(\text{Attack}, \boldsymbol{s}, \text{Intercept}), Q^*(\text{Attack}, \boldsymbol{s}, \text{Position})\}, \end{aligned} \tag{14}$$

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

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

$$V^*(\text{Pass}, \boldsymbol{s}) = \max_{\text{position } p} Q^*(\text{Pass}, \boldsymbol{s}, \text{KickTo}(p)), \tag{17}$$

$$V^*(\text{Intercept}, \boldsymbol{s}) = \max_{\text{position } p} Q^*(\text{Intercept}, \boldsymbol{s}, \text{NavTo}(p)), \tag{18}$$

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

$$Q^*(\text{Intercept}, \boldsymbol{s}, \text{NavTo}(p)) = V^*(\text{NavTo}(p), \boldsymbol{s}) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{NavTo}(p)) V^*(\text{Intercept}, \boldsymbol{s}'), \tag{20}$$

$$V^*(\text{KickTo}(p), \boldsymbol{s}) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\text{KickTo}(p), \boldsymbol{s}, \text{kick}(a, \theta)), \tag{21}$$

$$V^*(\text{NavTo}(p), \boldsymbol{s}) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\text{NavTo}(p), \boldsymbol{s}, \text{dash}(a, \theta)), \tag{22}$$

$$Q^*(\text{KickTo}(p), \boldsymbol{s}, \text{kick}(a, \theta)) = R(s, \text{kick}(a, \theta)) + \sum_{s'} P_t(s' \mid \boldsymbol{s}, \text{kick}(a, \theta)) V^*(\text{KickTo}(p), \boldsymbol{s}'), \tag{23}$$

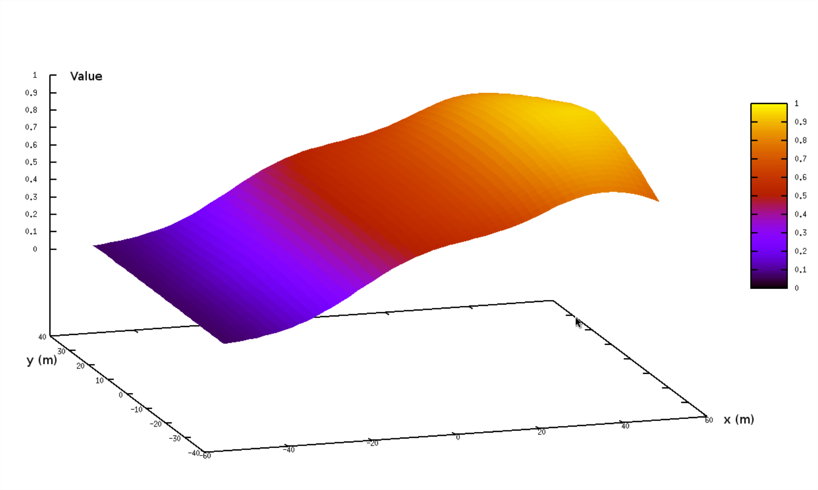


Figure 10: A heuristic function used in defense behaviors

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

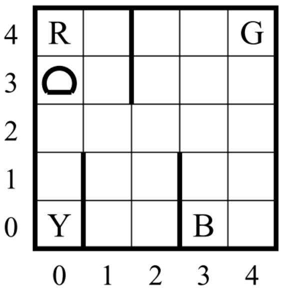


Figure 11: Taxi domain

Empirical Results in the Taxi Domain

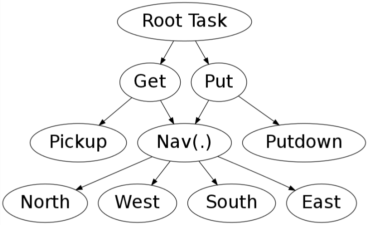


Figure 12: Task graph for Taxi

Table 2: 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.

Monte-Carlo Planning

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

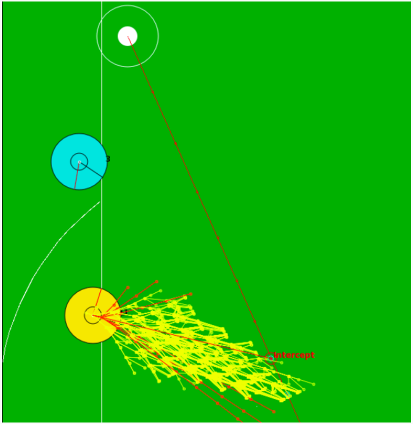


Figure 13: Search tree in *NavTo*

Monte-Carlo Tree Search

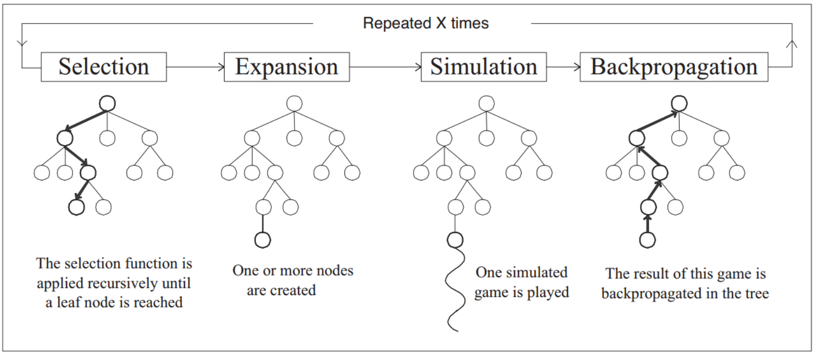


Figure 14: Outline of Monte-Carlo tree search (Chaslot et al., 2008)

- Rollout policy + Tree policy → Constantly improving policy

Resulting Asymmetric Search Tree

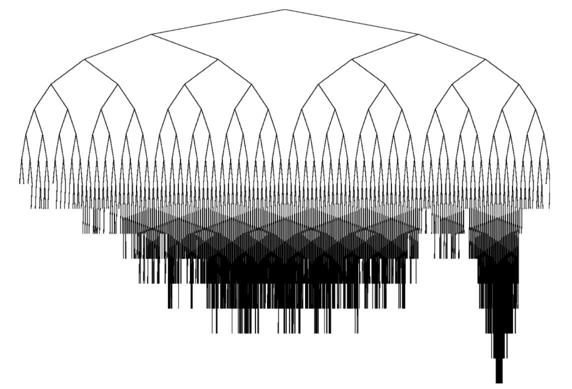


Figure 15: Asymmetric search tree (Coquelin & Munos, 2007)

The Exploration vs. Exploitation Dilemma

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

$$\mathrm{UCB}(s,a) = \bar{Q}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \tag{24}$$

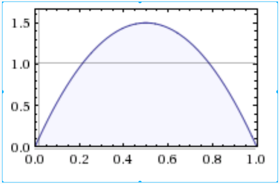
- Find the best action for root node with probability 1
- With suitable choice of c
- No principled ways to determine c

Thompson Sampling

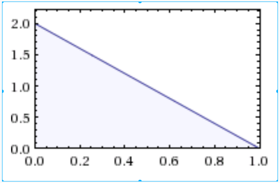
- Select an action based on its posterior probability of being optimal (Thompson, 1933)
- Can efficiently be approached by sampling
    1. θ_a: the hidden parameters of the distribution of X_a
    2. Z: the observed history
    3. Poster distribution of θ_a: $\Pr(\theta_a \mid Z)$
    4. Sample a set 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 actions: 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 16: Posteriors

Motivation

- Thompson sampling
  1. Theoretically achieves asymptotic optimality
  2. Empirically outperforms UCB
  3. Utilize more informative models in terms of prior distributions
- Basic idea for DNG-MCTS (Bai et al., 2013a) and D^2NG-POMCP (Bai et al., 2014)
  1. Model the parametric distribution of action reward
  2. Update the posterior distribution
  3. Use Thompson sampling to select action in MCTS

DNG-MCTS Algorithm

- $X_{s,\pi}$: 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:

$$X_{s,a,\pi} = R(s,a) + \gamma X_{s',\pi}, \tag{25}$$

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

Canadian Traveler Problem

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

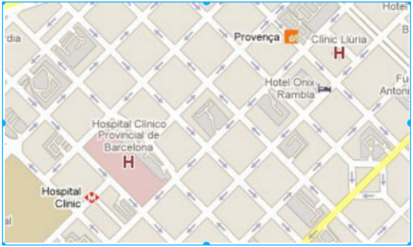


Figure 17: CTP

Canadian Traveler Problem (cont'd)

Table 3: 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

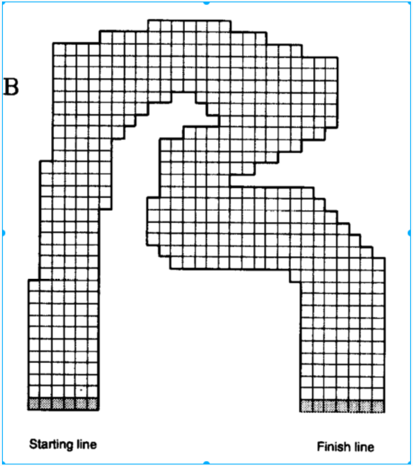


Figure 18: Race track

Race Track Problem (cont'd)

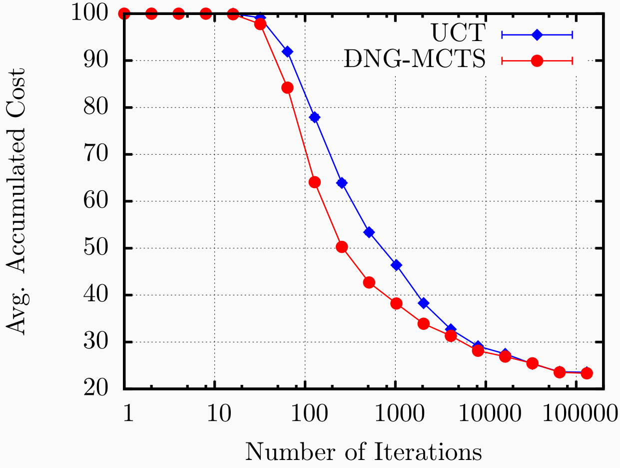


Figure 19: Racetrack-barto-big with random policy

D^2NG-POMCP Algorithm

- $X_{b,a}$: 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:

$$\Pr(X_{b,a} = r) = \sum_{s \in S} \mathbf{1}[R(s,a) = r] b(s), \tag{26}$$

$$f_{X_{b,\pi}}(x) = \sum_{s \in S} b(s) f_{X_{s,b,\pi}}(x) \tag{27}$$

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

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

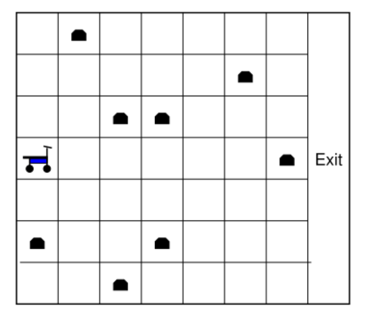


Figure 20: RockSample[7,8]

RockSample Problem (cont'd)

Table 4: Comparison in RockSample (given exactly 1 second per action)

RockSample	[7, 8]	[11,11]	[15,15]
States $\lvert s \rvert$	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^{2}NG-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

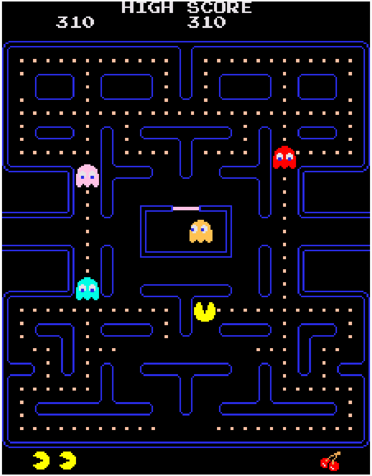
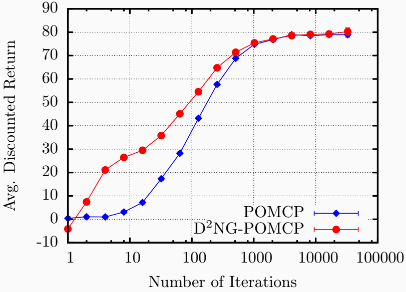


Figure 21: PocMan

PocMan Problem (cont'd)



(a) PocMan **(b)** PocMan

Figure 22: Performance of D^{2}NG-POMCP in PocMan

Multi-Agent Decision-Making

- Formation and role system
  - Teammate formation: $HomePosition_i(x_b, y_b)$
  - Opponent formation: $\Pr(x_1, y_1, \ldots, x_{11}, y_{11} \mid x_b, y_b)$
  - Roles: forward, midfielder, defender, goalie
  - Strategic setplays
- Multi-step planning model
  1. Pass the ball to teammate t
  2. Recursively plan t's future actions in terms of next passing/dribbling/shooting
- Teammate/opponent modelling
  - Rationality assumption

Summary

Summary

1. RoboCup soccer simulation 2d domain
    - Fully-distributed multi-agent stochastic system
    - Continuous state, observation and action spaces
2. WrightEagle soccer simulation team
    - Particle filter based belief update
    - MAXQ hierarchical structure
    - Heuristic search and Monte-Carlo techniques
3. Hierarchical online planning — MAXQ-OP
    - Exploit MAXQ hierarchical structure online
    - Terminating distribution estimation
4. Bayesian Monte-Carlo tree search for MDPs and POMDPs
    - Maintain posterior distributions of action rewards
    - Select an action according to its probability of being optimal

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

References II

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).

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.

References III

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.

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*.

References IV

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.) *AIIDE*. 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 cs/0703062*.

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

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.