

Optimization-Based Approximate Dynamic Programming

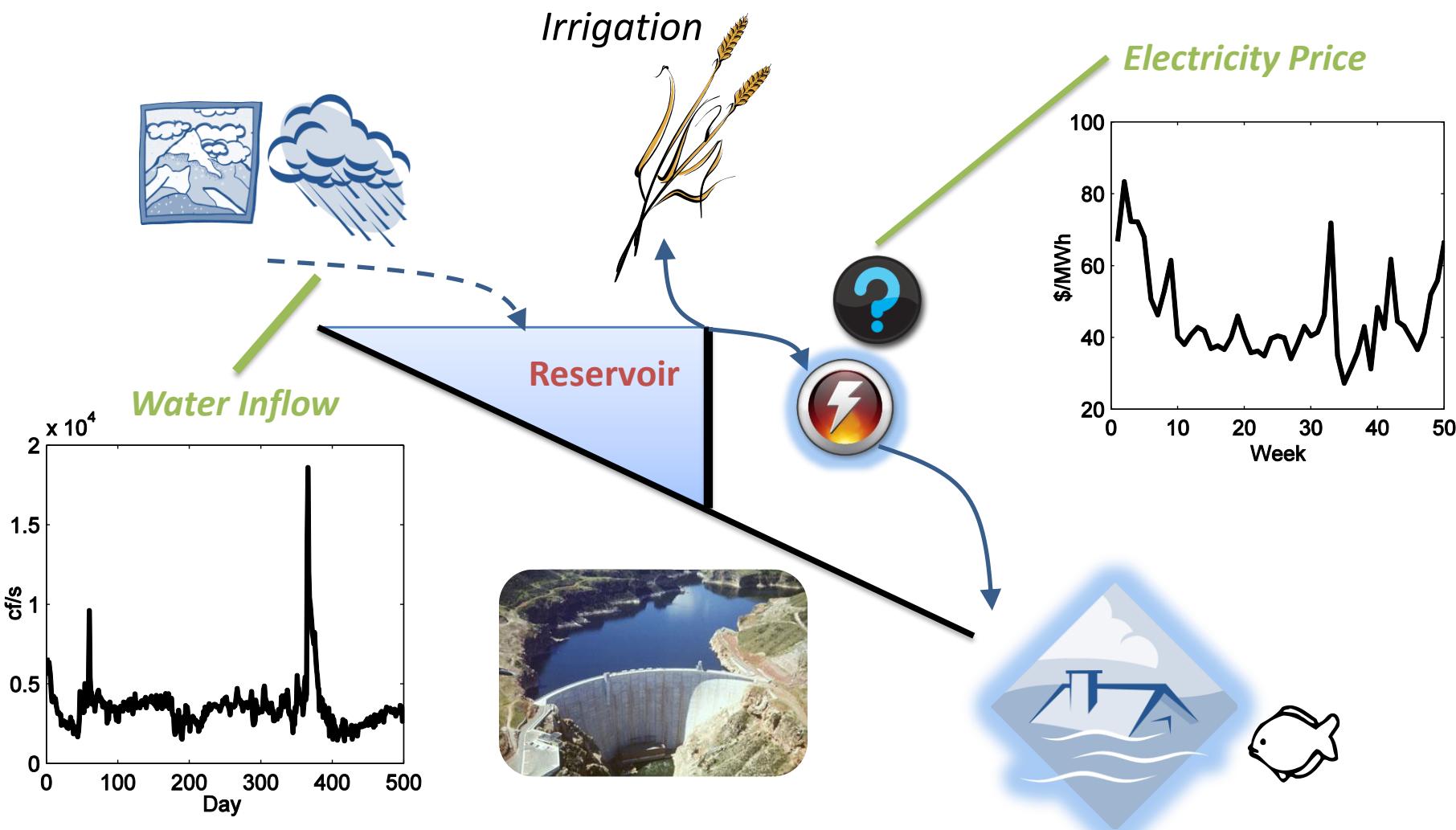
Marek Petrik

Committee members:
Shlomo Zilberstein
Andrew Barto
Sridhar Mahadevan
Ana Muriel
Ronald Parr

Thesis Objectives

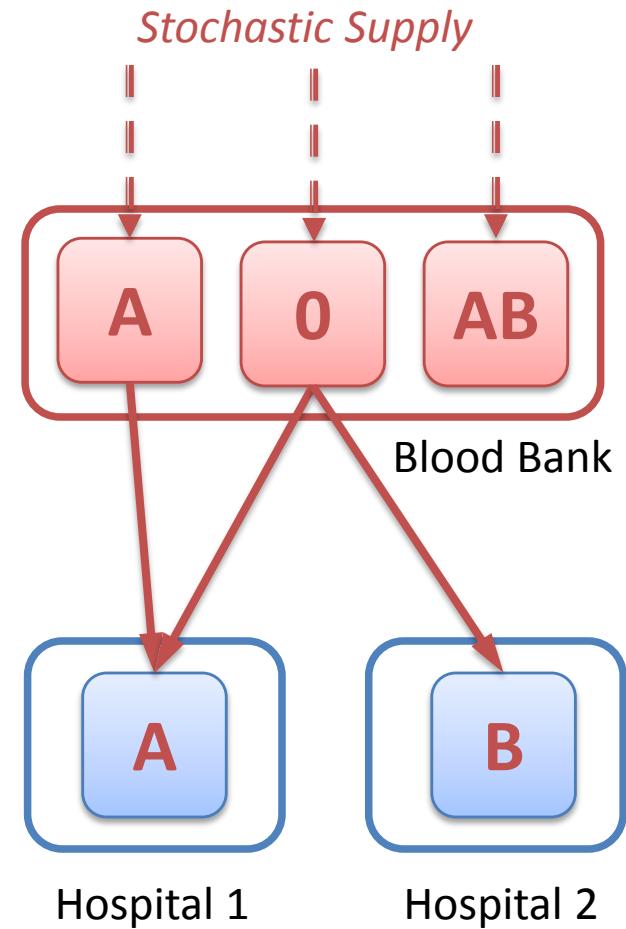
- Reinforcement learning
 - Combines **optimization** and **machine learning**
- **Challenge:** Existing methods are unreliable
 - Hard to use, analyze, and trust
- **Objective:** Develop more reliable methods that:
 - Provide better guarantees
 - Are easier to *use, analyze, understand*
- **My approach:**
 1. Deepen understanding of basic principles
 2. Build algorithms based on the basic principles

Application: Managing River Dam



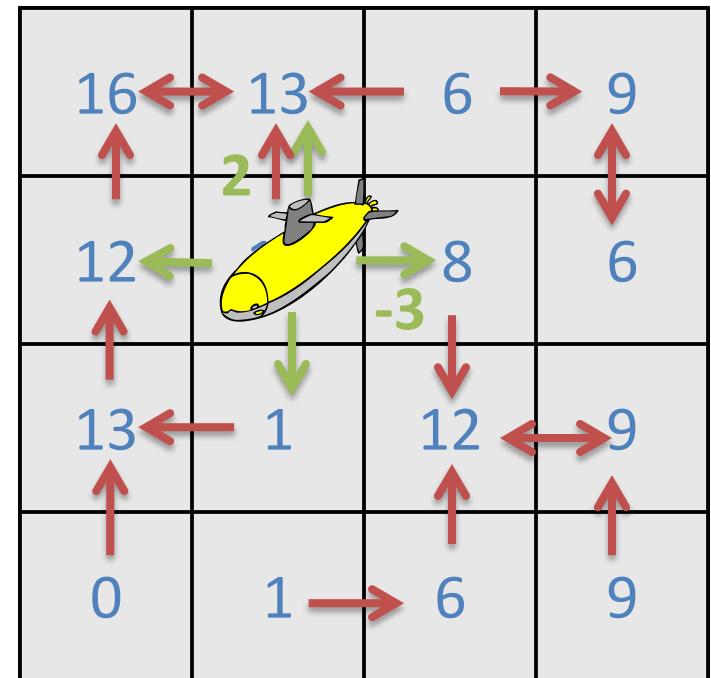
Application: Managing Blood Inventories

- Managing blood inventory
 - Minimize **shortage** – demand that is not satisfied
 - Maximize **utilization** – use before it perishes
- Take advantage of blood-type compatibility



Domain Model

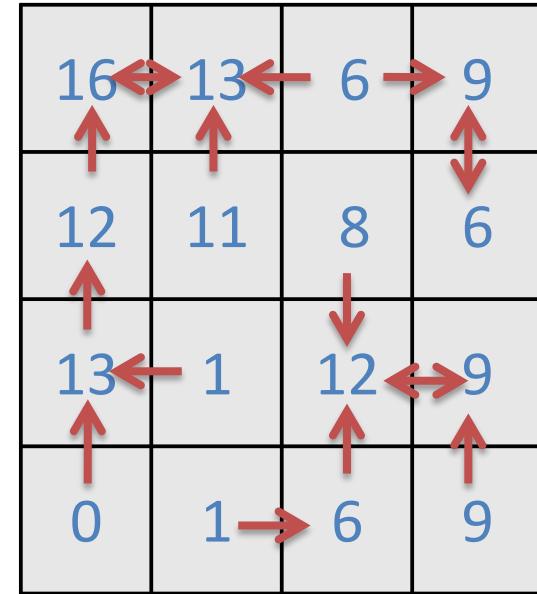
- Markov Decision Process
 - States (grid) S
 - Actions: A
- Optimal value function – best utility of being in each state
- Optimal policy – decision each state
- Optimal state visitation frequencies – how much time in each state



Maximize rewards
Infinite horizon: discount: γ

Value Function and Policy

- True solution: policy
- Calculate policy π from VF v
 - Take the best greedy action
- Calculate VF for policy v_π
 - Add rewards for policy
- Policy visitation frequencies
 - How much time policy spends in a state u_π
 - Upper bound (importance of a state) \bar{u}_π

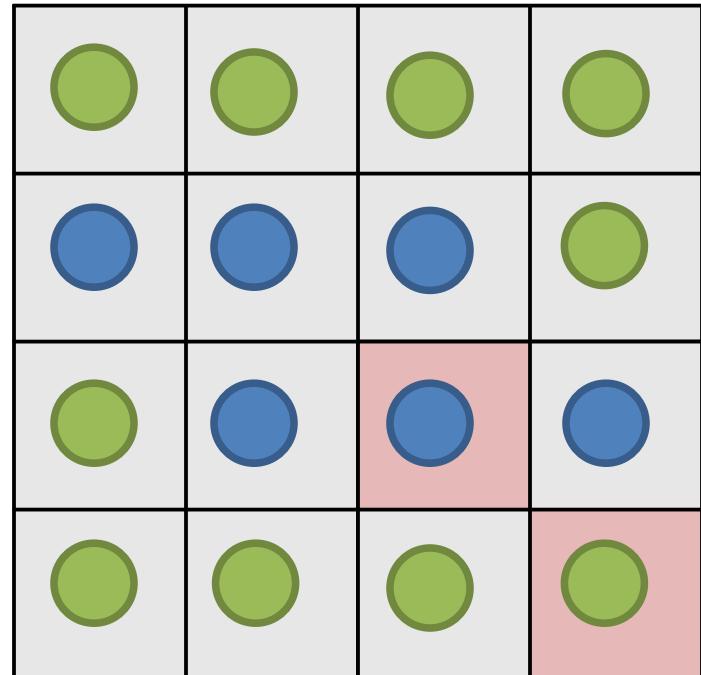


Approximate Value Function

- Too large – must approximate
- Value function based on state features

$$\text{Green circle} = 1 \quad \text{Blue circle} = 4$$

- Linear value function approximation
- **Representable** value functions



Restrict The Space of Value Functions

Small number of features

- Value function represented a small number of features

$$\tilde{v} = \Phi \times x$$

L₁ Regularization

- Large number of features
- Small volume

$$\tilde{v} = \Phi \times x$$

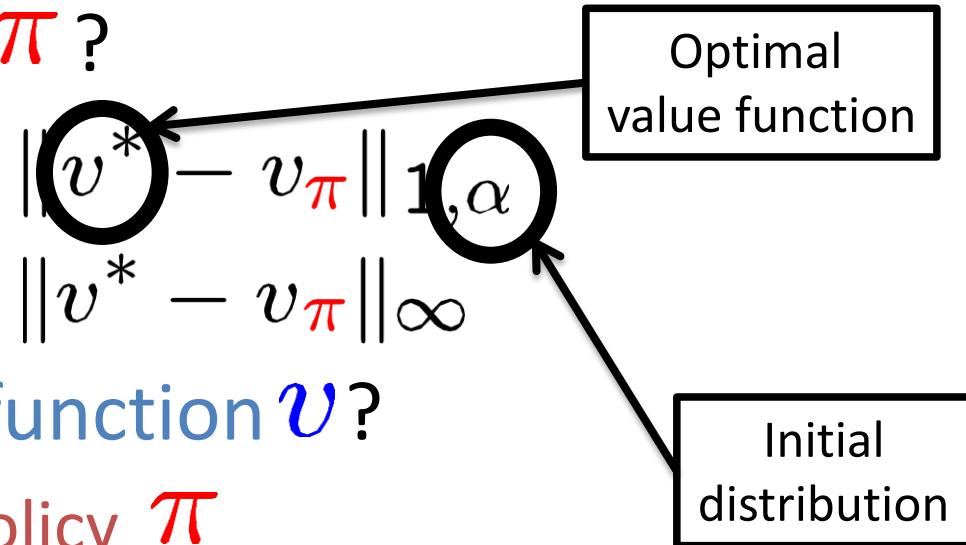
Representable value functions

$$\|x\|_1 \leq \psi$$

$$\mathcal{M} = \{v = \Phi x \mid \|x\|_{1,e} \leq \psi\}$$

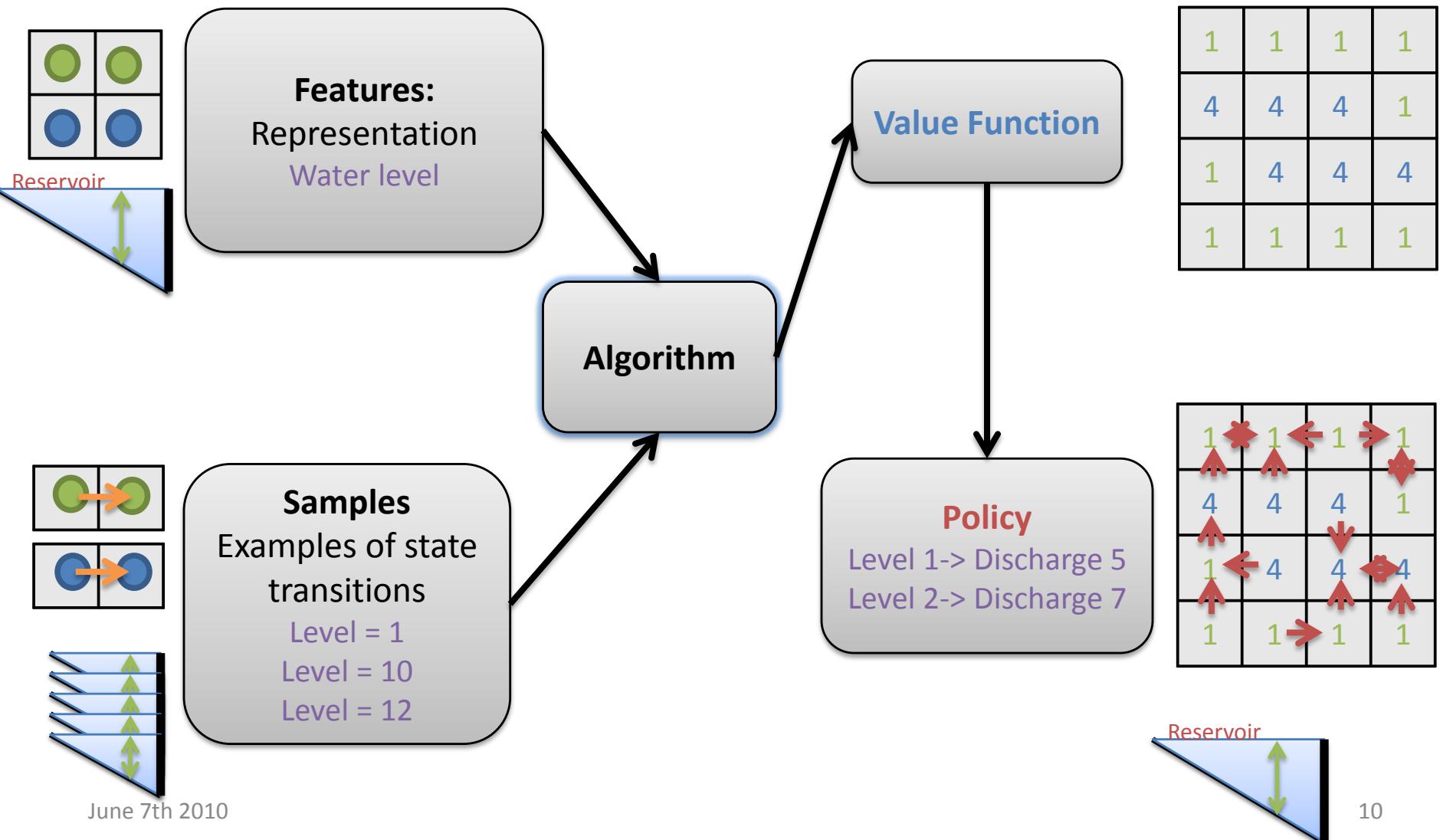
Value Function Approximation: Objectives

- How good is a policy π ?
- Expected policy loss $\|v^* - v_\pi\|_{1,\alpha}$
- Robust policy loss $\|v^* - v_\pi\|_\infty$
- How good is a value function v ?
 - Quality of a greedy policy π
- Desirable bounds: for some c

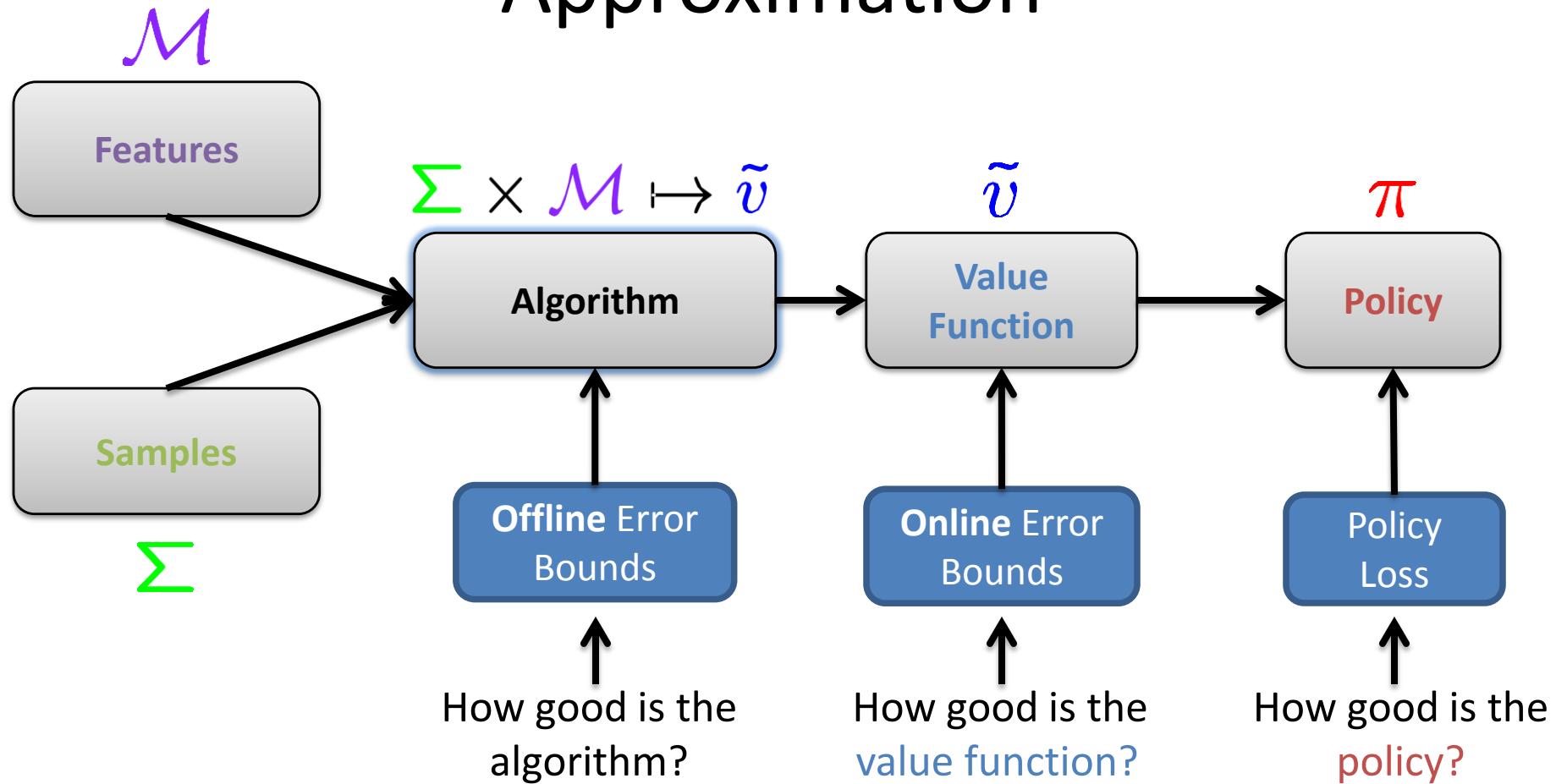


$$\|v^* - v_\pi\|_{1,\alpha} \leq c \min_{v \in \mathcal{M}} \|v^* - v\|_\infty$$

Components of Value Function Approximation



Error Bounds in Value Function Approximation

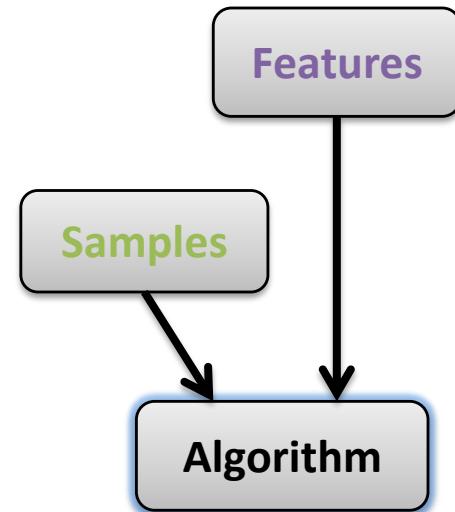


Objectives

- If solved a problem using RL then **patent it** [Powell 2007]
- **Major challenge:** Off-the-shelf methods that are easy to use by non-practitioners
- I propose algorithms guided by error bounds
 - Strong guarantees & easy analysis
 - Good practical performance
- Easier to use, because we know:
 - When they work
 - Why they do not work

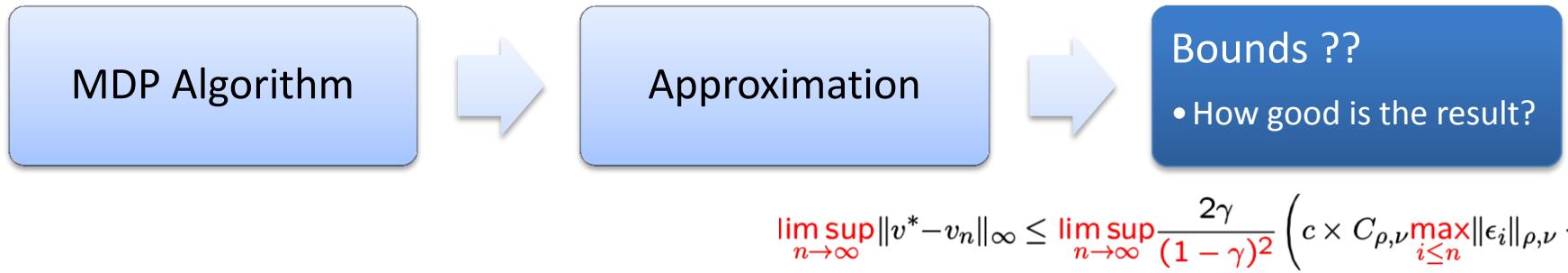
Why Is It Difficult?

- Many components must interact; balance errors:
 - **Representational error:** Due to features
 - **Sampling error:** Due to missing samples
 - **Algorithmic error:** Due to approximate optimization
- Better understand components to understand interaction

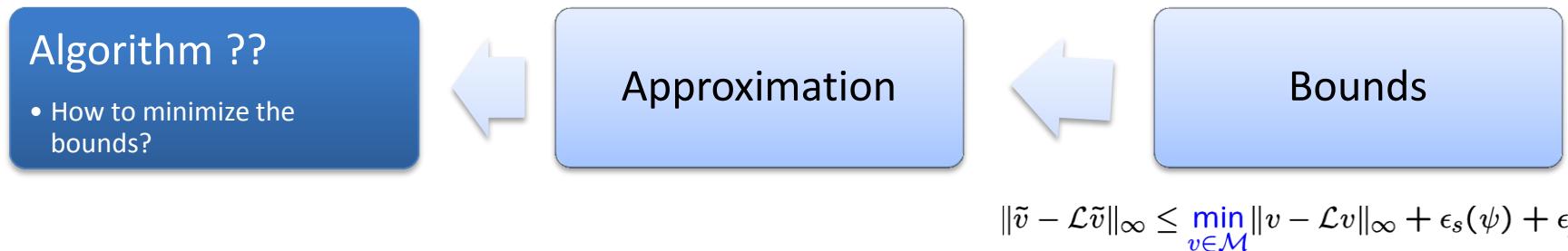


My Approach *

Classical Iterative Algorithms



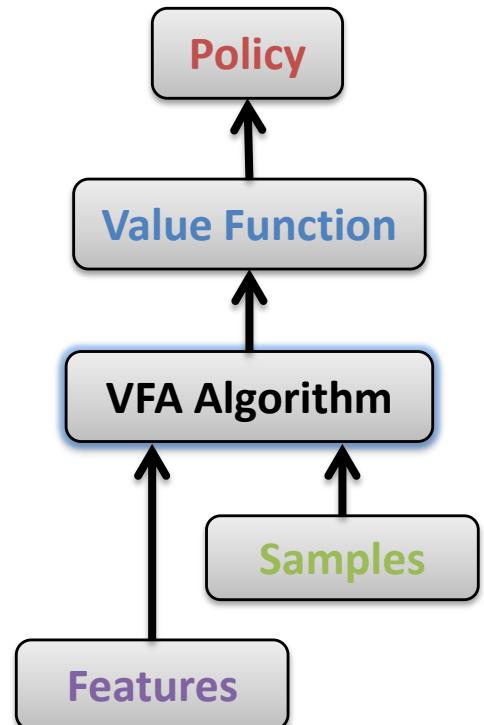
Optimization-based algorithms



Approximate Dynamic Programming = Value Function Approximation

Main Contributions

- Bottom-up: From simple to complex
- New and improved methods for
 1. Tight online error bounds
 2. VFA algorithms with strong guarantees
 3. Evaluating sample quality
 4. Automatically selecting features



Outline

1. Framework

Value function approximation

2. Optimization-based Formulations

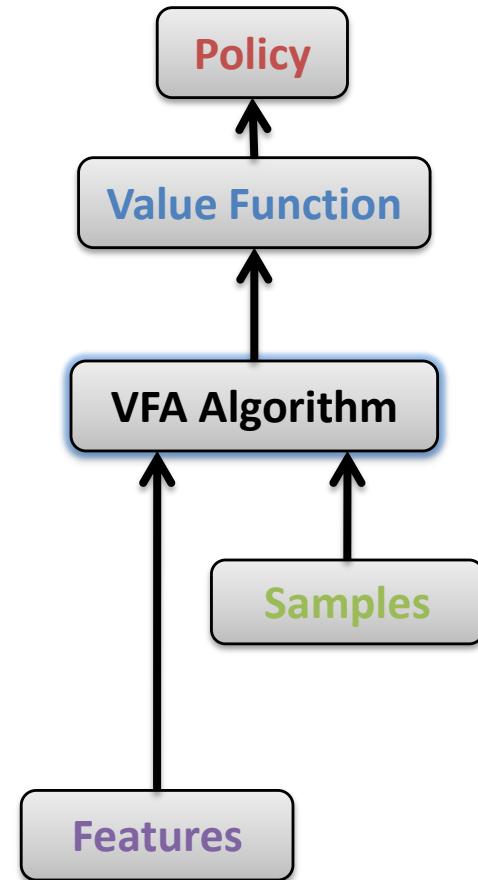
Approximate linear and bilinear
programming

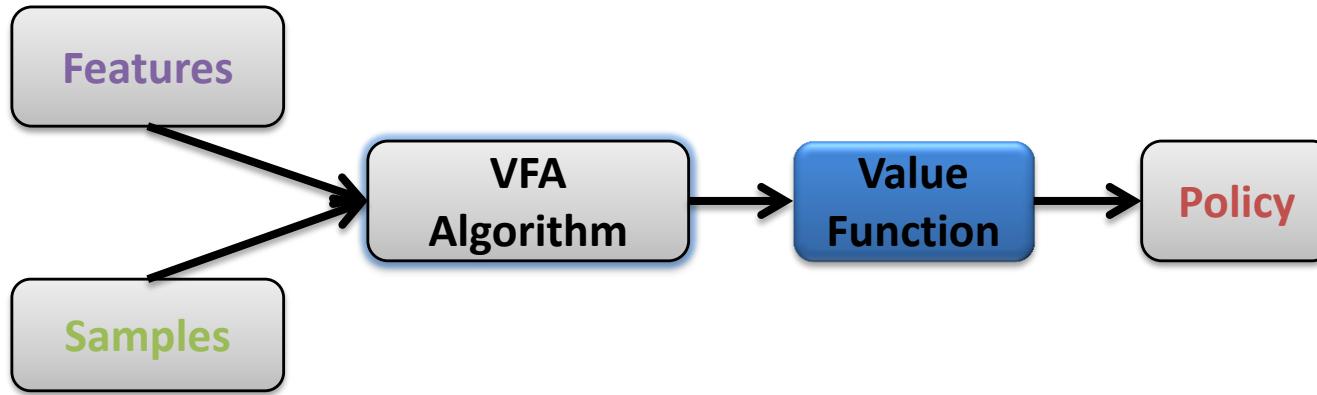
3. Sampling Bounds

How good are samples

4. Feature Selection

How to select good features

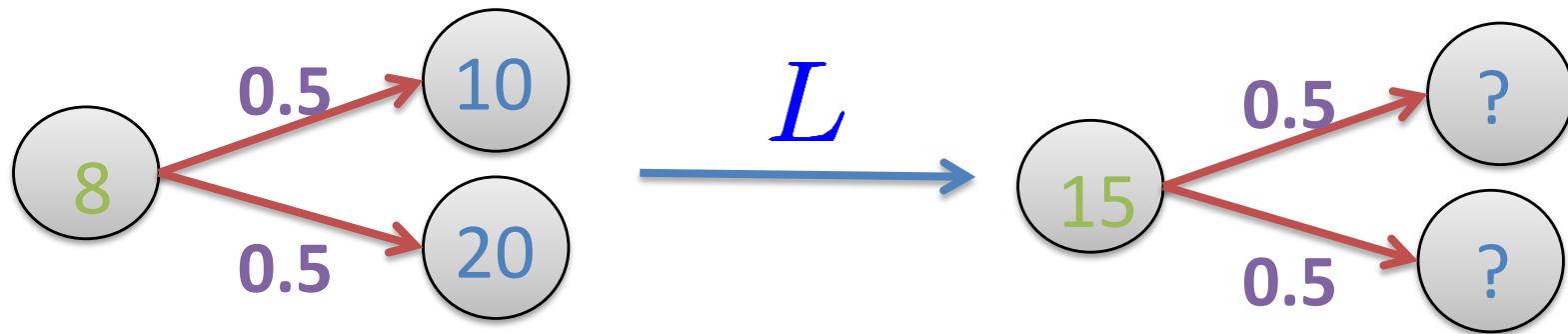




FRAMEWORK AND ITERATIVE ALGORITHMS

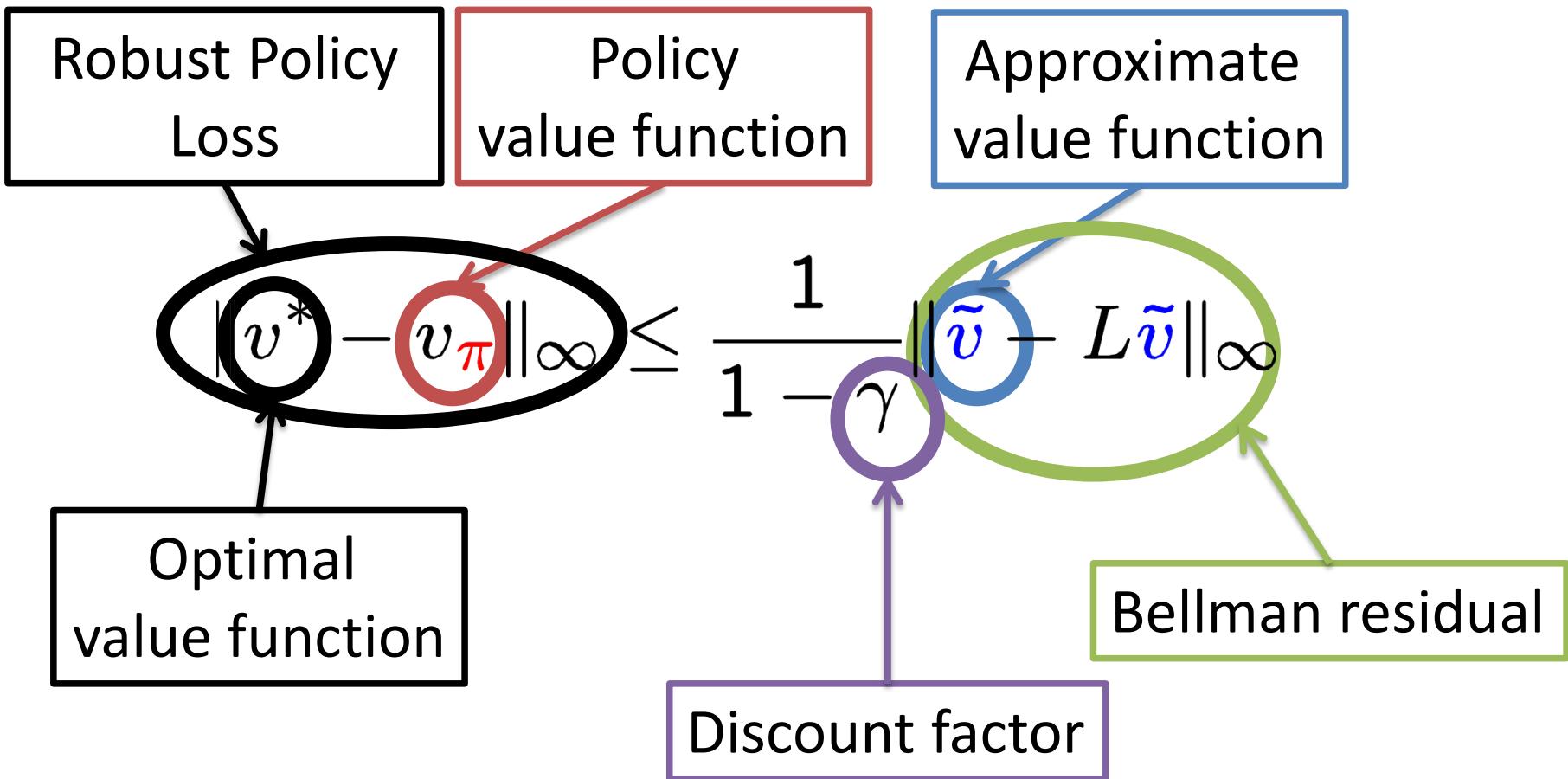
Bellman Operator L

- Propagates the expected value backwards
 - Transition probabilities 0.5

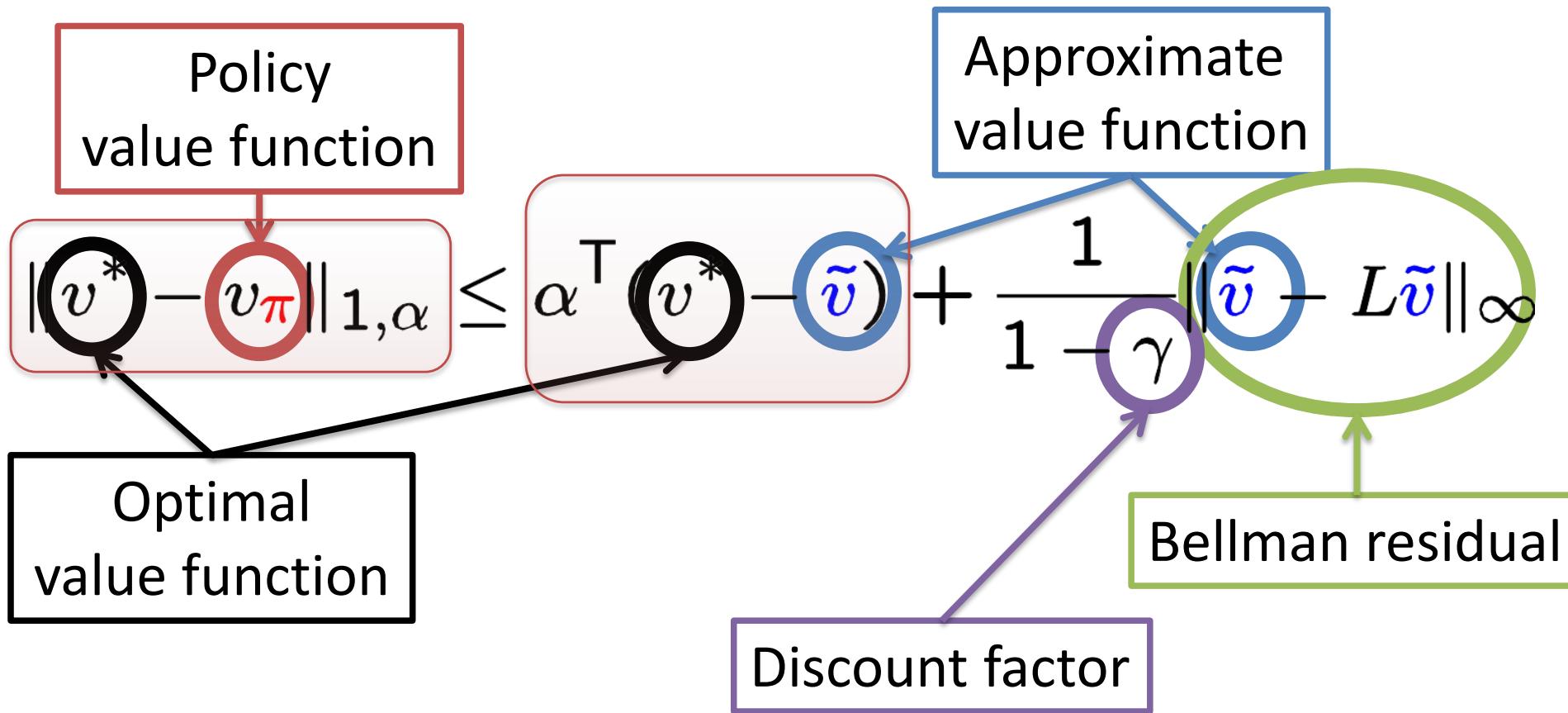


- Bellman residual $v - Lv$
 - BR: $|8 - (0.5 * 10 + 0.5 * 20)| = |8 - 15| = 7$
 - Measures self-consistency of the value function

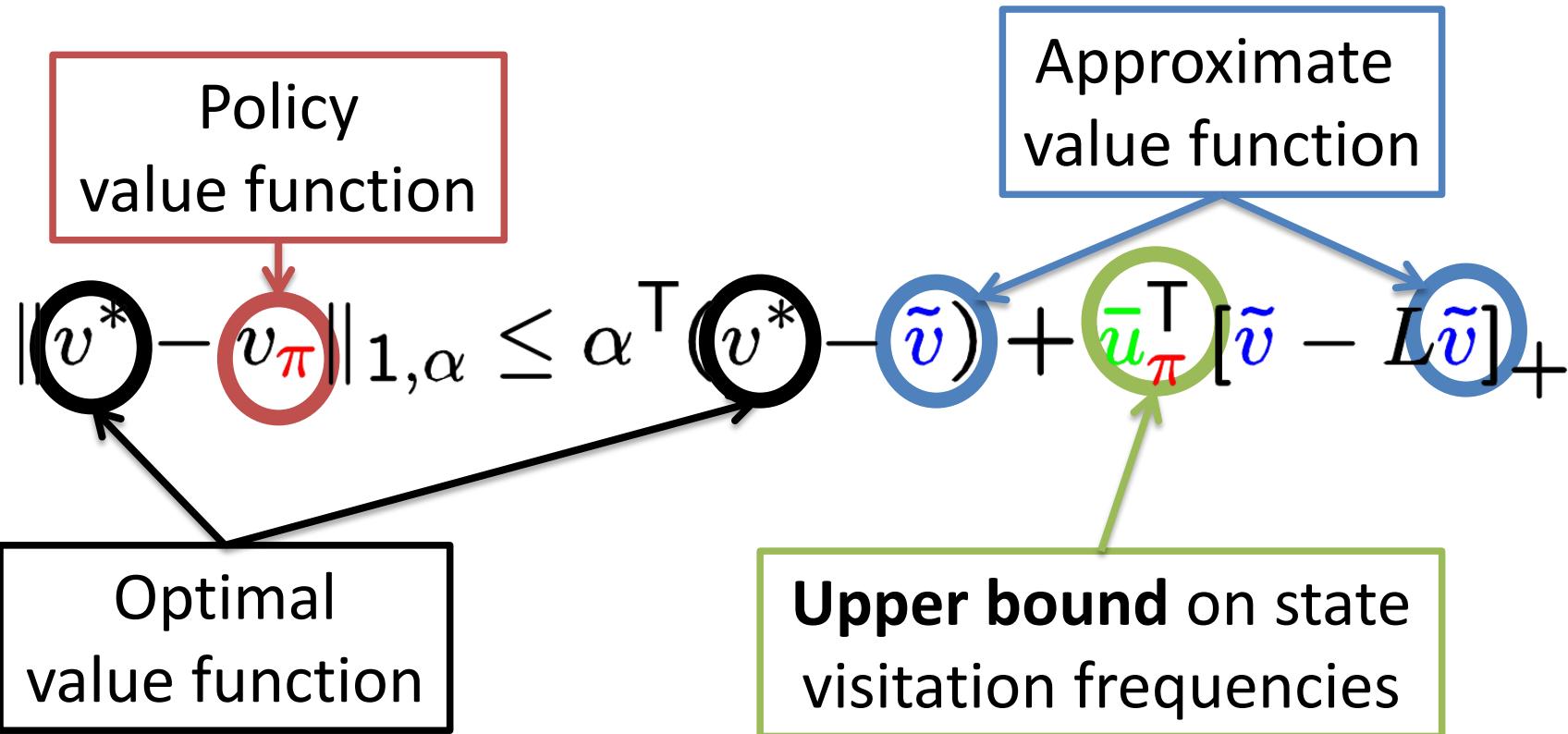
Online Error Bounds: Robust Policy Loss



Online Error Bounds: Expected Policy Loss *

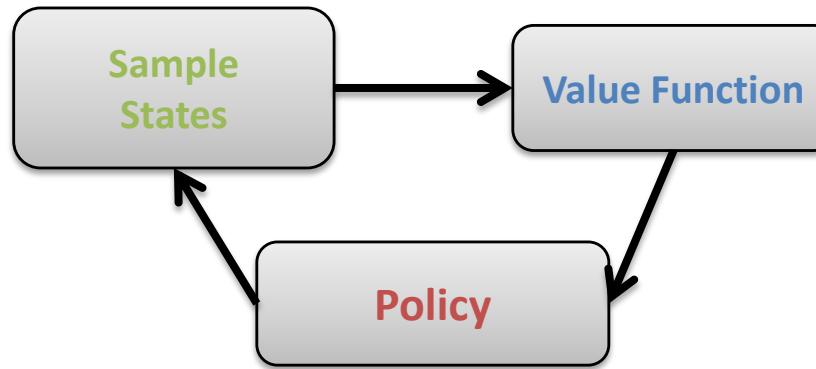


Online Error Bounds: Expected Policy Loss (Distribution) *



Approximate Policy Iteration

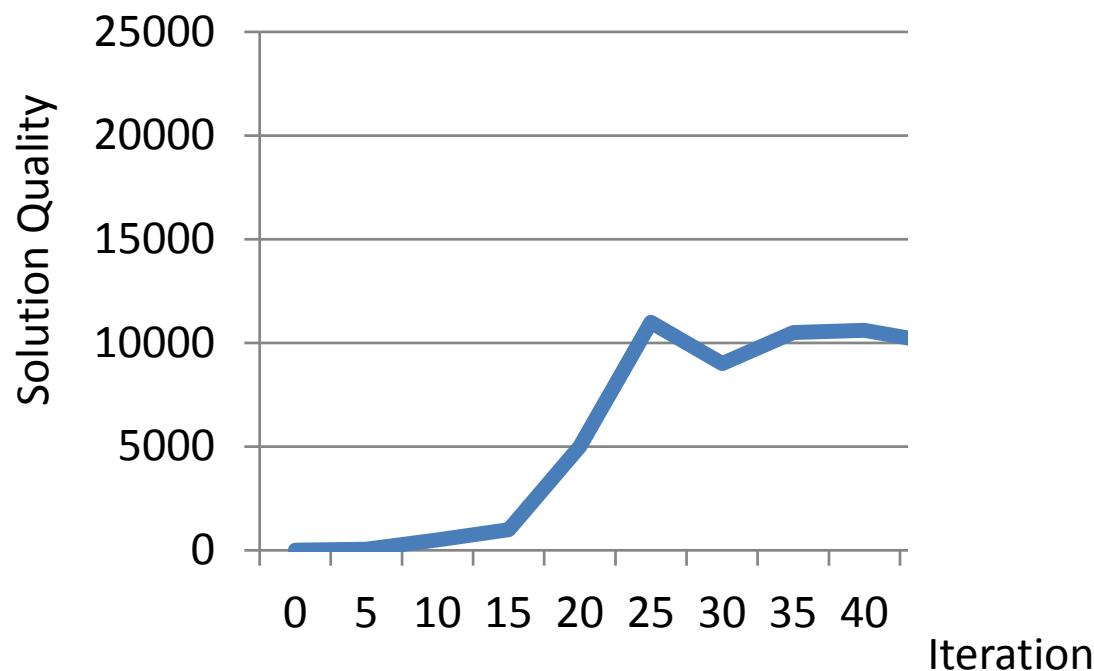
- Difficulty: Bellman operator is **nonlinear**
- API: Classical iterative VFA algorithm



- Bellman operator is **linear** for a fixed policy
- Based on *policy iteration*

Approximate Policy Iteration

- No convergence guarantees
- Do not know when to stop



Offline Bounds for API *

- Approximation Error Bounds

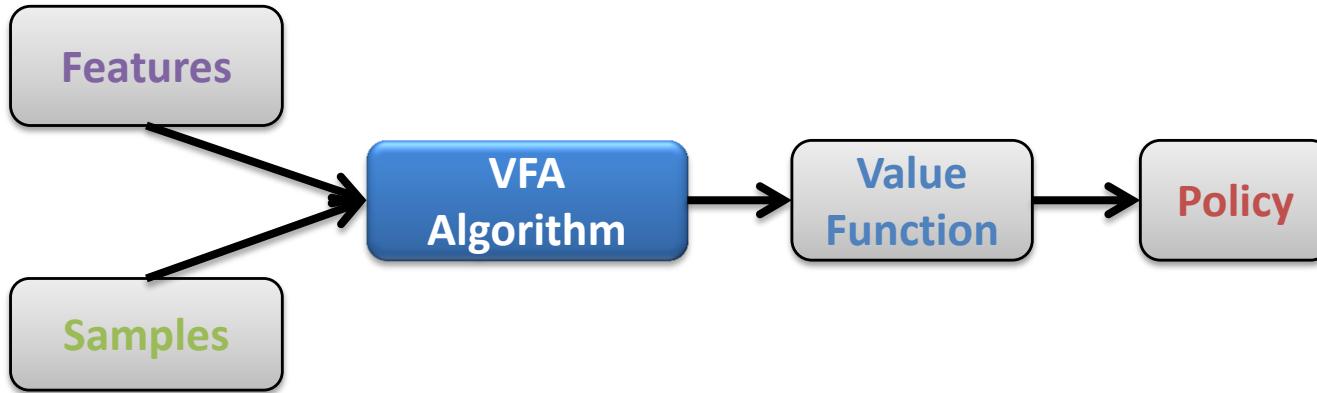
$$\limsup_{k \rightarrow \infty} \|v^* - v_{\pi^k}\|_\infty \leq \frac{2}{(1 - \gamma)^3} \limsup_{k \rightarrow \infty} \|\tilde{v}_k - L\tilde{v}_k\|_\infty$$

- No convergence
- Large constants
 - Possibly larger than value function

$$\gamma = 0.99 \quad \frac{1}{1 - \gamma} = 100 \quad \frac{1}{(1 - \gamma)^3} = 1000000$$

$$\limsup_{k \rightarrow \infty} \|v^* - v_{\pi^k}\|_\infty > c \min_{v \in \mathcal{M}} \|v - v^*\|_\infty$$

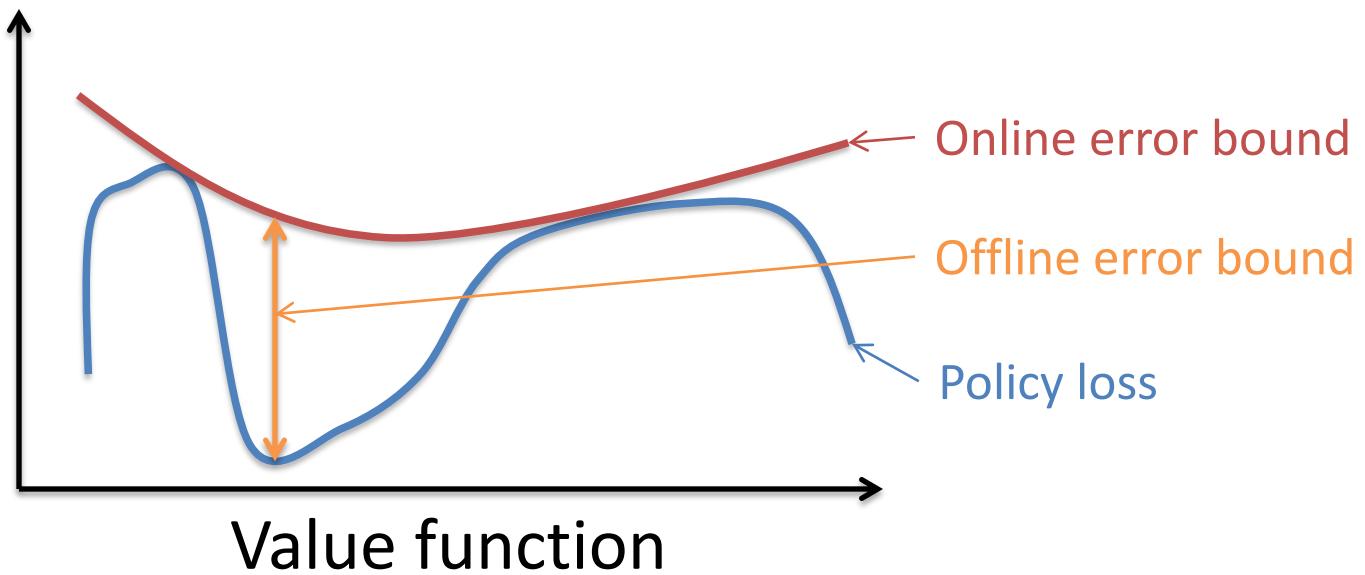
Desirable bound



OPTIMIZATION-BASED FORMULATIONS

Main Idea

- Minimize online bounds – the bounds should be as tight as possible



- For now assume that all samples are known

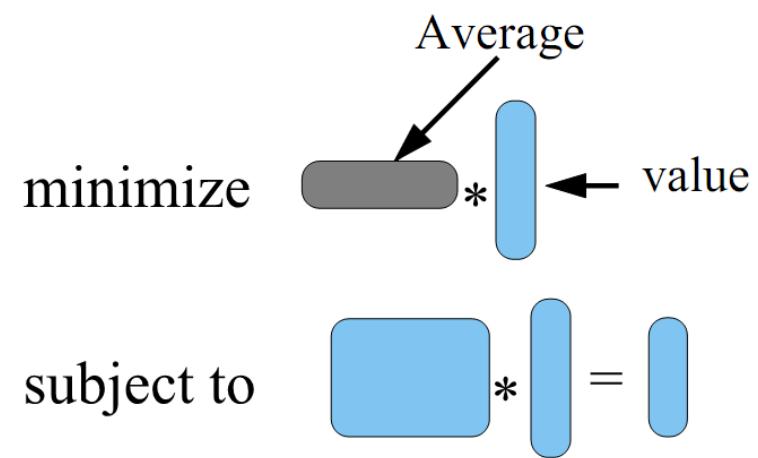
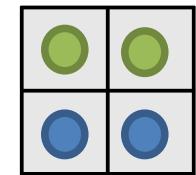
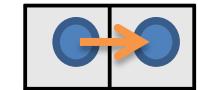
Approximate Linear Programming

- Classical approximation method
- Formulation
 - Constraints: transitions
 - Variables: features
- Guaranteed to converge
- Usually based on LP for MDPs

$$\min_{\boldsymbol{v}} \quad \boldsymbol{c}^T \boldsymbol{v}$$

$$\text{s.t.} \quad A\boldsymbol{v} \geq b$$

$$\boldsymbol{v} \in \mathcal{M}$$



Approximate Linear Programming as an Optimization-based Method *

- LP to minimize online error bounds
- Robust policy loss cannot be LP (P<NP)

$$\|v^* - v_{\pi}\|_\infty \leq \frac{1}{1-\gamma} \|L\tilde{v} - \tilde{v}\|_\infty$$

- Expected policy loss as LP

$$\|v^* - v_{\pi}\|_{1,\alpha} \leq \alpha^T(v^* - \tilde{v}) + \bar{u}_{\pi}^T [\tilde{v} - L\tilde{v}]_+$$

- Easy to formulate as an LP when \bar{u}_{π} is fixed
- Visitation frequencies \bar{u}_{π} depend on \tilde{v}

ALP: Offline Error Bounds *

- Bound for opt-based ALP

$$\|v^* - v_{\pi}\|_{1,\alpha} \leq \frac{2\bar{u}_{\pi}^T \mathbf{1}}{1 - \gamma} \min_{v \in \mathcal{M}} \|v^* - v\|_{\infty}$$
$$\propto \frac{2}{(1 - \gamma)^2} \min_{v \in \mathcal{M}} \|v^* - v\|_{\infty}$$

- Good value of \bar{u}_{π} is often unknown

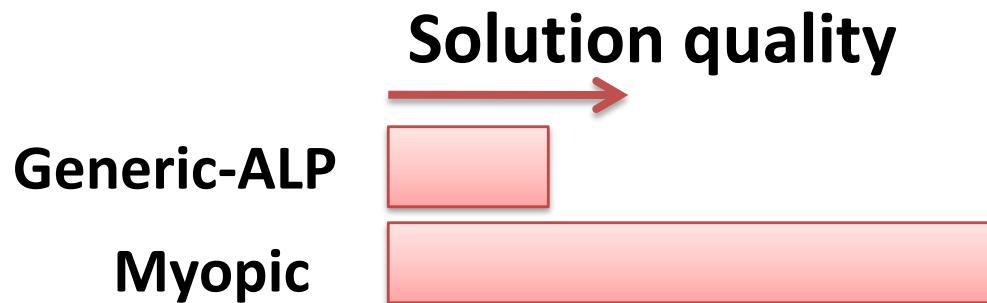
- Bound approximate value function

$$\|v^* - \tilde{v}\|_{1,c} \leq \frac{2}{1 - \gamma} \min_{v \in \mathcal{M}} \|v^* - v\|_{\infty}$$

Closest approximation

ALP: Practical Performance *

- Applied to blood inventory management



$$\|v^* - v_\pi\|_{1,\alpha} \propto \frac{2}{(1-\gamma)^2} \min_{v \in \mathcal{M}} \|v^* - v\|_\infty$$

- The bound is tight

$$\gamma = 0.99 \quad \frac{1}{1-\gamma} = 100 \quad \frac{1}{(1-\gamma)^2} = 10000$$

RALP: Alternative Formulation*

- Minimize a tighter **online** bound

$$\min_{\mathbf{v} \in \mathcal{M}} (\bar{\mathbf{u}}^T(\mathbf{I} - \gamma P^*) - \alpha^T)(\tilde{\mathbf{v}} - \mathbf{v}^*) + \bar{\mathbf{u}}^T [L\mathbf{v} - \mathbf{v}]_+$$

- Linear program:

$$\min_{\mathbf{v}} c^T \mathbf{v}$$

$$\text{s.t. } A\mathbf{v} \geq b$$

$$\min_{\mathbf{v} \in \mathcal{M}} c^T \mathbf{v} + \|[b - A\mathbf{v}]_+\|_{1,d}$$

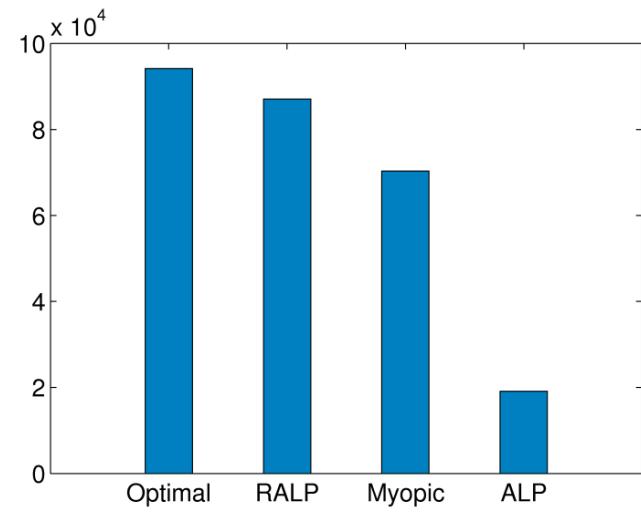
- Offline** error bound

$$\|\mathbf{v}^* - v_{\pi}\|_{1,\alpha} \propto \frac{1}{1-\gamma} \min_{\mathbf{v} \in \mathcal{M}} \|\mathbf{v} - \mathbf{v}^*\|_\infty$$

Does not depend on: $1/(1-\gamma)^2$

Approximate Linear Programming

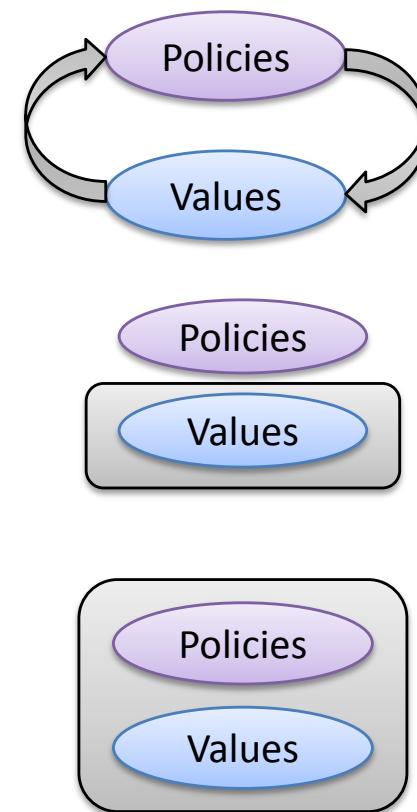
- RALP: good performance
- Alternatives necessary
 - Bounds are loose
 - Needs prior knowledge
- $\|v^* - v_\pi\|_{1,\alpha} \leq \dots + \bar{u}_\pi^\top [\tilde{v} - L\tilde{v}]_+$
 - Often does not work
- Why use ALP?
 - Easy to solve & analyze
 - Sometimes works well



Blood inventory management

Approximate Bilinear Programming

- Approximate policy iteration
- Approximate linear program
- Approximate bilinear program



Approximate Bilinear Programming: Derivation *

- Formulations for *all* online error bounds
- *Robust* policy loss

$$\|v^* - v_{\pi}\|_{\infty} \leq \frac{1}{1-\gamma} \|\tilde{v} - L\tilde{v}\|_{\infty}$$

- *Expected* policy loss (**no** prior knowledge)

$$\|v^* - v_{\pi}\|_{1,\alpha} \leq \alpha^T(v^* - \tilde{v}) + \frac{1}{1-\gamma} \|\tilde{v} - L\tilde{v}\|_{\infty}$$

- *Expected* policy loss (prior knowledge)

$$\|v^* - v_{\pi}\|_{1,\alpha} \leq \alpha^T(v^* - \tilde{v}) + \bar{u}_{\pi}^T [\tilde{v} - L\tilde{v}]_+$$

Approximate Bilinear Programming *

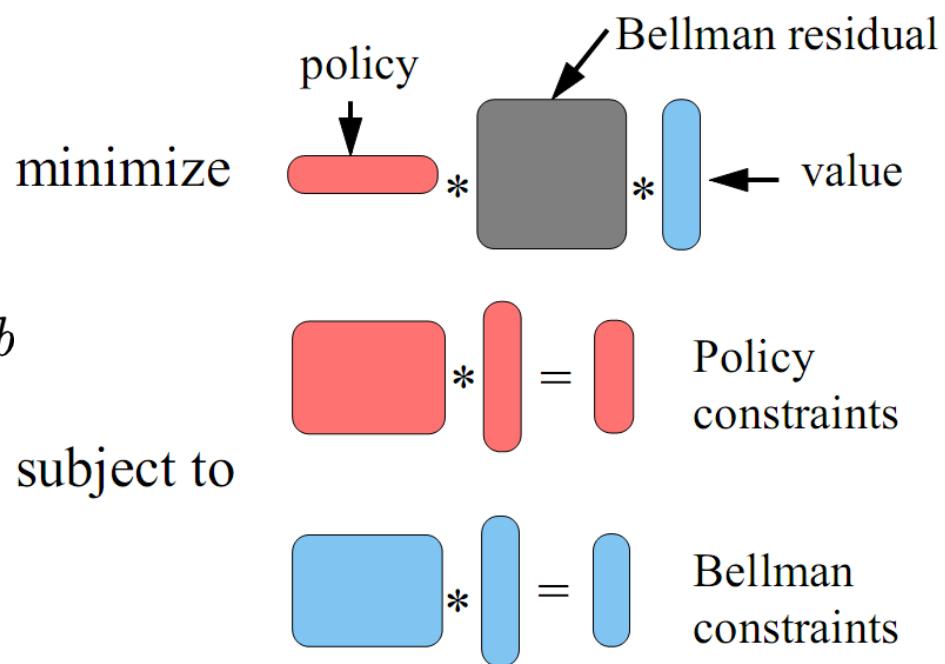
$$\min_{\pi | \lambda, \lambda', v} \pi^T \lambda + \lambda'$$

s.t. $B\pi = 1$ $A\mathbf{v} - b \geq \mathbf{0}$

$$\pi \geq \mathbf{0} \quad \lambda + \lambda' \mathbf{1} \geq A\mathbf{v} - b$$

$$\lambda, \lambda' \geq \mathbf{0}$$

$$v \in \mathcal{M}$$



ABP: Offline Error Bounds *

- Approximate Bilinear Programming

$$\|v_\pi - v^*\|_\infty \leq \frac{2}{1 - \gamma} \min_{v \in \mathcal{M}} \|Lv - v\|_\infty$$

- Approximate Linear Programming

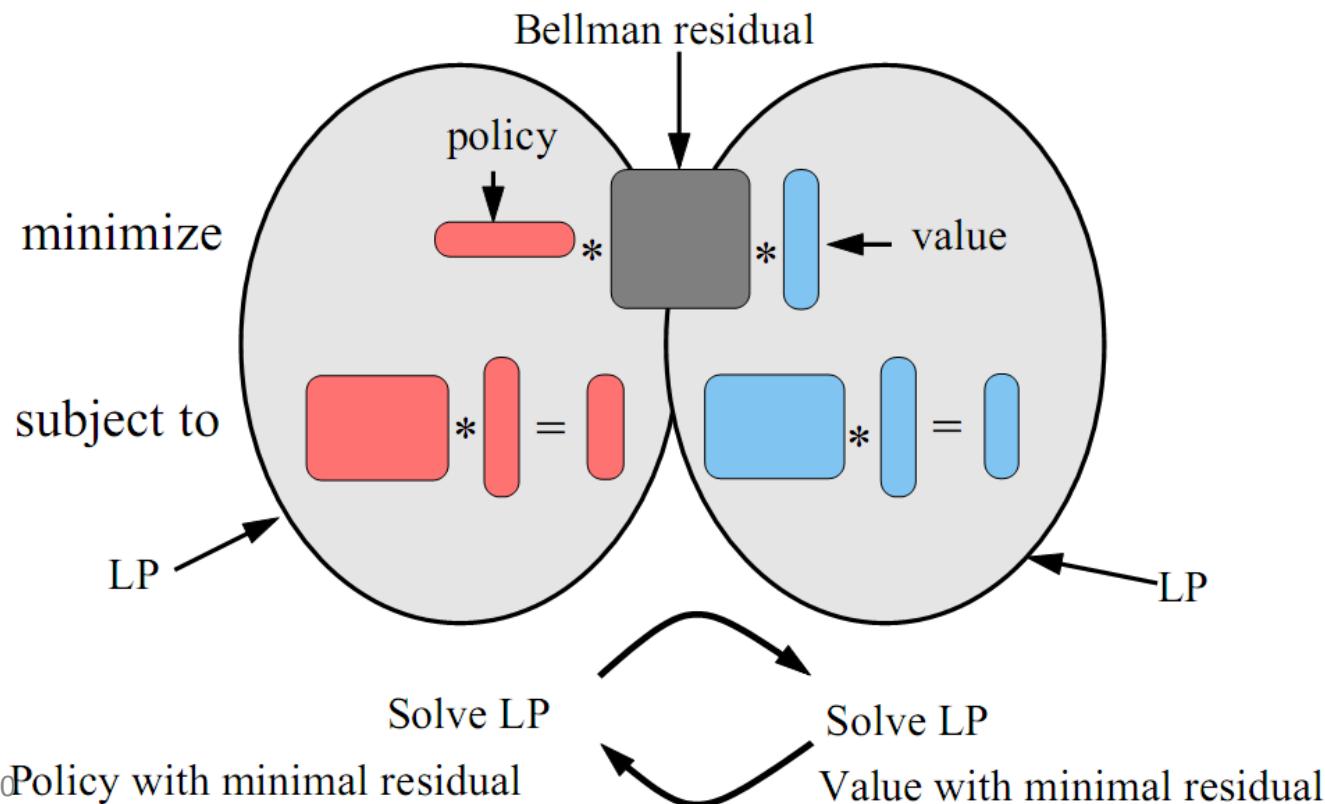
$$\|v_\pi - v^*\|_\infty \leq \frac{(2 + \gamma)|\mathcal{S}|}{(1 - \gamma)^2} \min_{v \in \mathcal{M}} \|v^* - v\|_\infty$$

- Approximate Policy Iteration

$$\limsup_{k \rightarrow \infty} \|v_{\pi^k} - v^*\|_\infty \leq \frac{2}{(1 - \gamma)^2} \limsup_{k \rightarrow \infty} \|\tilde{v}_k - v_k\|_\infty$$

Solving Approximate Bilinear Programs

- Value function approximation is NP hard
- Simple approximate algorithm

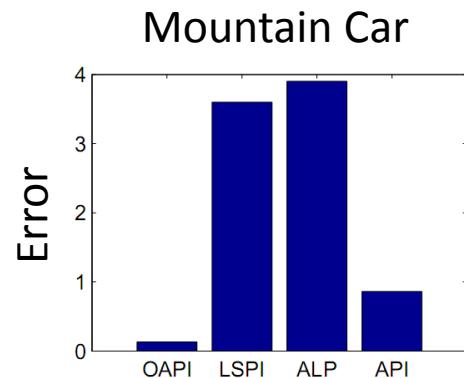


Approximate Bilinear Programming *

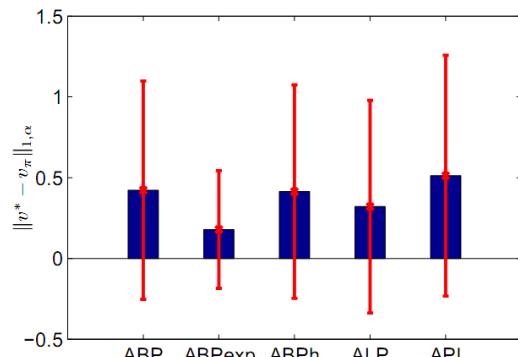
- ABP Guarantees
 - Tight value function approximation
 - Convergence
 - Match performance of API

- ABP, ALP works well in
 - Benchmark problems
 - Reservoir management

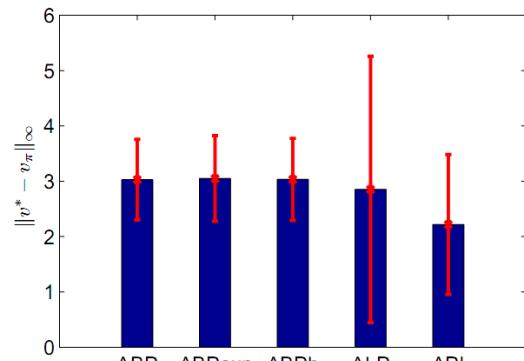
- Opportunity for better
 - Algorithms
 - Analysis



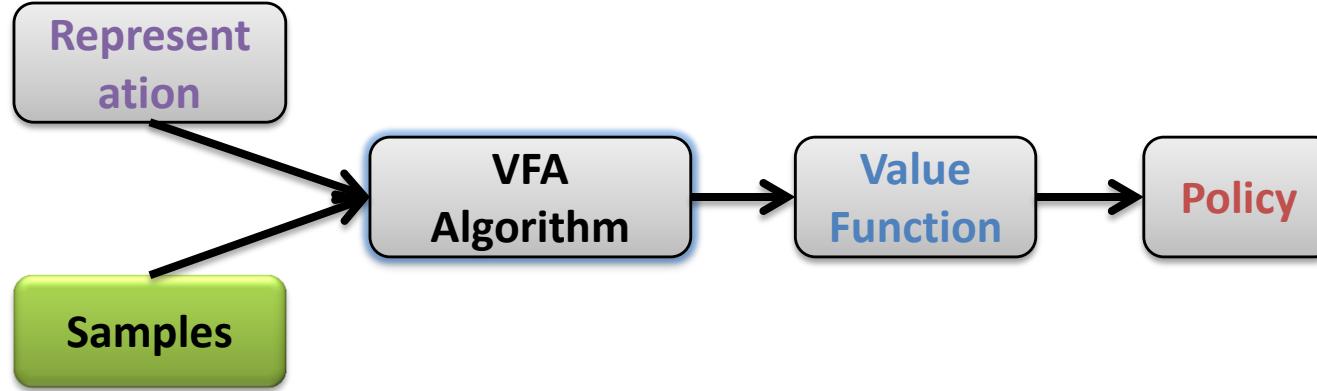
Benchmark Chain Problem



Expected Policy Loss



Robust Policy Loss



SAMPLING ERROR

Solution Based On Samples

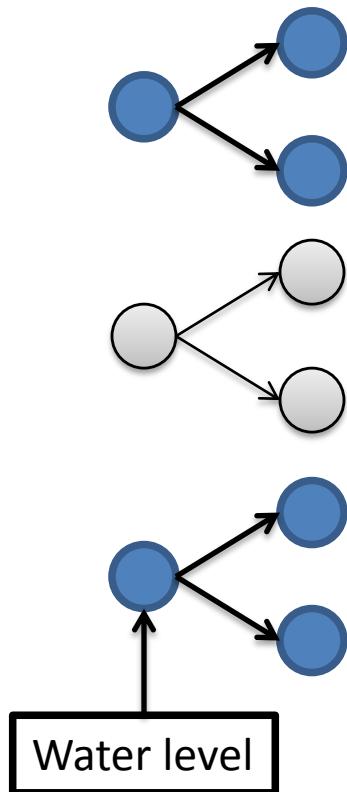
- Problems with many states
- Reservoir management
 - State = water level 
 - Would have solve the problem for **all** water levels
 - ALP: One constraint per state
- **Challenge:** Bound policy loss due to sampling
- No practical bounds exist
 - Too loose or impossible to compute

$$\begin{array}{l} \text{minimize} \quad \text{[gray rectangle]} * \text{[blue vertical bar]} \\ \text{subject to} \quad \text{[blue vertical bar]} * \text{[blue vertical bar]} = \text{[blue vertical bar]} \end{array}$$

Types Of Sampling Error

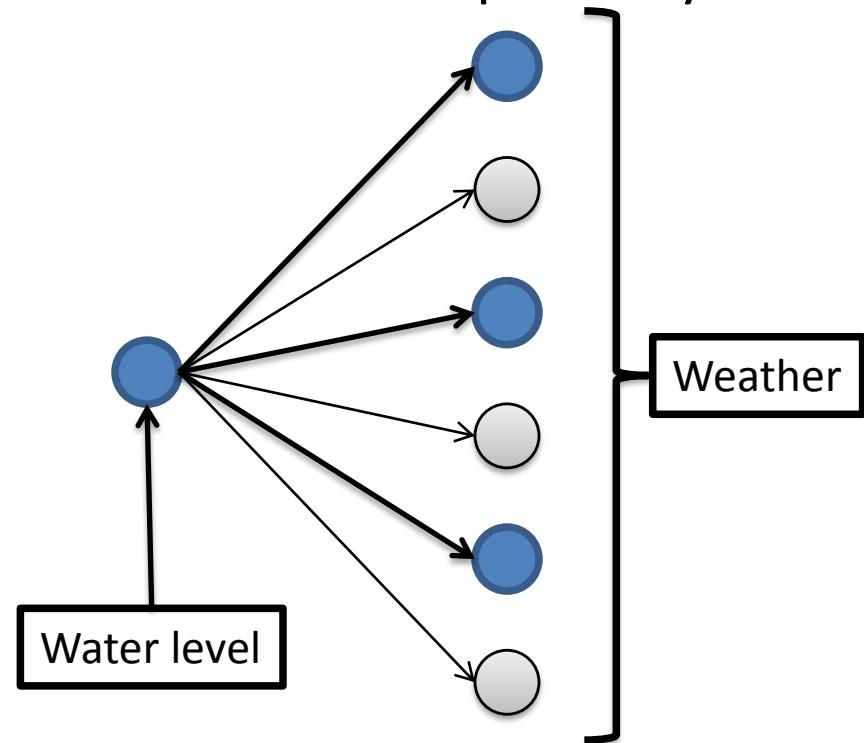
State selection error

- States that are not sampled



Transition estimation error

- States that are sampled, but are not known precisely



Bounding Sampling Error *

- **Approach:** Bounds using the simple analysis of optimization-based VFA
- Regularized representable value functions

$$\mathcal{M} = \{\textcolor{blue}{v} = \Phi x \mid \|x\|_{1,e} \leq \textcolor{green}{\psi}\}$$

$\epsilon_p(\textcolor{green}{\psi})$ — State selection error: how different are unsampled states

$\epsilon_s(\textcolor{green}{\psi})$ — Transition estimation error: how different are state transitions from the true value

Sampling: Offline Error Bounds *

- Bounds on policy loss

- ALP

$$\|\tilde{v} - v^*\|_{1,c} \leq \frac{2}{1-\gamma} \min_{v \in \mathcal{M}} \|v - v^*\|_\infty + 2\epsilon_c(\psi) + \frac{3\epsilon_s(\psi) + 2\epsilon_p(\psi)}{1-\gamma}$$

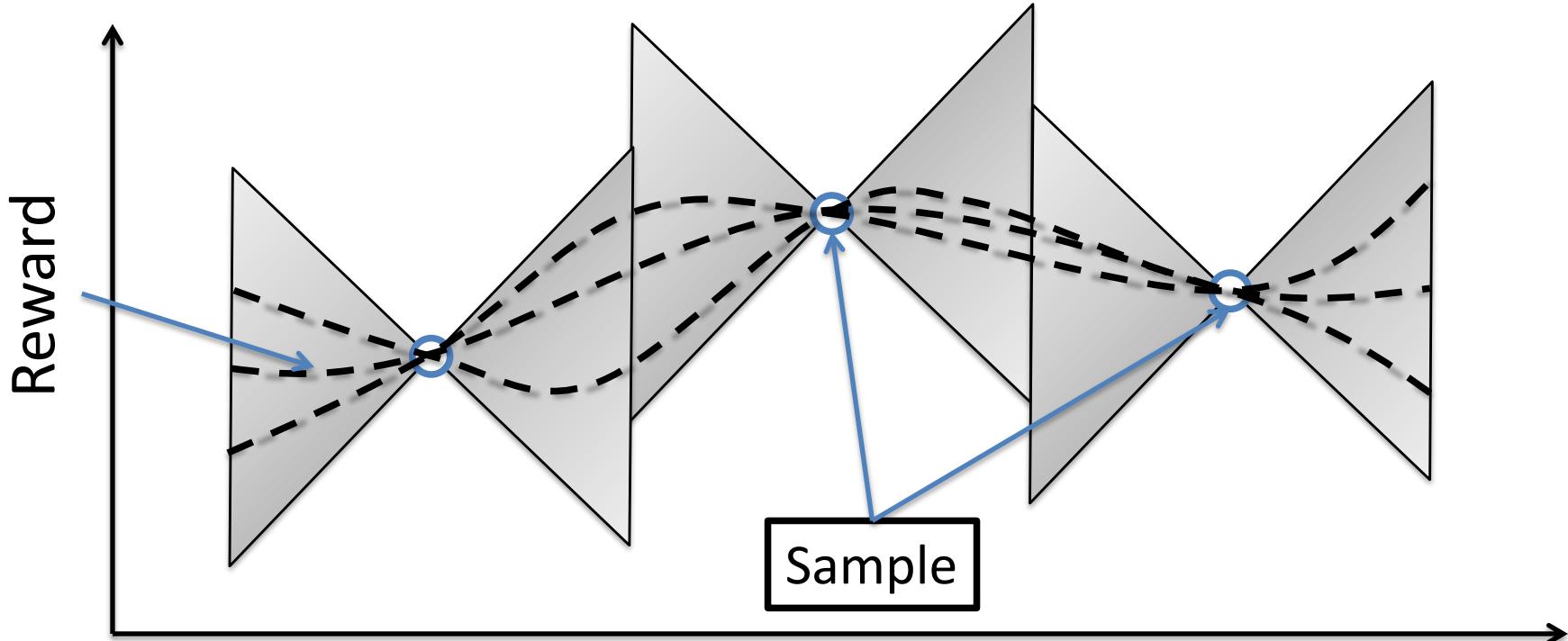
- ABP

$$\|\tilde{v} - L\tilde{v}\|_\infty \leq \min_{v \in \mathcal{M}} \|v - Lv\|_\infty + 2\epsilon_s(\psi) + \epsilon_p(\psi)$$

- Assumptions too general to be useful directly
 - How to estimate them?

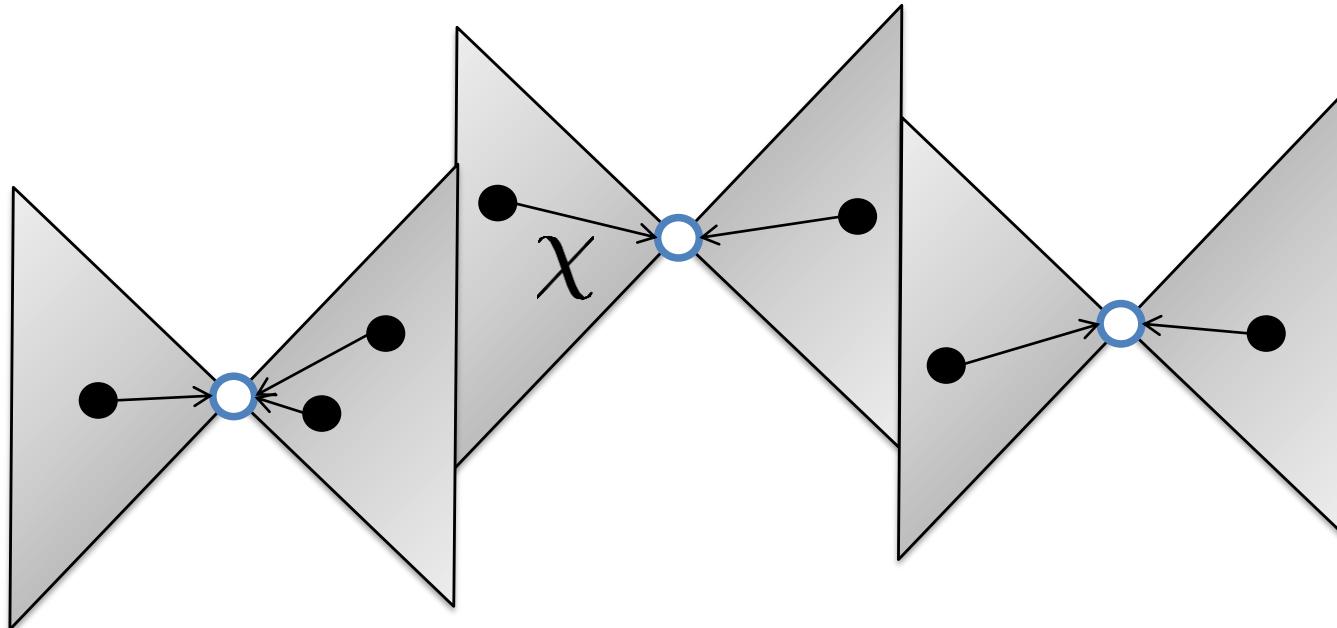
State Selection Error: Local Modeling Assumption

- States that are close are similar
- Seen a Lipschitz continuity



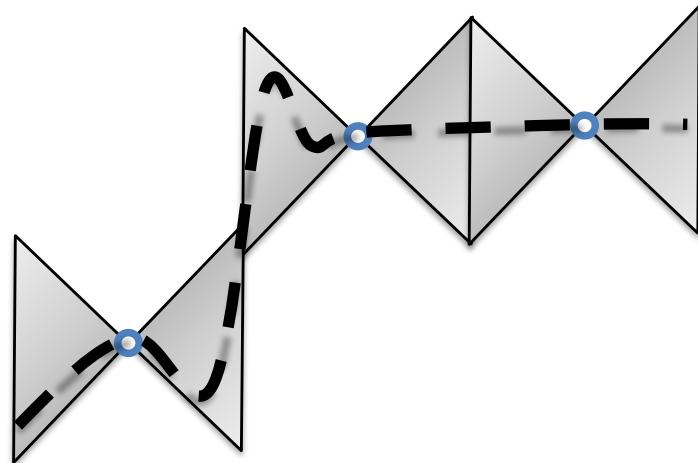
Basic Approach *

- How to use that samples are known?
- Map the states to samples $\chi : \mathcal{S} \rightarrow \Sigma$



Problems with Local Modeling

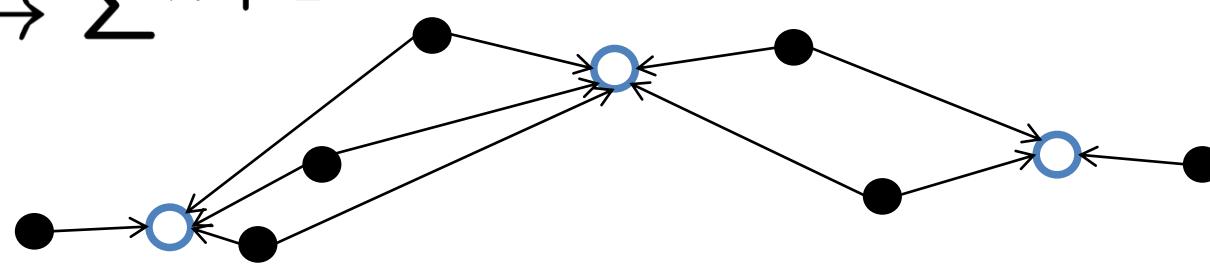
- Assumption must apply to the full state space
 - Bounds are too loose to be useful
 - Functions tend to be constant with small discontinuities
- Hard to estimate the similarity



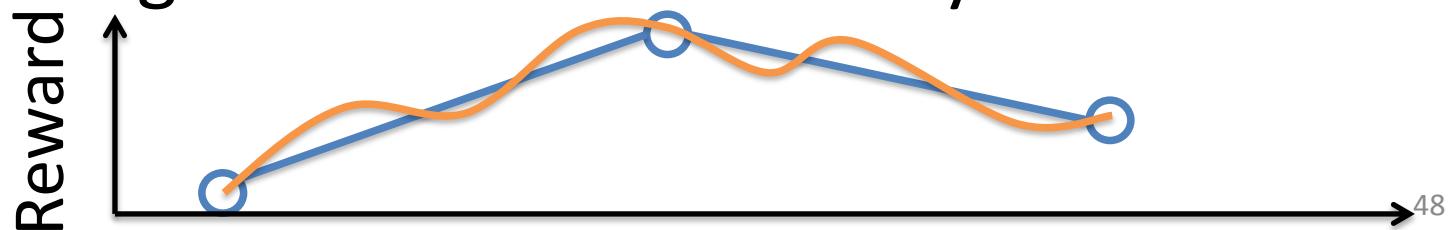
Better Bounds *

- Use the global structure of the problem
 - Usually more than a single sample available
 - Map a state to multiple samples

$$\chi : \mathcal{S} \rightarrow \Sigma^{n+1}$$



- Linear segments do not incur any error

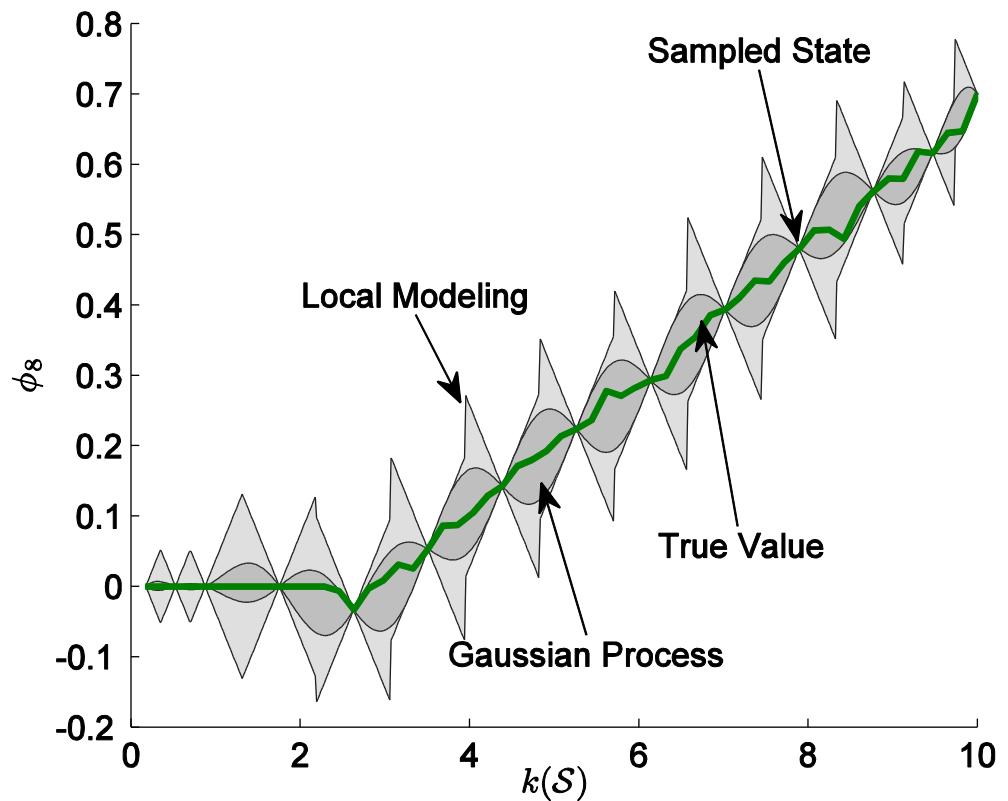


Bayesian Regression Assumption *

- Local modeling:
 - Worst-case guarantees: pessimistic assumptions
 - Local analysis
- Use Bayesian regression to generalize to unsampled states
 - Take advantage of global structure
 - Expectation instead of hard bounds – easy to specify
- Drawback:
 - Probabilistic guarantees

Comparison To Local Modeling *

- Gaussian processes
- Tight bounds (10-100x tighter than LM)
- Flexible assumptions
- Can be used in practice



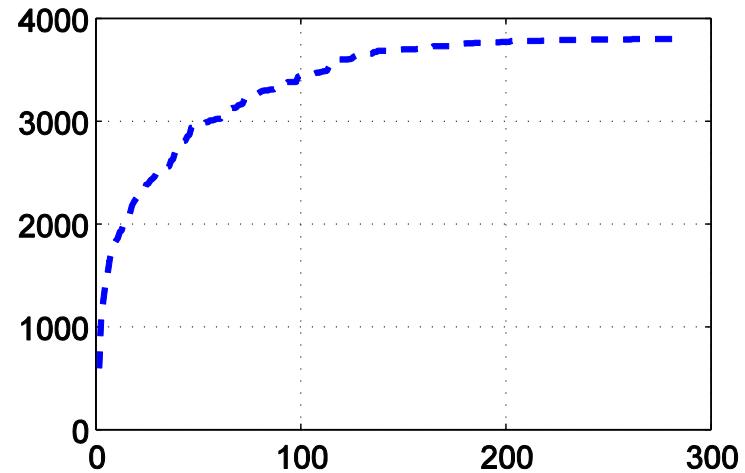
Transition Estimation Error *

- Offline error bound

$$P[\epsilon_s(\psi) > \epsilon] \leq 2|\tilde{\Sigma}|_a |\phi| \exp\left(\frac{2(\epsilon/(\psi \cdot \gamma))^2}{M_\phi n}\right)$$

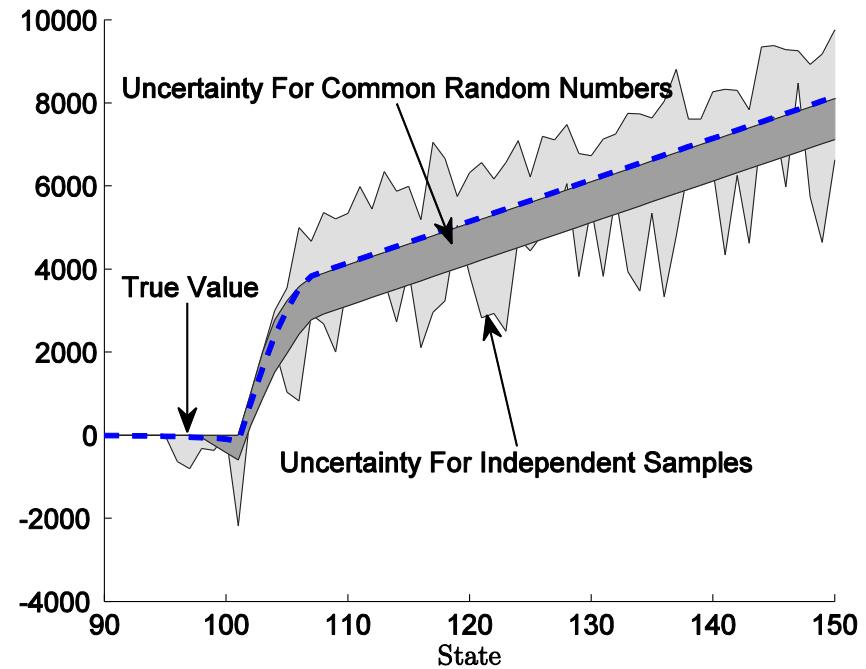
- Depends on the number of samples $|\tilde{\Sigma}|_a$

- The bound is tight
- Need many weather patterns need to be sampled per volume



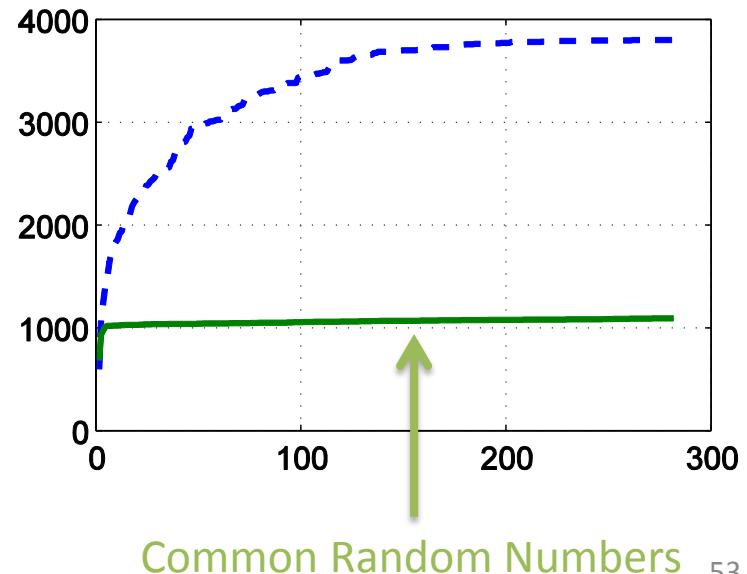
Common Random Numbers

- The bounds are tight because state are estimated independently
- **Idea:** Use the same weather to evaluate every volume



Common Random Numbers *

- Growth function $\tau(m)$
 - How similar is behavior for all kinds of volumes
 - Independent of number of samples
- Similar concept to VC dimension



Sampling Error

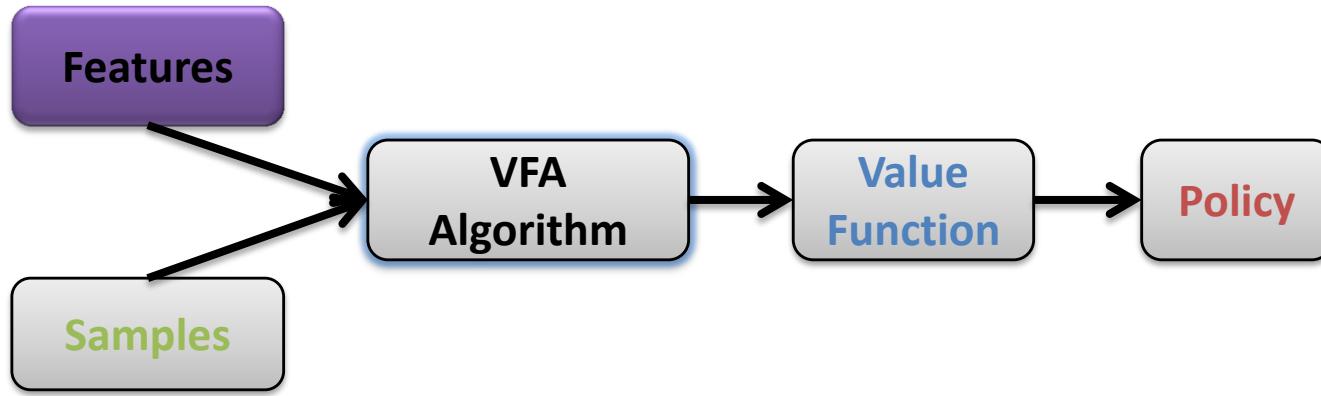
- Approximate Policy Iteration [Farahmand et al. 2009]

$$\limsup_{n \rightarrow \infty} \|v^* - v_n\|_\infty \leq \limsup_{n \rightarrow \infty} \frac{2\gamma}{(1 - \gamma)^2} \left(c \times C_{\rho, \nu} \max_{i \leq n} \|\epsilon_i\|_{\rho, \nu} + \dots \right)$$

- Theoretically interesting
- Hard/impossible to use in practice
- Approximate Bilinear Programming

$$\|\tilde{v} - L\tilde{v}\|_\infty \leq \min_{v \in \mathcal{M}} \|v - Lv\|_\infty + 2\epsilon_s(\psi) + \epsilon_p(\psi)$$

- Easy to use general properties
- Can be used in practice (limited)



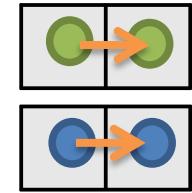
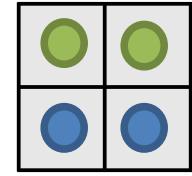
FEATURE SELECTION

Choosing Features

- Must approximate the value function
- Need a small number of features
 - Optimization problems easy to solve
 - Minimizes sampling error
- Features are hard to choose
 - Must know which states are important

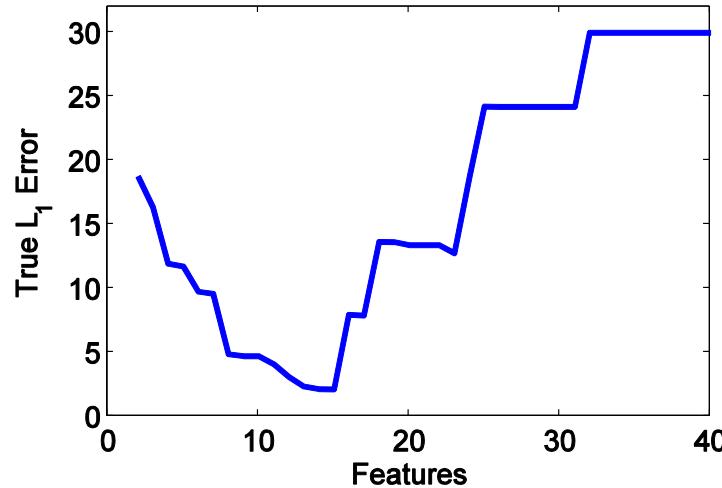
Possible binary features in reservoir management:

$$l \in [1, 6] \ l \in [7, 10] \ l \in [3, 12] \ l \in [10, 15] \dots$$



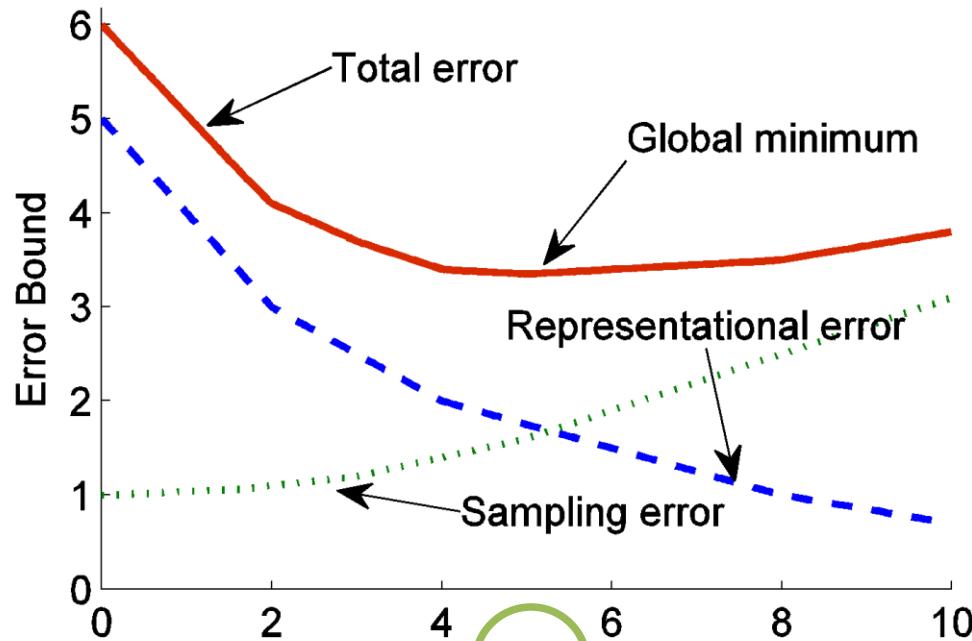
Number of Features and Solution Quality *

Solution quality



Automatically choose **features** based on
sampling bounds

Selecting Features To Balance Errors*



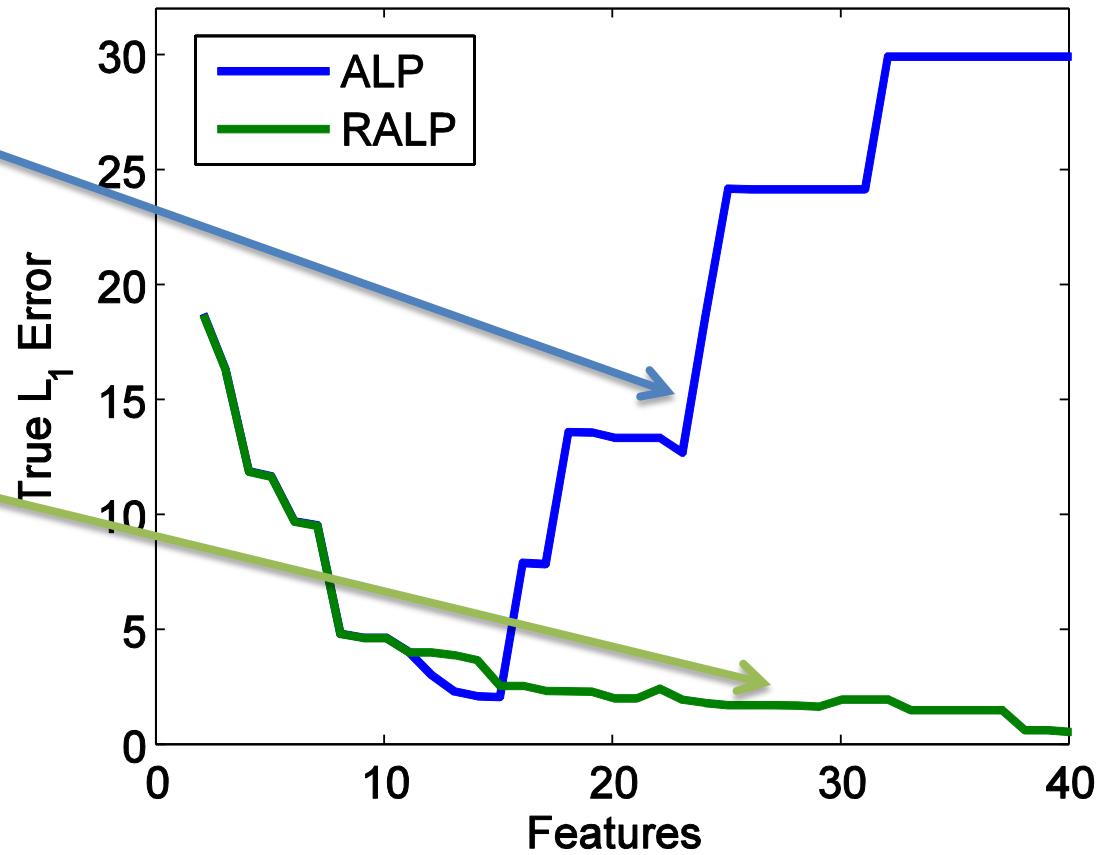
- Determine the global minimum
 - **Homotopy method**: efficiently calculate the solution for all values of ψ

Automatic Feature Selection

ALP

Regular ALP

Automatic
 ψ



Feature Selection

- When sampling bounds available
 - Can select appropriate features/regularization
- Performance does not decrease with more features
 - Flexibility in specifying features
- Outperforms other algorithms in benchmark problems

CONCLUSION

Conclusion

- Iterative algorithms have weak guarantees
 - Unreliable
 - Hard to analyze
- New & improved optimization-based algorithms
 - Decouple objective from algorithm
 - Strong guarantees
 - Easy to analyze and use
- (More) Practical sampling bounds: Use Gaussian processes
- Feature selection: Balance feature complexity with samples

Algorithms for Value Function Approximation

Classical Iterative Algorithms

- Based on MDP algorithms
- Simple algorithms
- Complex analysis
 - Weak guarantees
 - Hard to analyze
 - Hard to use

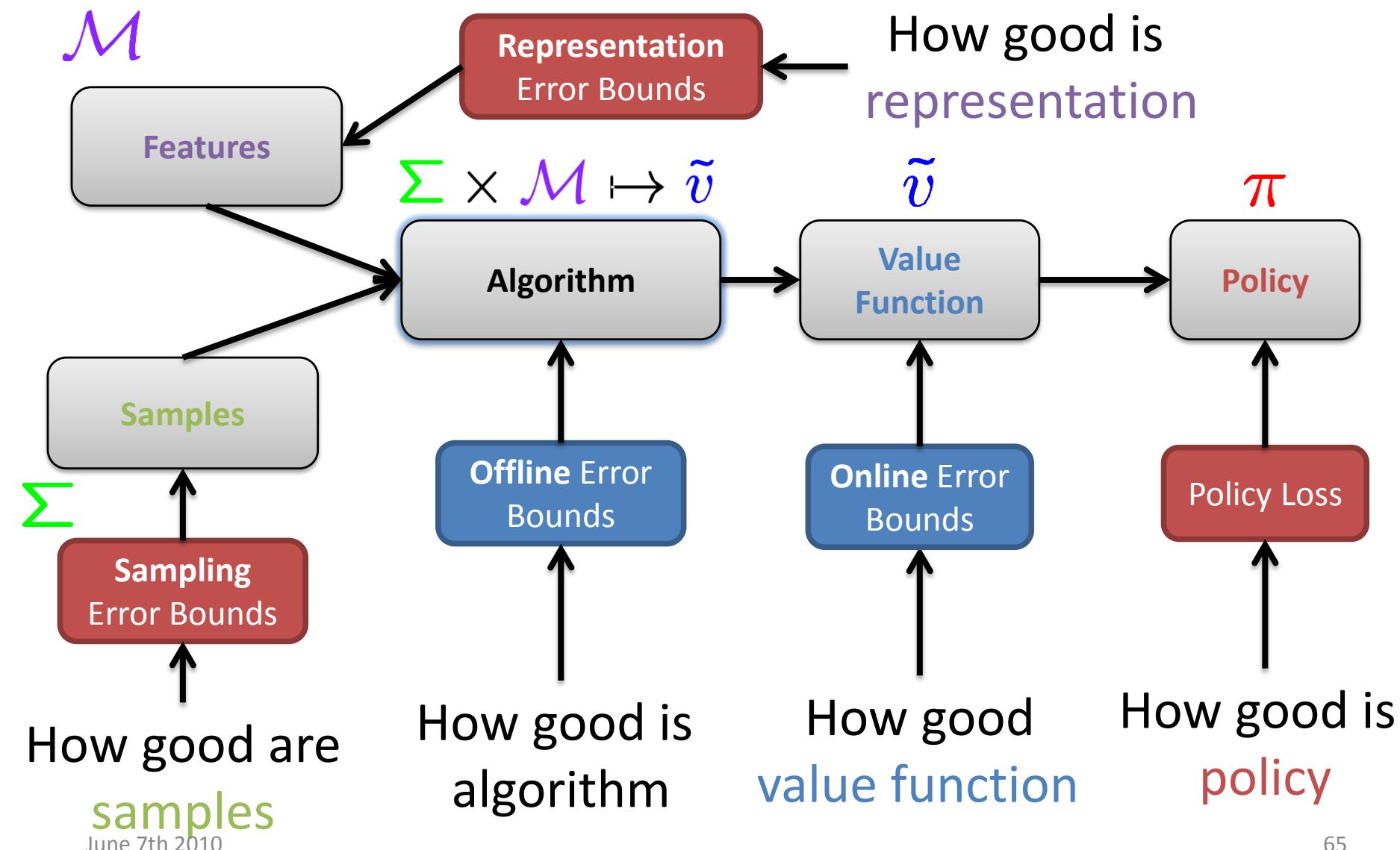
Optimization-based Algorithms

- Based on value-function bounds
- More complex algorithms
- Simple analysis
 - Strong guarantees
 - Sampling bounds
 - Feature selection

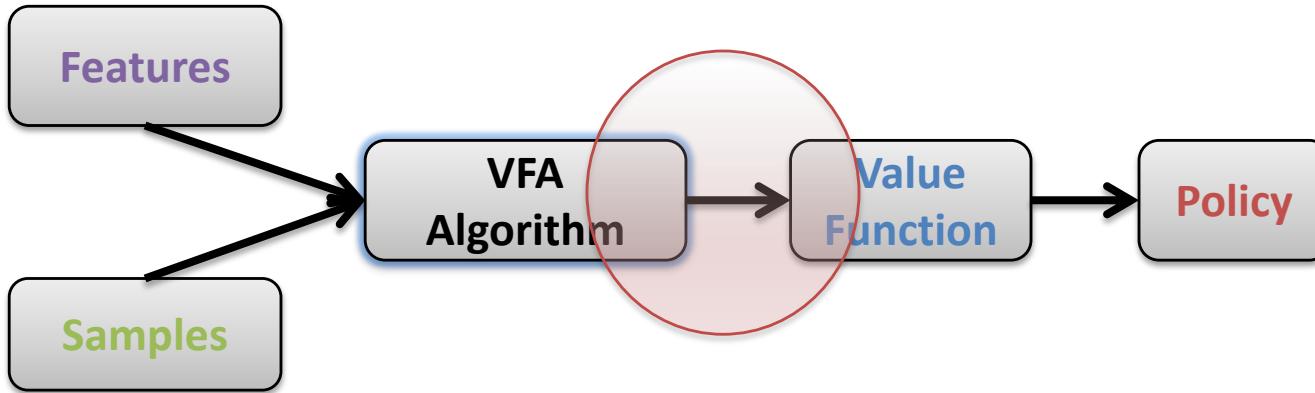
Contributions

- Analysis of API [*NIPS08, ECML/ML09*]
- New, robust VFA formulations [*ICAPS08*]
 - ALP: new derivation and formulation [*ICML09*]
 - ABP: robust/expected policy loss [*NIPS09, JMLR?*]
- Sampling bounds and feature selection
 - New better sampling bounds [*ICML10, NIPS?*]
 - Methods for feature selection [*IJCAI07, ICML10*]
- Mathematical optimization algorithms
 - Homotopy methods [*ICML10*]
 - Bilinear program solvers [*AAAI07, JAIR09*]

Future Work: Error Bounds in VFA



APPENDIX



MATHEMATICAL OPTIMIZATION ALGORITHMS

Algorithmic Considerations

- Optimization-based formulation require solving optimization problems
- Approximate linear programs
 - Solving large linear programs
 - Mature solvers that can solve large problems
- Approximate bilinear programs
 - NP hard to solve optimally
 - Few solvers are available

Solving Approximate Linear Programs

- Small number of features
 - Easy to solve

minimize  * 

subject to  *  = 

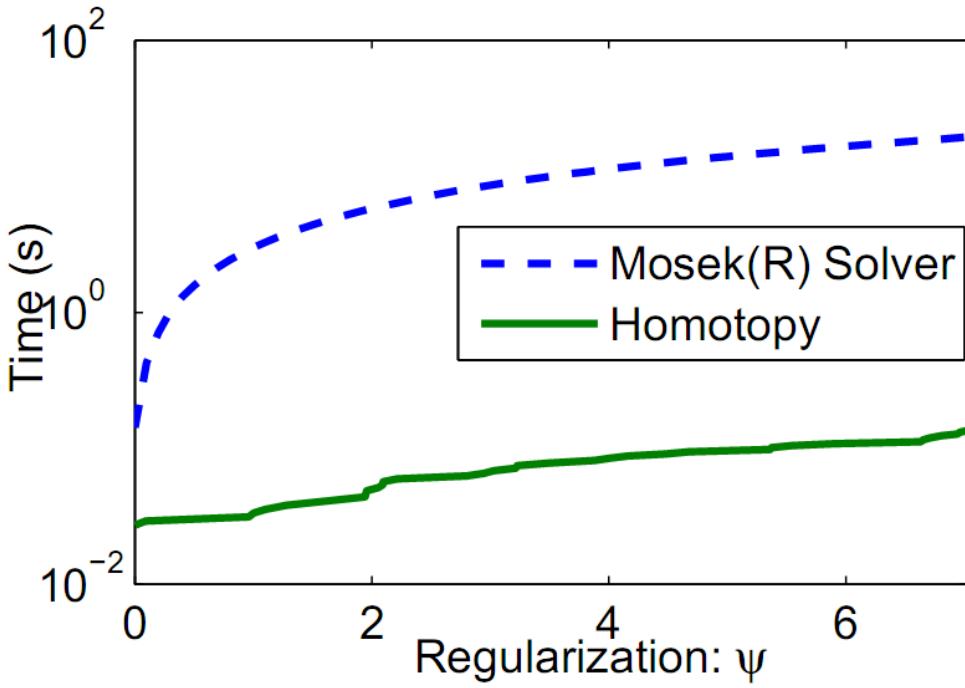
- Large number of features
 - Harder to solve

minimize  * 

subject to  *  = 

Homotopy Methods for Approximate Linear Programs *

- Solve for large ALPs with **regularization**



Important: Address degenerate solutions – dual regularization

$$\begin{aligned} \max_{y, \lambda} \quad & b^T y - \psi \lambda + \frac{1}{\chi} y^T y \\ \text{s.t.} \quad & A^T y - e \lambda \leq c \\ & y, \lambda \geq 0 \end{aligned}$$

$$\begin{aligned} \min_x \quad & c^T x + \chi \frac{1}{2} \| [Ax - b]_+ \|^2 \\ \text{s.t.} \quad & e^T x \leq \psi \\ & x \geq 0 \end{aligned}$$

Solving Bilinear Programs

- Few bilinear program solvers
- Easier to solve when the number of bilinear terms is small
 - Method for solving such BPs (*)
- An instance of a global optimization problem
 - Concave quadratic minimization
 - Mixed integer linear program
- Commercial solvers available

Standard MILP Transformation

$$\begin{array}{lll} \min_{w, \lambda_1, z, \lambda_2, q_1, q_2} & s_2^T z + \lambda_1^T b_1 \\ \text{s.t.} & A_1 A_2^T \lambda_2 - A_1 r_2 + B_1 w \geq b_1 & A_2 A_1^T \lambda_1 - A_2 r_1 + B_2 z \geq b_2 \\ & A_1^T \lambda_1 = r_1 + C y & A_2^T \lambda_2 = r_2 + C^T x \\ & B_1^T \lambda_1 = s_1 & B_2^T \lambda_2 = s_2 \\ & A_1 A_2^T \lambda_2 - A_1 r_2 + B_1 w - b_1 \leq (1 - q_1)M \\ & A_2 A_1^T \lambda_1 - A_2 r_1 + B_2 z - b_2 \leq (1 - q_2)M \\ & \lambda_1 \leq M q_1 & \lambda_2 \leq M q_2 \\ & \lambda_1 \geq \mathbf{0} & \lambda_2 \geq \mathbf{0} \\ & q_1 \in \{0, 1\}^n & q_2 \in \{0, 1\}^n \end{array}$$

- Hard to analyze
- MILP solvers work very poorly

MILP Formulation of ABP *

$$\begin{array}{ll}\min_{z, \pi, \lambda, \lambda', v} & 1^T z + \lambda' \\ \text{s.t.} & z \geq \lambda - (\tau - \pi) \\ & \lambda + \lambda' 1 \geq A v - b \\ & A v - b \geq 0 \\ & B \pi = 1 \quad \lambda, \lambda' \geq 0 \\ & v \in \mathcal{M} \quad \pi \in \{0, 1\}^n\end{array}$$

- Orders of magnitude faster to solve
- Easier to analyze
- Still not fast enough

OTHER SLIDES

Types of Algorithm

- **Policy based:** *policy search*
 - Does not explicitly use a value function
 - Great results, but hard to analyze
- **Value function based:** *approximate dynamic programming*
 - Calculate value function as an intermediate step
 - Stronger guarantees, good performance

All efficient methods for sequential decision problems estimate value function as an intermediate step.

Richard Sutton, MSRL 2009

Lower Bounds on API Performance *

- A trivial bound can be tighter for sufficiently high discount factors

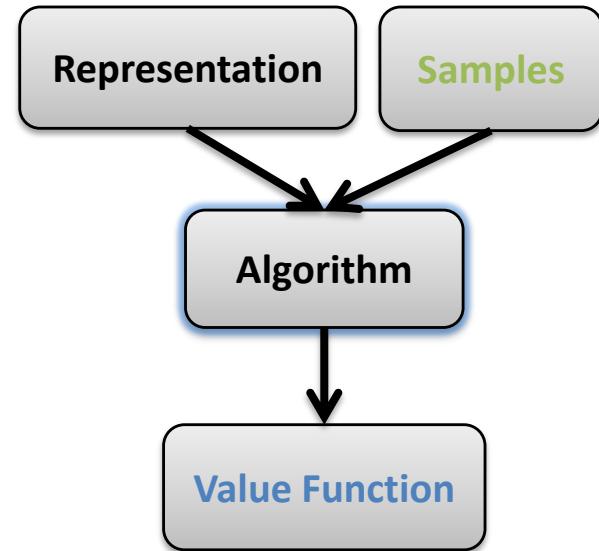
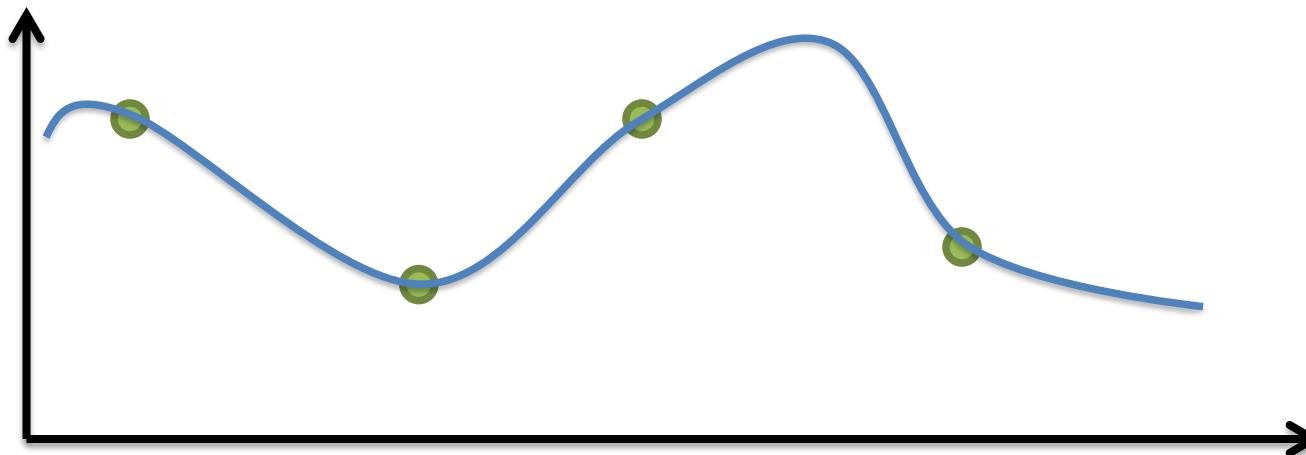
$$\limsup_{k \rightarrow \infty} \|v^* - v_{\pi^k}\|_\infty \leq \frac{c}{1 - \gamma} < \frac{2}{(1 - \gamma)^3} \limsup_{k \rightarrow \infty} \|\tilde{v}_k - L\tilde{v}_k\|_\infty$$

- Impossible to get bounds in terms of the best approximation

$$\limsup_{k \rightarrow \infty} \|v^* - v_{\pi^k}\|_\infty > c \min_{v \in \mathcal{M}} \|v - Lv\|_\infty$$

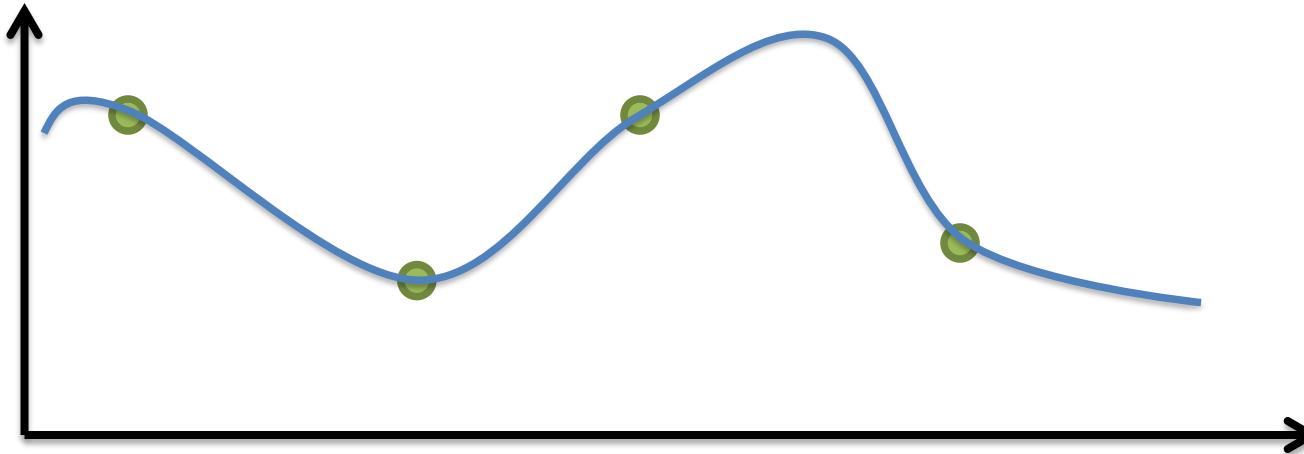
Algorithms for VF Approximation

- Combines features and samples
- Resembles machine learning regression

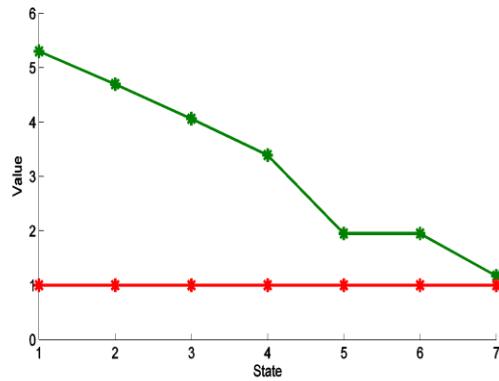
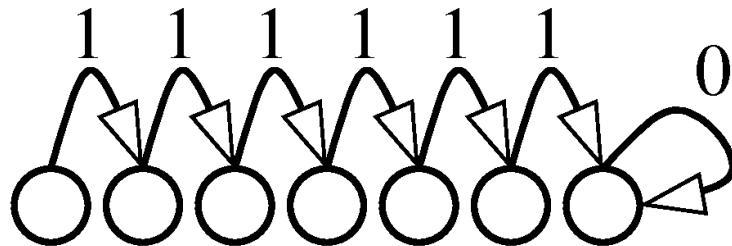


Relationship To Machine Learning

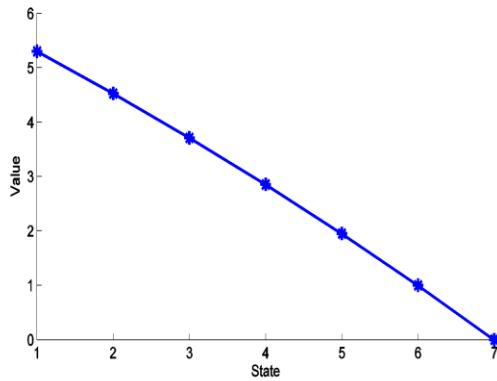
- No distribution in regression
- Distribution depends on the policy
 - policy depends on the value function
- The guarantees must be different



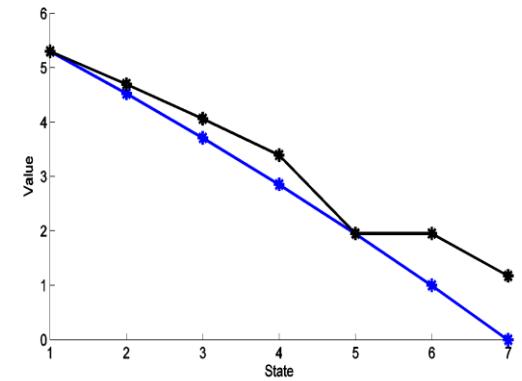
Value Function Approximation



Features

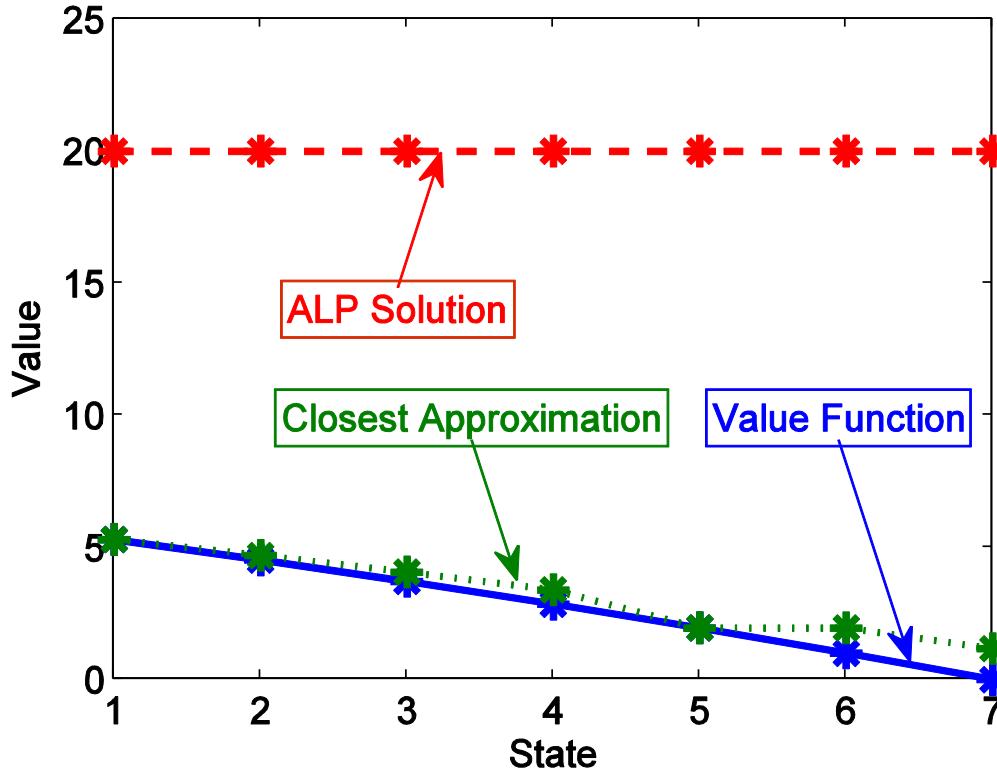


Optimal
Value Function



Approximate
Value Function

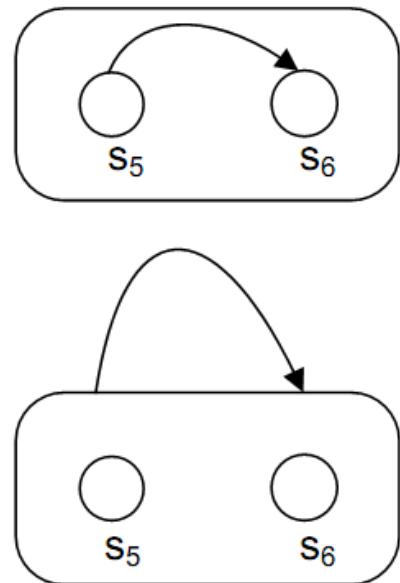
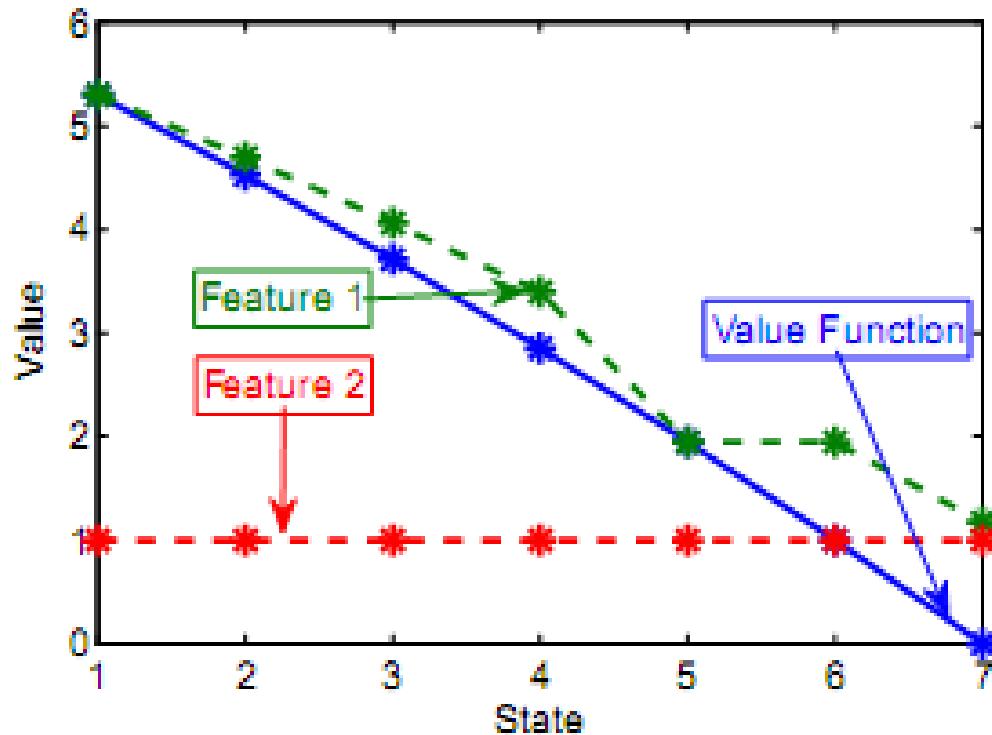
The Bound Is Tight *



- Modify ALP to reduce the bound
 - Improve practical performance

Why Approximation Fails? *

- Virtual Loop

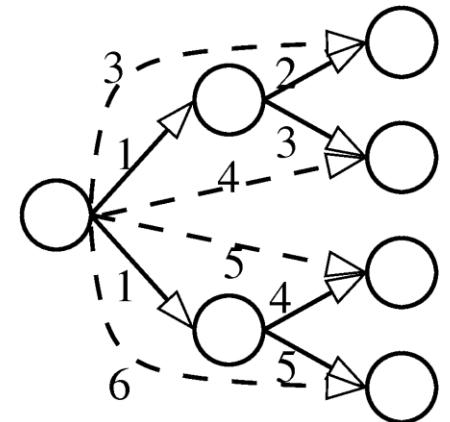


Expanding Constraints *

- Roll out selected constraints
 - Use dual values
- Offline error bounds

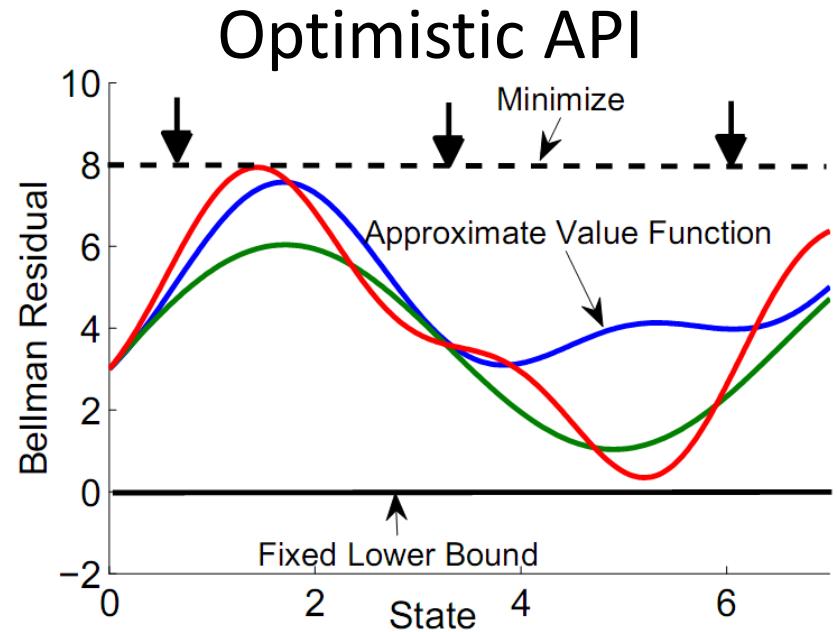
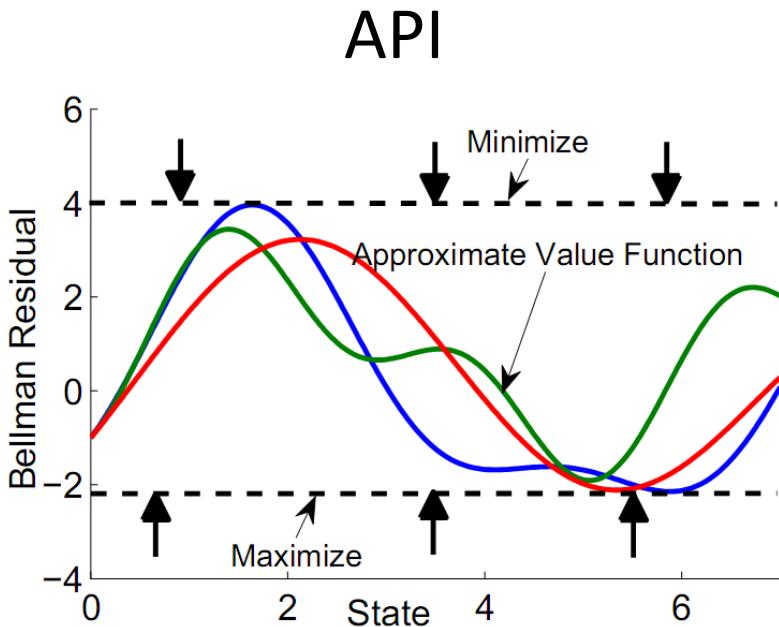
$$\|\tilde{v}_{\textcolor{blue}{t}} - v^*\|_1 \leq \frac{2}{1 - \gamma^{\textcolor{blue}{t}}} \min_{v \in \mathcal{M}} \|v^* - v\|_\infty$$

$$\gamma = 0.99 \quad \frac{1}{1 - \gamma} = 100 \quad \textcolor{blue}{t} = 10 \quad \frac{1}{1 - \gamma^{\textcolor{blue}{t}}} = 10.5$$



Simple Approximate Algorithm for ABP

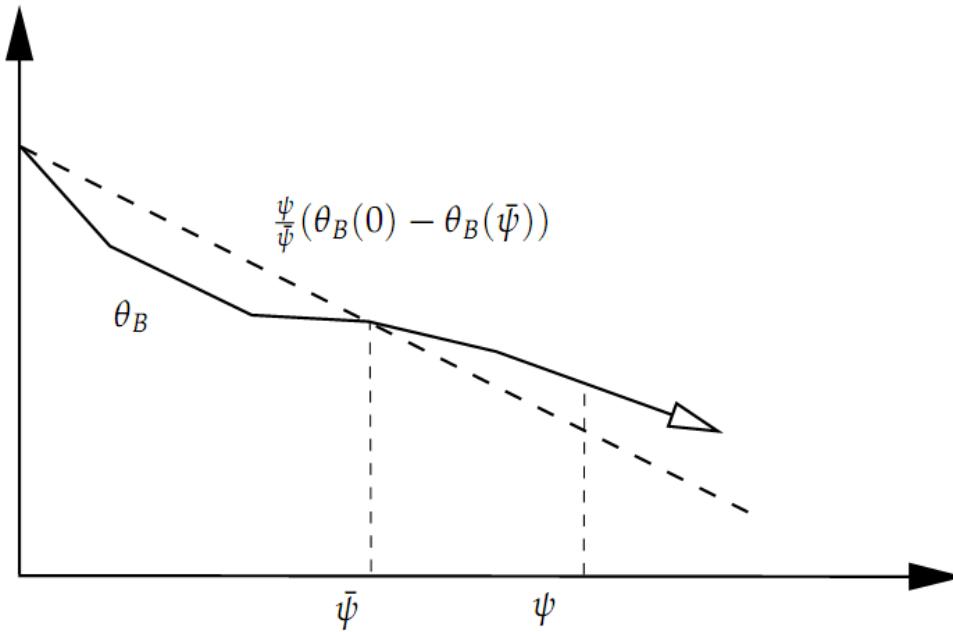
- Motivates approximate policy iteration
- Converges



Automatic Feature Selection

ABP

- ALP relies on convexity
- ABP is NOT convex
 - But can use a similar property



Future Work: Modeling

- Data is used directly
 - Not enough data
 - Incorporating prior knowledge?
- Interaction of models and optimization
 - Integrate machine learning components

Future Work: Optimization-based Algorithms

- Optimization-based ADP methods in practice
 - Performance?
 - What is a good value function?
 - Implications for iterative algorithms?
- Algorithms for linear/bilinear programs
- Tighter bounds
 - Better ways to samples