The Linear Programming Approach to Approximate Dynamic Programming

Daniela Pucci de Farias
(joint work with Ben Van Roy)

Massachusetts Institute of Technology
Outline

- Markov decision processes
- Approximate Dynamic Programming
- Approximate linear programming
- Performance and Error Analysis
- Constraint Sampling
Markov Decision Processes

- (finite) state space $S$
Markov Decision Processes

- (finite) state space $S$
- (finite) action sets $A_x$
Markov Decision Processes

- (finite) state space $S$
- (finite) action sets $A_x$
- costs $g_a(x)$
Markov Decision Processes

- (finite) state space $S$
- (finite) action sets $A_x$
- costs $g_a(x)$
- transition probabilities $P_a(x, y)$
Markov Decision Processes

- (finite) state space $S$
- (finite) action sets $A_x$
- costs $g_a(x)$
- transition probabilities $P_a(x, y)$
- discount factor $\alpha$
Markov Decision Processes

- (finite) state space $S$
- (finite) action sets $A_x$
- costs $g_a(x)$
- transition probabilities $P_a(x, y)$
- discount factor $\alpha$
- Minimize $E \left[ \sum_{t=0}^{\infty} \alpha^t g_{a(t)}(x(t)) \right]$
Tetris

- \( x \in S \): wall configuration and current piece
- \( a \in A_x \): Piece placement
- \( P_a(x, \cdot) \): Distribution of next piece
- \( g_a(x) \): number of rows eliminated
Examples

- Scheduling/routing in queueing networks
- Dynamic resource allocation
- Asset allocation/risk management
- Power management in devices
Dynamic Programming

- Bellman’s equation

\[ J(x) = \min_{a \in A_x} E \left[ g_a(x) + \alpha J(y) \right] \]
Dynamic Programming

- Bellman’s equation

\[ J(x) = \min_{a \in A_x} E [g_a(x) + \alpha J(y)] \]

- Value iteration, policy iteration, linear programming
Dynamic Programming

- Bellman’s equation

\[ J(x) = \min_{a \in A_x} E \left[ g_a(x) + \alpha J(y) \right] \]

- Value iteration, policy iteration, linear programming

- Obtain an optimal policy

\[ u^*(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha J^*(y) \right] \]
Dynamic Programming

- Bellman’s equation

\[ J(x) = \min_{a \in A_x} E \left[ g_a(x) + \alpha J(y) \right] \]

- Value iteration, policy iteration, linear programming

- Obtain an optimal policy

\[ u^*(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha J^*(y) \right] \]

- The curse of dimensionality
Outline

- Markov decision processes
- Approximate Dynamic Programming
- Approximate linear programming
- Performance and error analysis
- Constraint Sampling
Value Function Approximation

- Approximate $J^* \approx \tilde{J}_r$, for some $r \in \mathbb{R}^K$
Value Function Approximation

- Approximate $J^* \approx \tilde{J}_r$, for some $r \in \mathbb{R}^K$
- Generate a policy

$$u(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha \tilde{J}_r(y) \right]$$
Value Function Approximation

- Approximate $J^* \approx \tilde{J}_r$, for some $r \in \mathbb{R}^K$
- Generate a policy

$$u(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha \tilde{J}_r(y) \right]$$

- Linearly parameterized approximators

$$\tilde{J}_r(x) = (\Phi r)(x) = \sum_{k=1}^{K} r(k) \phi_k(x)$$
Value Function Approximation

- Approximate $J^* \approx \tilde{J}_r$, for some $r \in \mathbb{R}^K$
- Generate a policy

$$u(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha \tilde{J}_r(y) \right]$$

- Linearly parameterized approximators

$$\tilde{J}_r(x) = (\Phi r)(x) = \sum_{k=1}^{K} r(k) \phi_k(x)$$

- Design a function approximator $\tilde{J}_r$
Value Function Approximation

- Approximate $J^* \approx \tilde{J}_r$, for some $r \in \mathbb{R}^K$
- Generate a policy

$$u(x) \in \arg\min_{a \in A_x} E \left[ g_a(x) + \alpha \tilde{J}_r(y) \right]$$

- Linearly parameterized approximators

$$\tilde{J}_r(x) = (\Phi r)(x) = \sum_{k=1}^{K} r(k) \phi_k(x)$$

- Design a function approximator $\tilde{J}_r$
- Compute parameters $r \in \mathbb{R}^K$ so that $\tilde{J}_r \approx J^*$
Tetris

- 22 features / basis functions
  - Column heights
  - Differences between heights of consecutive columns
  - Maximum height
  - Number of holes
  - Constant function
Approximate DP: Examples

- American options pricing
  (Longstaff & Schwartz, 2001, Tsitsiklis & Van Roy, 2001)

- Job-shop scheduling
  (Zhang & Dietterich, 1996)

- Elevator scheduling
  (Crites & Barto, 1996)

- Backgammon
  (Tesauro, 1995)
Outline

- Markov decision processes
- Approximate Dynamic Programming
- Approximate linear programming
- Performance and error analysis
- Constraint Sampling
LP Formulation of DP

\[
\begin{align*}
\max_J & \quad \sum_x c(x)J(x) \\
\text{s.t.} & \quad g_a(x) + \alpha \sum_y P_a(x, y)J(y) \geq J(x), \quad \forall x, \forall a
\end{align*}
\]
LP Formulation of DP

\[
\begin{align*}
\max_J & \quad \sum_x c(x)J(x) \\
\text{s.t.} & \quad g_a(x) + \alpha \sum_y P_a(x, y)J(y) \geq J(x), \ \forall x, \ \forall a
\end{align*}
\]

- \( J \leq J^* \) for all feasible \( J \)
LP Formulation of DP

\[
\max_J \sum_x c(x)J(x) \\
\text{s.t.} \quad g_a(x) + \alpha \sum_y P_a(x, y)J(y) \geq J(x), \ \forall x, \ \forall a
\]

- \( J \leq J^* \) for all feasible \( J \)
- LP solution is \( J^* \) for all \( c > 0 \)
LP Formulation of DP

\[
\max_J \sum_x c(x)J(x)
\]

s.t. \( g_a(x) + \alpha \sum_y P_a(x, y)J(y) \geq J(x), \forall x, \forall a \)

- \( J \leq J^* \) for all feasible \( J \)
- LP solution is \( J^* \) for all \( c > 0 \)
- one variable per state
- one constraint per state-action pair
Approximate Linear Programming

\[
\max_J \sum_x c(x) J(x)
\]

s.t. \( g_a(x) + \alpha \sum_y P_a(x, y) J(y) \geq J(x), \forall x, \forall a \)
Approximate Linear Programming

\[
\max_J \sum_x c(x) J(x)
\]

\[
s.t. \quad g_a(x) + \alpha \sum_y P_a(x, y) J(y) \geq J(x), \forall x, \forall a
\]

- 
  Idea: Consider only solutions \( J = \Phi r \)
Approximate Linear Programming

\[
\max_r \sum_x c(x)(\Phi_r)(x)
\]

\[
\text{s.t. } g_a(x) + \alpha \sum_y P_a(x, y)(\Phi_r)(y) \geq (\Phi_r)(x), \forall x, \forall a
\]

Idea: Consider only solutions \( J = \Phi_r \)
Approximate Linear Programming

\[
\max_r \sum_x c(x)(\Phi r)(x)
\]

s.t. \[ g_a(x) + \alpha \sum_y P_a(x,y)(\Phi r)(y) \geq (\Phi r)(x), \forall x, \forall a \]

- Idea: Consider only solutions \( J = \Phi r \)
- one variable per basis function
- one constraint per state-action pair
Approximate Linear Programming

\[
\max_r \sum_x c(x)(\Phi_r)(x)
\]

s.t. \( g_a(x) + \alpha \sum_y P_a(x, y)(\Phi_r)(y) \geq (\Phi_r)(x), \forall x, \forall a \)

- Idea: Consider only solutions \( J = \Phi_r \)
- one variable per basis function
- one constraint per state-action pair
  \( \Rightarrow \) efficient constraint sampling
Some History

- early work
  - Schweitzer and Seidmann (1985)
  - Trick and Zin (1993, 1997)
  - Gordon (1995)
Some History

- early work
  - Schweitzer and Seidmann (1985)
  - Trick and Zin (1993, 1997)
  - Gordon (1995)

- analytical and computational tool
  - Morrison and Kumar (1999)
  - Paschalidis and Tsitsiklis (2000)
  - Adelman (2002)
Some History

- early work
  - Schweitzer and Seidmann (1985)
  - Trick and Zin (1993, 1997)
  - Gordon (1995)

- analytical and computational tool
  - Morrison and Kumar (1999)
  - Paschalidis and Tsitsiklis (2000)
  - Adelman (2002)

- more extensive analysis and implementation in large problems
  - Schuurmans and Patrascu (2001)
  - de Farias and Van Roy (2001, 2002)
  - Guestrin et al. (2002)
  - Poupart et al. (2002)
Outline

- Markov decision processes
- Approximate Dynamic Programming
- Approximate linear programming
- Performance and error analysis
- Constraint Sampling
Theory on Value Function Approximation

- Goals
  - Understand what algorithms are doing
  - Figure out which variations work and when
  - Reduce trial and error
  - Improve performance

© Will my algorithm compute weights \( \tilde{r} \) that make good use of my basis functions?

"Competitive" bound

If \( r \) can come within \( \varepsilon \) of \( J \), then algorithm A will compute \( \tilde{r} \) such that

1. \( \tilde{r} \) is within \( O(\varepsilon) \) of \( J \)
2. The greedy policy \( u \) is \( O(\varepsilon) \)-optimal
Theory on Value Function Approximation

- Goals
  - Understand what algorithms are doing
  - Figure out which variations work and when
  - Reduce trial and error
  - Improve performance

- Quality of ultimate approximation limited by choice of $\Phi$
Theory on Value Function Approximation

- Goals
  - Understand what algorithms are doing
  - Figure out which variations work and when
  - Reduce trial and error
  - Improve performance
- Quality of ultimate approximation limited by choice of $\Phi$
- Will my algorithm A compute weights $\tilde{r}$ that make good use of my basis functions $\Phi$?
Theory on Value Function Approximation

- **Goals**
  - Understand what algorithms are doing
  - Figure out which variations work and when
  - Reduce trial and error
  - Improve performance

- Quality of ultimate approximation limited by choice of $\Phi$

- Will my algorithm A compute weights $\tilde{r}$ that make good use of my basis functions $\Phi$?

- “Competitive” bound
  - If $\Phi r$ can come within $\epsilon$ of $J^*$, then algorithm A will compute $\tilde{r}$ such that
    1. $\Phi \tilde{r}$ is within $O(\epsilon)$ of $J^*$
    2. the greedy policy $u$ is $O(\epsilon)$–optimal
Notation

- $\|J\|_\infty = \max_x |J(x)|$
- weighted norms:

\[
\|J\|_{1,\nu} = \sum_x \nu(x)|J(x)|, \quad \|x\|_{\infty,\nu} = \max_x \nu(x)|J(x)|
\]
Graphical Interpretation of Approximate LP

Even with arbitrarily small $k$, we can have arbitrarily large $\varepsilon$, (or infeasibility!)

http://www.mit.edu/~pucci~p.18/29
Graphical Interpretation of Approximate LP

\[
J^* = \min J \text{ subject to } T J \geq J, \quad J = \Phi r
\]

Even with arbitrarily small \( k \), we can have arbitrarily large \( k J \) (or infeasibility!)

http://www.mit.edu/~pucci – p. 18/29
Graphical Interpretation of Approximate LP

Even with arbitrarily small $k$, we can have arbitrarily large $k$, or infeasibility!

$J = \Phi r$

$TJ \geq J$

$J(1)$

$J(2)$

$\Phi r^*$

$\Phi r$

$J^*$

http://www.mit.edu/~pucci – p. 18/29
Even with arbitrarily small $\| J^* - \Phi r^* \|_\infty$, we can have arbitrarily large $\| J^* - \Phi \tilde{r} \|$ (or infeasibility!)
Simple bound: If $\Phi v = e$ for some $v$,

$$\| J^* - \Phi \tilde{r} \|_{1,c} \leq \frac{2}{1 - \alpha} \| J^* - \Phi r^* \|_{\infty}$$
Error and performance bounds

- Simple bound: If $\Phi \nu = e$ for some $\nu$,

$$\| J^* - \Phi \tilde{\nu} \|_{1,c} \leq \frac{2}{1 - \alpha} \| J^* - \Phi r^* \|_{\infty}$$

- Limitations:
  - state-relevance weights?
  - maximum norm to assess architecture
Error and performance bounds

- Simple bound: If $\Phi v = e$ for some $v$,
  \[ ||J^* - \Phi \tilde{r}||_{1,c} \leq \frac{2}{1 - \alpha} ||J^* - \Phi r^*||_{\infty} \]

- Limitations:
  - state-relevance weights?
  - maximum norm to assess architecture
  - “Lyapunov function” $V > 0$:
    \[ \alpha \max_a E[V(y) | x, a] \leq \beta V(x) \]
Error and performance bounds

- Simple bound: If $\Phi v = e$ for some $v$,

$$\|J^* - \Phi \tilde{v}\|_{1,c} \leq \frac{2}{1 - \alpha} \|J^* - \Phi r^*\|_{\infty}$$

- Limitations:
  - state-relevance weights?
  - maximum norm to assess architecture
  - “Lyapunov function” $V > 0$:

$$\alpha \max_a E[V(y)|x, a] \leq \beta V(x)$$

- Theorem: If $\Phi v$ is a “Lyapunov function” for some $v$,

$$\|J^* - \Phi \tilde{v}\|_{1,c} \leq \frac{2c^T \Phi v}{1 - \beta} \|J^* - \Phi r^*\|_{\infty, 1/\Phi v}$$

Error Bound Insights

- Error proportional to best in architecture
Error Bound Insights

Error proportional to best in architecture

\[ \| J^* - \Phi r^* \|_{\infty, 1/V} = \max_x \frac{|J^*(x) - (\Phi r^*)(x)|}{V(x)} \]
Error Bound Insights

- Error proportional to best in architecture

\[ \| J^* - \Phi r^* \|_{\infty, 1/V} = \max_x \frac{|J^*(x) - (\Phi r^*)(x)|}{V(x)} \]

- \( V(x) \) large in rarely visited states \( \Rightarrow \) good scaling properties
Error Bound Insights

- Error proportional to best in architecture

\[ \| J^* - \Phi r^* \|_{\infty, 1/V} = \max_x \frac{|J^*(x) - (\Phi r^*)(x)|}{V(x)} \]

- \( V(x) \) large in rarely visited states ⇒ good scaling properties

- For multiclass queueing networks, error uniformly bounded on
  - size of the state space
  - dimension of the state space
Error Bound Insights

- Error proportional to best in architecture

\[ \| J^* - \Phi r^* \|_{\infty, 1/V} = \max_x \frac{| J^*(x) - (\Phi r^*)(x) |}{V(x)} \]

- \( V(x) \) large in rarely visited states \( \Rightarrow \) good scaling properties

- For multiclass queueing networks, error uniformly bounded on
  - size of the state space
  - dimension of the state space

- Performance bound:

\[ \| J_{\tilde{u}} - J^* \|_{1, \pi_{\tilde{u}}} \leq \frac{1}{1 - \alpha} \| J^* - \Phi \tilde{r} \|_{1, \pi_{\tilde{u}}} \]

- We have bound on \( \| J^* - \Phi \tilde{r} \|_{1, c} \)
Example: 8-dimensional queueing network

- Minimize total number of jobs in the system
Example: 8-dimensional queueing network

- Minimize total number of jobs in the system
- Linear and quadratic basis functions
- State-relevance weights with exponential decay
Example: 8-dimensional queueing network

- Minimize total number of jobs in the system
- Linear and quadratic basis functions
- State-relevance weights with exponential decay
- Average cost:

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALP</td>
<td>136.67</td>
</tr>
<tr>
<td>LBFS</td>
<td>153.82</td>
</tr>
<tr>
<td>FIFO</td>
<td>163.63</td>
</tr>
<tr>
<td>LONGEST</td>
<td>168.66</td>
</tr>
</tbody>
</table>
**Tetris**

- **Comparison against reported results**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average Score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>variation on TD (Bertsekas and Ioffe)</td>
<td>3500</td>
<td>many hours</td>
</tr>
<tr>
<td>variation on policy gradient (Kakade)</td>
<td>6000</td>
<td>days</td>
</tr>
<tr>
<td>ALP (Farias* and Van Roy)</td>
<td>5000</td>
<td>hours</td>
</tr>
</tbody>
</table>

* not me!

Remarks:
3 minutes to solve the approximate LP, rest of the time spent on simulation.

Solution is very sensitive to **c**.
## Tetris

- **Comparison against reported results**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average Score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>variation on TD (Bertsekas and Ioffe)</td>
<td>3500</td>
<td>many hours</td>
</tr>
<tr>
<td>variation on policy gradient (Kakade)</td>
<td>6000</td>
<td>days</td>
</tr>
<tr>
<td>ALP (Farias* and Van Roy)</td>
<td>5000</td>
<td>hours</td>
</tr>
</tbody>
</table>

* not me!

- **Remarks:**
  - 3 minutes to solve the approximate LP, rest of the time spent on simulation
  - solution is very sensitive to $c$
Outline

- Markov decision processes
- Approximate Dynamic Programming
- Approximate linear programming
- Performance and error analysis
- Constraint Sampling
Constraint Sampling in the Approximate LP

- one constraint per state-action pair
Constraint Sampling in the Approximate LP

- one constraint per state-action pair
- many constraints in low-dimensional space ⇒ redundancy
Constraint Sampling in the Approximate LP

- one constraint per state-action pair
- many constraints in low-dimensional space $\Rightarrow$ redundancy
- Problem-specific approaches in the literature:
  - Grötschel and Holland (1991)
  - Morrison and Kumar (1999)
  - Guestrin et al. (2002)
  - Schuurmans and Patrascu (2002)
Constraint Sampling in the Approximate LP

- one constraint per state-action pair
- many constraints in low-dimensional space \( \Rightarrow \) redundancy
- Problem-specific approaches in the literature:
  - Grötschel and Holland (1991)
  - Morrison and Kumar (1999)
  - Guestrin et al. (2002)
  - Schuurmans and Patrascu (2002)
- Generic approach? Complexity bounds?
The Reduced LP

$$\max_r \sum_x c(x)(\Phi r)(x)$$

s.t. $$g_a(x) + \alpha \sum_y P_a(x, y)(\Phi r)(y) \geq (\Phi r)(x), \forall x, \forall a$$
The Reduced LP

$$\max_r \sum_x c(x)(\Phi_r)(x)$$

s.t. $$g_a(x) + \alpha \sum_y P_a(x, y)(\Phi_r)(y) \geq (Phir)(x), \forall (x, a) \in \mathcal{N}$$

$$r \in \mathcal{B}$$

- $\mathcal{N}$ contains i.i.d. state-action pairs
The Reduced LP

\[
\max_r \sum_x c(x)(\Phi r)(x)
\]

s.t. \( g_a(x) + \alpha \sum_y P_a(x, y)(\Phi r)(y) \geq (Phir)(x), \forall (x, a) \in \mathcal{N} \)

\( r \in \mathcal{B} \)

- \( \mathcal{N} \) contains i.i.d. state-action pairs
- \( \mathcal{B} \) is a bounding box
The Reduced LP

\[
\max_r \sum_x c(x)(\Phi r)(x)
\]

s.t. \( g_a(x) + \alpha \sum_y P_a(x, y)(\Phi r)(y) \geq (Phir)(x), \forall (x, a) \in \mathcal{N} \)

\( r \in \mathcal{B} \)

- \( \mathcal{N} \) contains i.i.d. state-action pairs
- \( \mathcal{B} \) is a bounding box

- Theorem: With ideal sampling distribution, if

\[
|\mathcal{N}| = \text{poly} \left( p, |A|, \frac{1}{1 - \alpha}, \frac{1}{\epsilon}, \log \frac{1}{\delta}, \theta_{\mathcal{N}, V} \right)
\]

then with probability at least \( 1 - \delta \),

\[
\| J^* - \Phi \hat{r} \|_{1,c} \leq \| J^* - \Phi \tilde{r} \|_{1,c} + \epsilon \| J^* \|_{1,c}.
\]
Remarks on Constraint Sampling

- Sample complexity is
  - polynomial in number of basis functions
  - independent of dimensions of the state space
Remarks on Constraint Sampling

Sample complexity is

- polynomial in number of basis functions
- independent of dimensions of the state space
- linear on maximum number of actions per state $|A|$
Remarks on Constraint Sampling

- Sample complexity is
  - polynomial in number of basis functions
  - independent of dimensions of the state space
  - linear on maximum number of actions per state $|A|$
    but can do with $\log |A|$
Remarks on Constraint Sampling

- Sample complexity is
  - polynomial in number of basis functions
  - independent of dimensions of the state space
  - linear on maximum number of actions per state $|A|$
    but can do with $\log|A|$

- “ideal” distribution
- “Bounding set” $\mathcal{N}$
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \ i \in \mathcal{I}, \ r \in \mathbb{R}^p \]

- well-approximated with \( \text{poly}(p) \) constraints
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \ i \in \mathcal{I}, \ r \in \mathbb{R}^p \]

- well-approximated with \( \text{poly}(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \ i \in \mathcal{I}, \ r \in \mathbb{R}^p \]

- well-approximated with \( poly(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \ b_i]\| = 1 \)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \ i \in \mathcal{I}, \ r \in \mathbb{R}^p \]

- well-approximated with \( poly(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \ b_i]\| = 1 \)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \quad i \in \mathcal{I}, \quad r \in \mathbb{R}^p \]

- well-approximated with \( \text{poly}(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \ b_i]\| = 1 \)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \; i \in \mathcal{I}, \; r \in \mathbb{R}^p \]

- well-approximated with \( \text{poly}(p) \) constraints
- constraint characterized by vector \([A_i \; b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \; b_i]\| = 1 \)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \; i \in \mathcal{I}, \; r \in \mathbb{R}^p \]

- well-approximated with \( \text{poly}(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \| [A_i \ b_i] \| = 1 \)

http://www.mit.edu/~pucci – p. 27/29
**Intuition for constraint sampling**

\[ A_i r + b_i \geq 0, \ i \in I, \ r \in \mathbb{R}^p \]

- well-approximated with \( poly(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \ b_i]\| = 1 \)
Intuition for constraint sampling

\[ A_i r + b_i \geq 0, \ i \in \mathcal{I}, \ r \in \mathbb{R}^p \]

- well-approximated with \( poly(p) \) constraints
- constraint characterized by vector \([A_i \ b_i] \in \mathbb{R}^{p+1}\)
- for feasibility, assume w.l.g. \( \|[A_i \ b_i]\| = 1 \)
In Short...

- Approximate dynamic programming: central ideas and issues
In Short...

- Approximate dynamic programming: central ideas and issues
- Approximate linear programming: analysis, performance and error bounds
In Short...

- Approximate dynamic programming: central ideas and issues
- Approximate linear programming: analysis, performance and error bounds
  - first approximation error bounds for arbitrary basis functions and decisions
  - uniform bounds for multiclass queueing networks
In Short...

- Approximate dynamic programming: central ideas and issues
- Approximate linear programming: analysis, performance and error bounds
  - first approximation error bounds for arbitrary basis functions and decisions
  - uniform bounds for multiclass queueing networks
- Forthcoming:
  - analysis of case $\alpha \uparrow 1$
    - Lyapunov function argument breaks down
    - state-relevance weights $c$ disappear
  - relaxation of Lyapunov function argument
  - new variant of approximate LP
  - improved error bounds
Future Work

- Choice of state-relevance weights $c$
- Address norm discrepancy between error bound and performance bound
Future Work

- Choice of state-relevance weights $c$
  - Address norm discrepancy between error bound and performance bound
- Adaptive selection of basis functions
Future Work

- Choice of state-relevance weights $c$
  - Address norm discrepancy between error bound and performance bound
- Adaptive selection of basis functions
- Online versions of the algorithm
  - Robustness to model uncertainty
  - Incremental solution of the LP
  - Learning the Q function instead of the value function
Future Work

- Choice of state-relevance weights $c$
  - Address norm discrepancy between error bound and performance bound
- Adaptive selection of basis functions
- Online versions of the algorithm
  - Robustness to model uncertainty
  - Incremental solution of the LP
  - Learning the Q function instead of the value function
- Issues on constraint sampling
Future Work

- Choice of state-relevance weights $c$
  - Address norm discrepancy between error bound and performance bound
- Adaptive selection of basis functions
- Online versions of the algorithm
  - Robustness to model uncertainty
  - Incremental solution of the LP
  - Learning the Q function instead of the value function
- Issues on constraint sampling
- Specific applications: how far can we push guarantees?