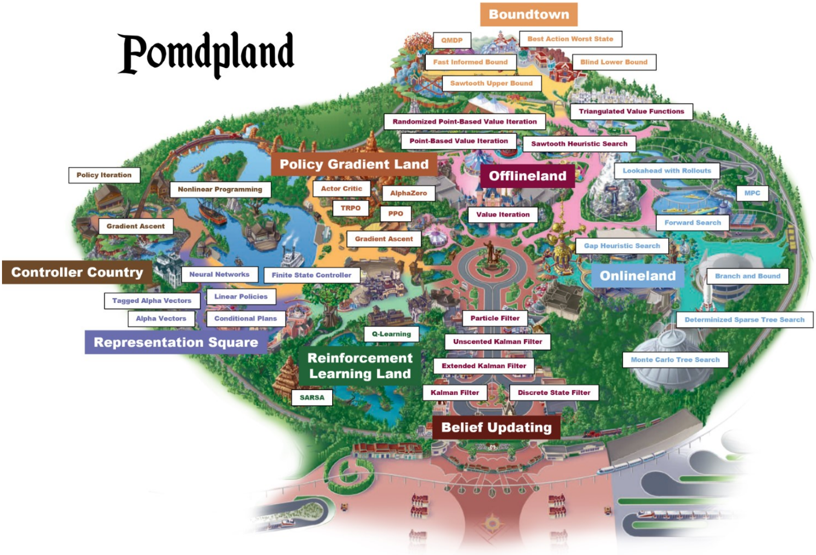
A tour of Pomdpland

Algorithms for sequential decision making under uncertainty

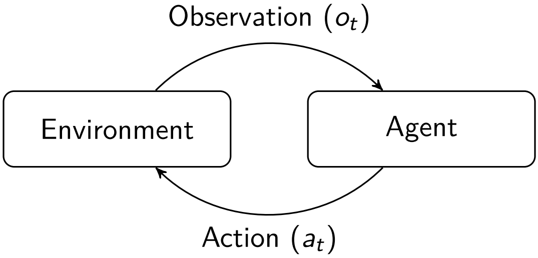
Mykel J. Kochenderfer

Stanford University

August 9, 2022


Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Representation Square
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

POMDP definition



POMDP definition

Symbol	Description	Example
$\mathcal{S}$	states	hungry (h), not hungry ($\neg h$)
$\mathcal{A}$	actions	feed (f), not feed ($\neg f$)
$\mathcal{O}$	observations	crying (c), not hungry ($\neg c$)
$T(s' \mid s, a)$	transition model	$T(h \mid \neg h, \neg f) = 0.2$
$R(s, a)$	reward model	$R(h, f) = -6$
$O(o \mid a, s')$	observation model	$P(c \mid \neg f, h) = 0.9$

POMDP derived functions

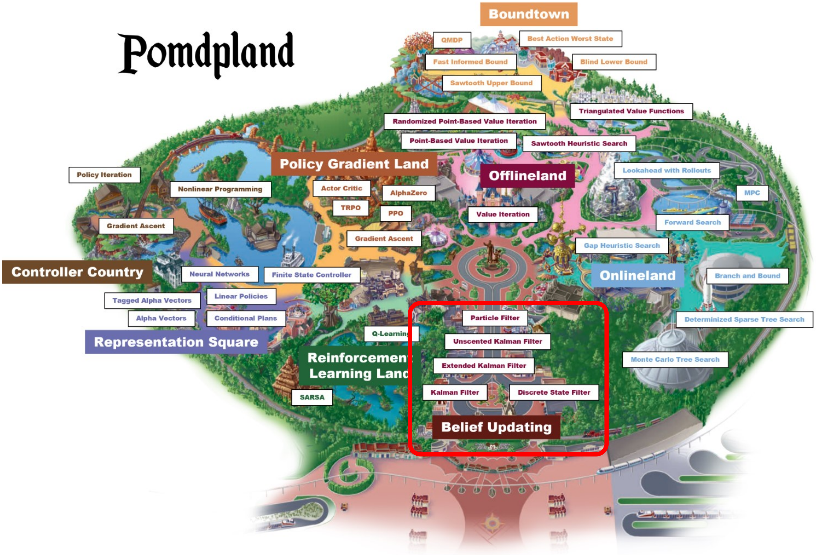
Symbol	Description
$b(s)$	belief
$U(b)$	utility (or value) function
$\pi(b)$	policy

Applications

1. Aircraft collision avoidance
2. Automated driving
3. Breast cancer screening
4. Financial consumption and portfolio allocation
5. Distributed wildfire surveillance
6. Mars science exploration

Contributing disciplines

1. Economics
2. Psychology
3. Neuroscience
4. Computer science
5. Engineering
6. Mathematics
7. Operations research



Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Particle Filter
Alpha Vectors
Conditional Plans
Determinized Sparse Tree Search
Q-Learning
Unscented Kalman Filter
Representation Square
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Beliefs

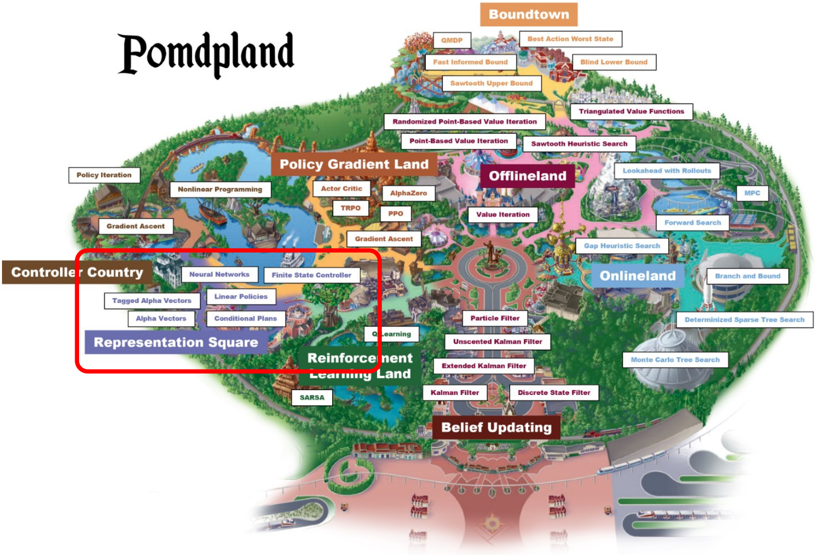
- Discrete probability distribution (e.g., $\mathbf{b} = [0.2, 0.8]$)
- Parameters of a distribution (e.g., $b \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$)
- Collection of particles (e.g., $[3.4, 2.2, 9.6]$)

Belief updating

- How do update our belief b given that we took action a and observed o?
- Bayes' rule tells us how to do this:

$$b'(s') = O(o \mid a, s') \sum_{s} T(s' \mid s, a) b(s)$$

- We can use different approaches to define $b' = \text{Update}(b, a, o)$
  - Discrete state filter
  - Kalman filter
  - Extended Kalman filter
  - Unscented Kalman filter
  - Particle filter



Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Representation Square
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Linear policies

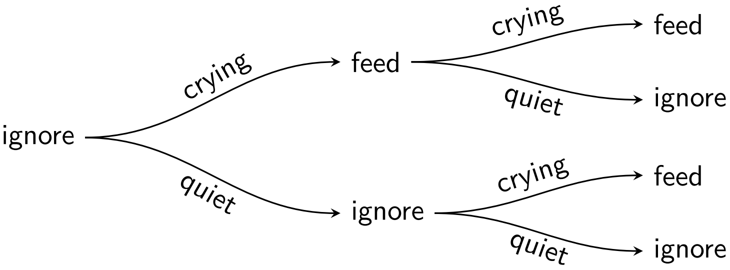
- $\pi(b) = \mathbf{A}\hat{\mathbf{s}}$ where $\hat{\mathbf{s}}$ is the most likely state according to b
- Optimal for some problems (e.g., linear quadradic Gaussian)
- Good approximation for many other problems

Neural network policies

- Handles very high dimensional problems (e.g., image inputs)
- Can use sequence of observations instead of beliefs
- May use LSTM, GRU, transformer architectures for tracking relevant past information

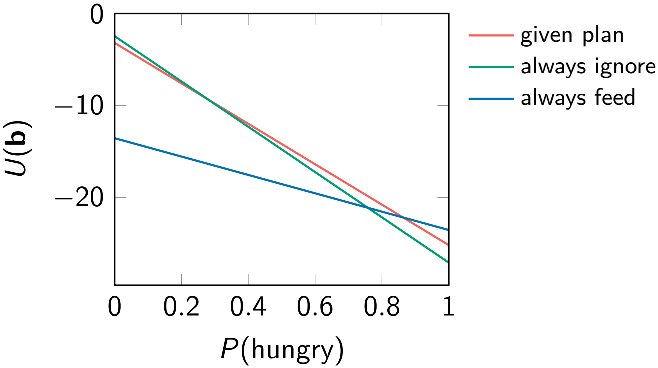
Conditional plans

Represent action to given any possible sequence of observations, up to some horizon



Alpha vector policies

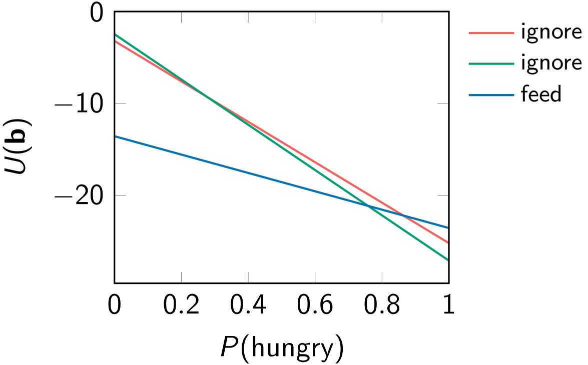
An alpha vector $\boldsymbol{\alpha}$ defines a hyperplane over the belief space representing the value of executing a conditional plan $U(\mathbf{b}) = \boldsymbol{\alpha}^\top \mathbf{b}$



- $U(b) = \max_{\alpha \in \Gamma} \boldsymbol{\alpha}^\top \mathbf{b}$, where Γ contains alpha vectors
- $\pi(b) = \arg\max_a [R(b, a) + \gamma \sum_o P(o \mid b, a) U(\text{Update}(b, a, o))]$

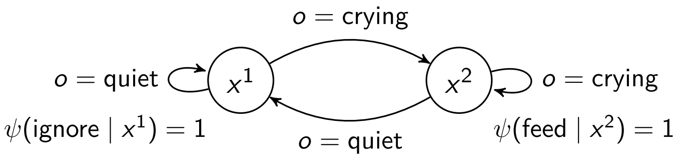
Tagged alpha vector policies

A tagged alpha vector is just an alpha vector labeled with the root action of the corresponding conditional plan

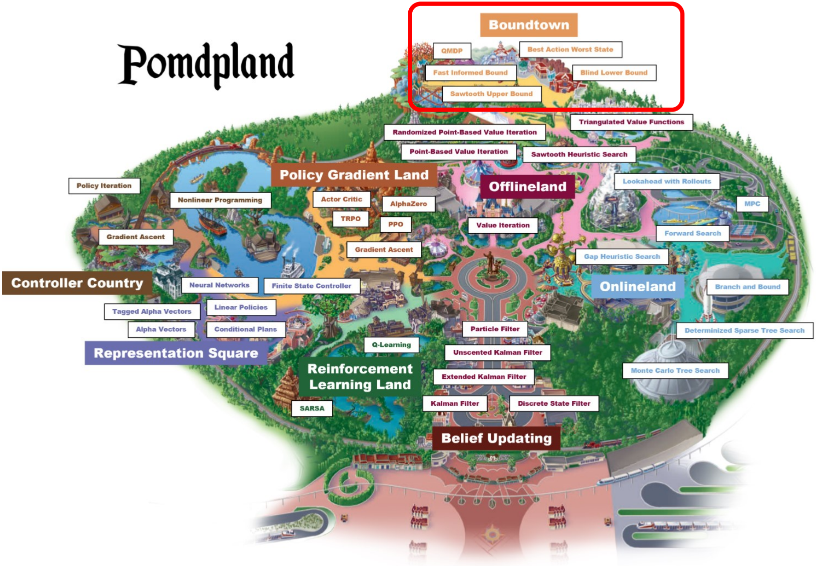


Given our current belief, we find the maximizing alpha vector and execute the corresponding action

Finite state controller policies

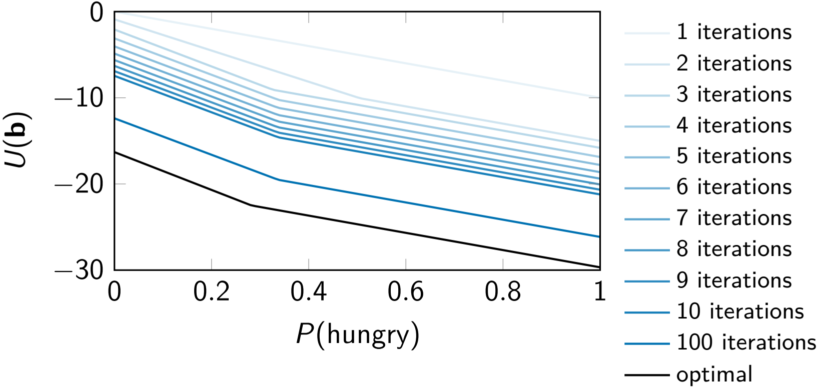


- Actions are associated with nodes
- Observations are associated with edges
- Associations can be stochastic


Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
Lookahead with Rollouts
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Unscented Kalman Filter
Representation Square
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

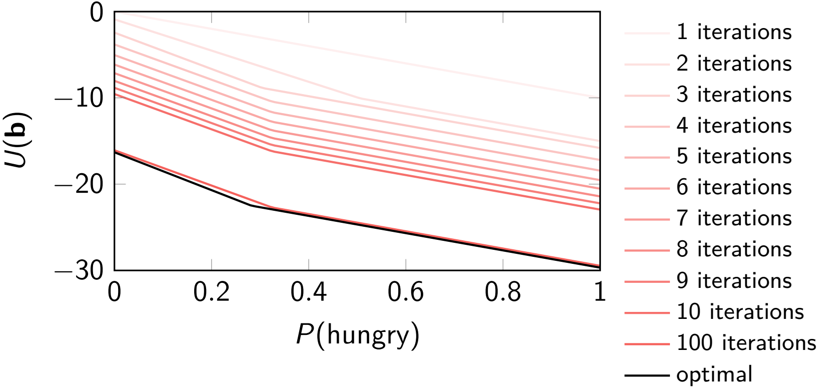
QMDP upper bound

- Assumes all uncertainty vanishes after first action
- Provides an upper bound
- Can be represented using one alpha vector per action



FIB upper bownd

- Upper bound no less tight (and generally tighter) than QMDP
- One alpha vector per action
- Uses observation model in calculations
- More expensive than QMDP, but tightness may be worthwhile



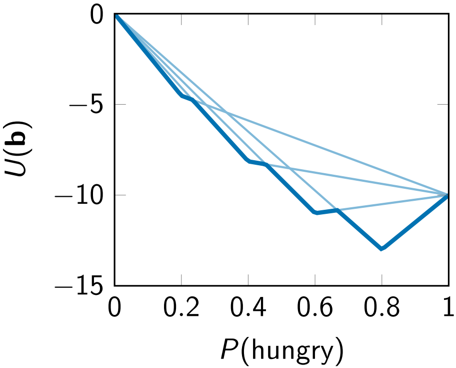
Sawtooth upper bound

Store a set of belief-utility pairs:

$$V = \{(b_1, U(b_1)), \ldots, (b_m, U(b_m))\}$$

with the requirement that V contains all the standard basis beliefs:

$$E = \{e_1 = [1, 0, \ldots, 0], \ldots, e_n = [0, 0, \ldots, 1]$$



BAWS lower bound

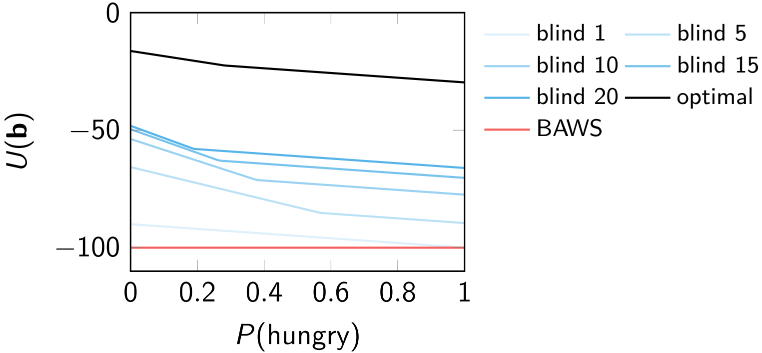
BAWS: best-action from worst state forever

$$r_{\text{baws}} = \max_{a} \sum_{k=1}^{\infty} \gamma^{k-1} \min_{s} R(s,a) = \frac{1}{1-\gamma} \max_{a} \min_{s} R(s,a)$$

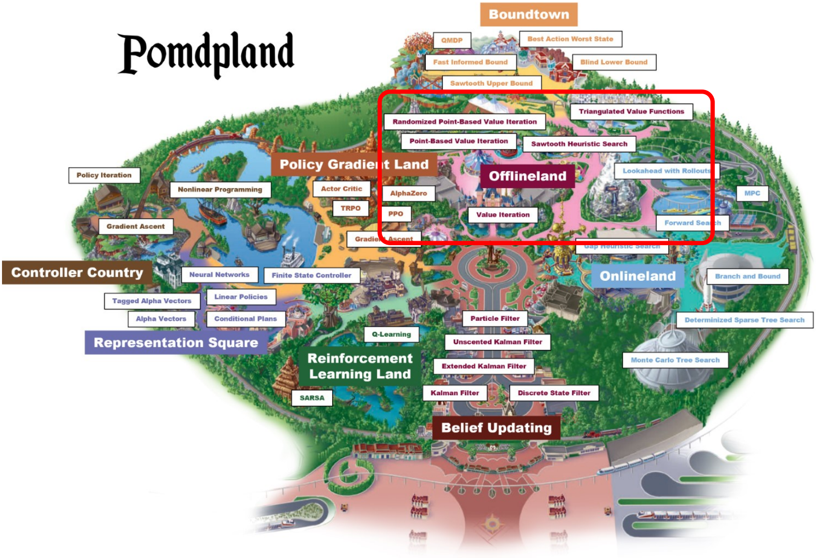
Generally pretty loose

Blind lower bound

Use alpha vectors corresponding to committing to different actions forever



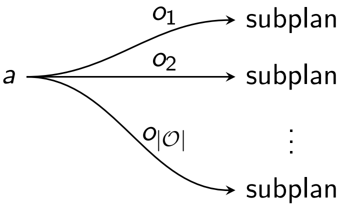
Pomdpland

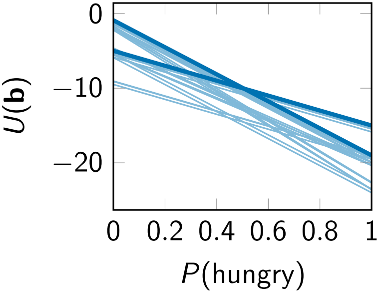


Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Representation Square
Q-Learning
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Value iteration

1. Construct all one-step plans
2. Prune dominated one-step plans based on alpha vectors
3. Create two-step plans by using remaining one-step plans as subplans
4. Prune dominated two-step plans
5. etc.





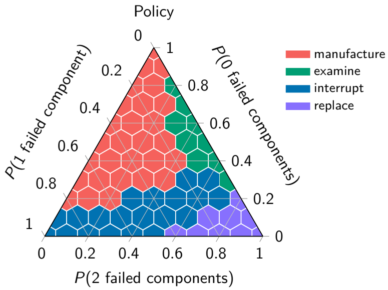
Point-based value iteration

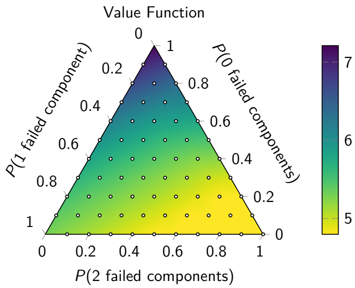
- Compute m different alpha vectors $\Gamma = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m\}$, each associated with different belief points $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$
- Each iteration ensures Γ preserves a lower bound
- Iteratively update the alpha vectors using point-based backup at their corresponding belief states

$$\boldsymbol{\alpha} = \text{Backup}(\Gamma, \mathbf{b})$$

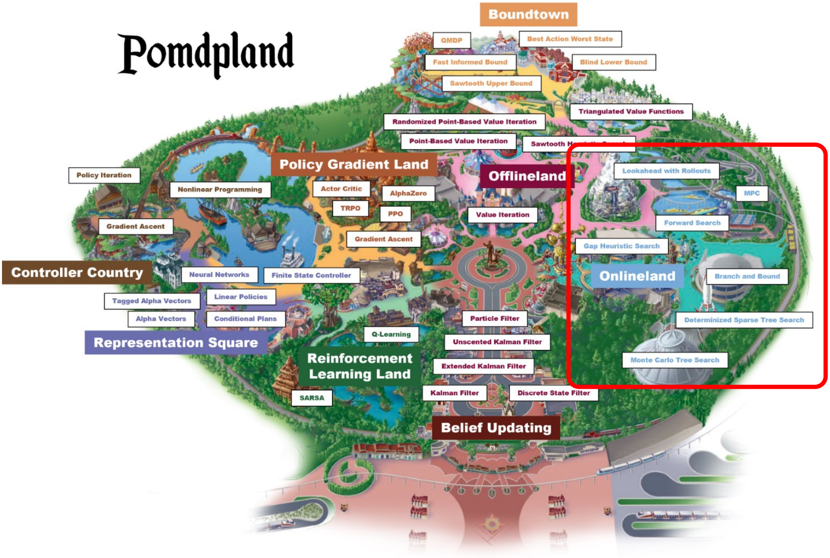
- Many variations of this basic approach

Triangulated value iteration





Pomdpland

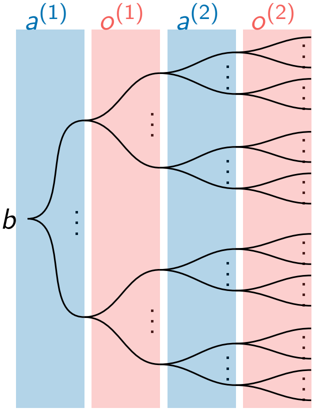

Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth H
Policy Gradient Land
Offlineland
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
TRPO
PPO
Value Iteration
Gradient Ascent
Gradient Ascent
Lookahead with Rollouts
MPC
Forward Search
Gap Heuristic Search
Onlineland
Branch and Bound
Controller Country
Neural Networks
Finite State Controller
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Representation Square
Q-Learning
Particle Filter
Unscented Kalman Filter
Determinized Sparse Tree Search
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Lookahead with rollouts

- Try out each action and estimate reward by running a rollout simulation; pick the best one
- The rollout simulation just runs some heuristic policy
- Quality depends on rollout policy
- Super simple and fast

Forward search

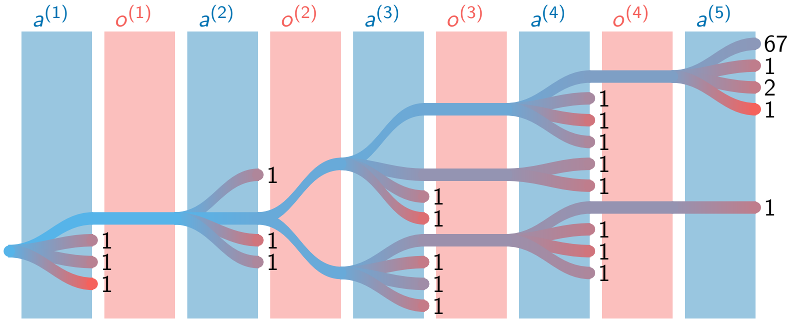
- Consider every possible sequence of actions and observations
- Pick the best first action, and replan



Branch and bound

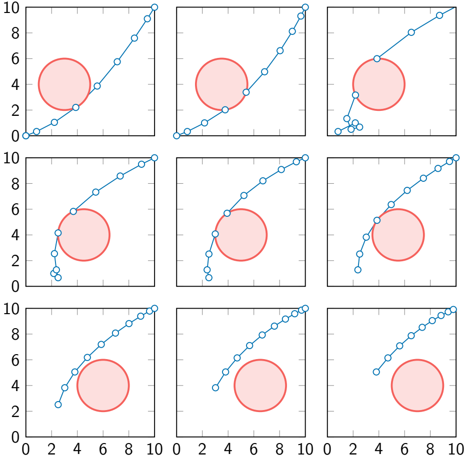
- Use upper bound and lower bounds on $U(b)$ to prune space
- Worst case, as expensive as forward search
- In practice, much faster than forward search
- Preserves optimality (up to horizon)

Monte Carlo tree search

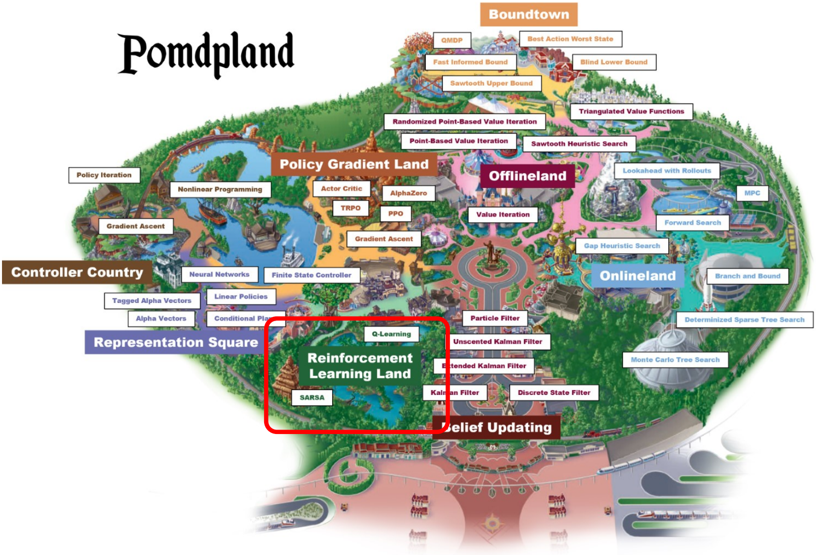


Model predictive control

Optimize sequence of actions, and replan at every step



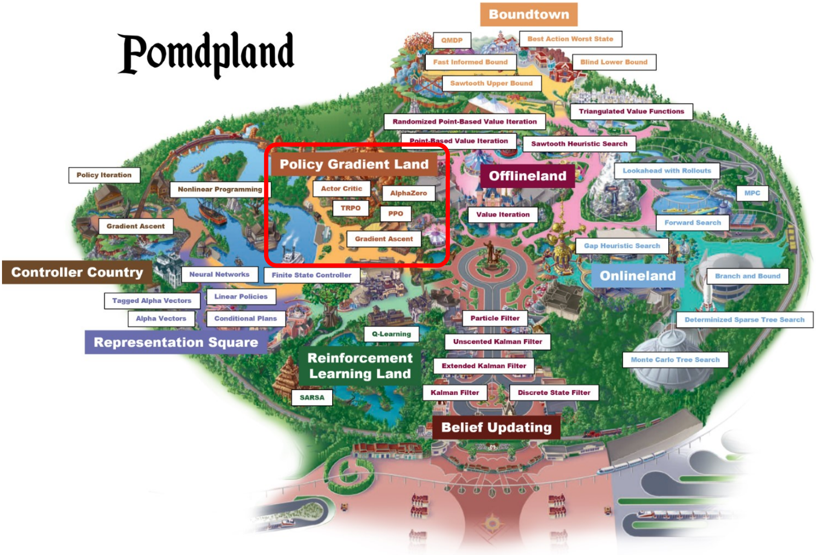
Pomdpland



Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Pla
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Representation Square
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Reinforcement learning

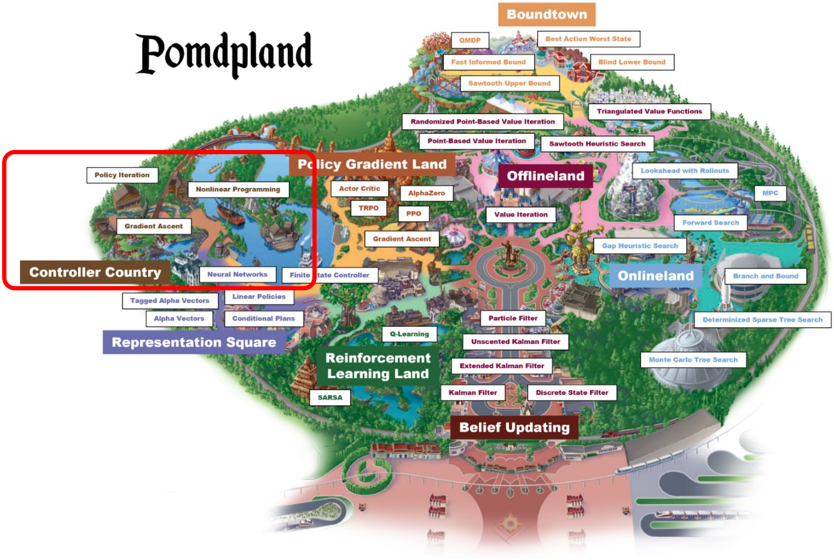
- Learn a good policy by interacting with the real world or simulator
- Often requires many simulation steps
- Can use neural network to represent value function, use this to drive exploration
- Different approaches for updating value function, exploring actions
- State-of-the-art for some problems



Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Representation Square
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Reinforcement learning

- Can directly optimize parameters of a policy
- Gradient-based optimization approaches tend to be more efficient when optimizing over many parameters
- Evaluation of objective function involves simulating policy
- Can use a representation of policy to drive optimization (e.g., AlphaZero and other actor-critic methods)


Pomdpland
Boundtown
QMDP
Best Action Worst State
Fast Informed Bound
Blind Lower Bound
Sawtooth Upper Bound
Triangulated Value Functions
Randomized Point-Based Value Iteration
Point-Based Value Iteration
Sawtooth Heuristic Search
Policy Gradient Land
Offlineland
Lookahead with Rollouts
Policy Iteration
Nonlinear Programming
Actor Critic
AlphaZero
MPC
TRPO
PPO
Value Iteration
Forward Search
Gradient Ascent
Gradient Ascent
Gap Heuristic Search
Controller Country
Neural Networks
Finite State Controller
Onlineland
Branch and Bound
Tagged Alpha Vectors
Linear Policies
Alpha Vectors
Conditional Plans
Particle Filter
Determinized Sparse Tree Search
Q-Learning
Representation Square
Unscented Kalman Filter
Reinforcement Learning Land
Extended Kalman Filter
Monte Carlo Tree Search
SARSA
Kalman Filter
Discrete State Filter
Belief Updating

Policy evaluation

$$U(x,s) = \sum_a \psi(a \mid x)(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) \sum_o O(o \mid a,s') \sum_{x'} \eta(x' \mid x,a,o) U(x',s'))$$

- x node in controller
- $\psi(a \mid x)$ probability of taking action a in node x
- $\eta(x' \mid x, a, o)$ probability of transitioning to node x' give we are in node x, take action a, and observe o

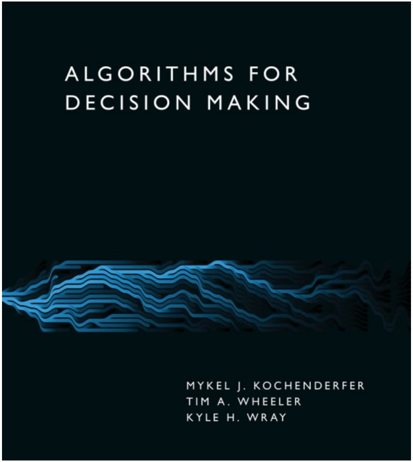
Given a controller and initial belief b, select initial node $x^* = \arg\max_x \sum_s U(x,s)b(s)$

Policy iteration

1. Start with any initial controller
2. Evaluate controller
3. Improve policy by adding new nodes and edges
4. Prune useless nodes
5. Go to step 2

Nonlinear programming

$$
\begin{aligned}
\underset{U,\psi,\eta}{\text{maximize}} \quad & \sum_{s} b(s) U(x^1, s) \\
\text{subject to} \quad & U(x, s) = \sum_{a} \psi(a \mid x)\Big(R(s, a) + \gamma \sum_{s'} T(s' \mid s, a) \sum_{o} \ldots, \\
& \psi(a \mid x) \geq 0 \quad \text{for all } x, a, \\
& \sum_{a} \psi(a \mid x) = 1 \quad \text{for all } x, \\
& \eta(x' \mid x, a, o) \geq 0 \quad \text{for all } x, a, o, x', \\
& \sum_{x'} \eta(x' \mid x, a, o) = 1 \quad \text{for all } x, a, o
\end{aligned}
\tag{1}
$$



algorithmsbook.com/decisionmaking