Robust Optimization and Learning in Approximate Dynamic Programming Using Kernels

Theodore B. Trafalis

School of Industrial and Systems Engineering
University of Oklahoma

Talk
2nd Winter School on Data Analytics (WSDA 2017)
Nizhny Novgorod,Russia
November 03-04, 2017

Motivations and Objectives -Motivations-

Decision making is ubiquitous in science and engineering. Dynamic Programming (DP) provides a structured approach, but the CURSE OF DIMENSIONALITY makes finding an optimal policy:

- challenging and sometimes even impossible to calculate
- lacking computation efficiency
- requiring off-line calculations
- ignoring uncertainty within the problem parameters

Motivations and Objectives -Motivations-

Decision making is ubiquitous in science and engineering. Dynamic Programming (DP) provides a structured approach, but the CURSE OF DIMENSIONALITY makes finding an optimal policy:

- challenging and sometimes even impossible to calculate
- lacking computation efficiency
- requiring off-line calculations
- ignoring uncertainty within the problem parameters

Motivations and Objectives -Motivations-

Decision making is ubiquitous in science and engineering. Dynamic Programming (DP) provides a structured approach, but the CURSE OF DIMENSIONALITY makes finding an optimal policy:

- challenging and sometimes even impossible to calculate
- lacking computation efficiency
- requiring off-line calculations
- ignoring uncertainty within the problem parameters

Motivations and Objectives -Motivations-

Decision making is ubiquitous in science and engineering.
Dynamic Programming (DP) provides a structured approach, but the CURSE OF DIMENSIONALITY makes finding an optimal policy:

- challenging and sometimes even impossible to calculate
- lacking computation efficiency
- requiring off-line calculations
- ignoring uncertainty within the problem parameters

Since similar situations require similar actions we propose to use CLUSTERING (grouping). Therefore:

- find optimal and near optimal policies
- make computation more efficient
- search for policies on-line
- Incorporate robustness to get stable policies

Motivations and Objectives-Objectives-

Since similar situations require similar actions we propose to use CLUSTERING (grouping). Therefore:

- find optimal and near optimal policies
- make computation more efficient
- search for policies on-line
- Incorporate robustness to get stable policies

Motivations and Objectives-Objectives-

Since similar situations require similar actions we propose to use CLUSTERING (grouping). Therefore:

- find optimal and near optimal policies
- make computation more efficient
- search for policies on-line
- Incorporate robustness to get stable policies

Motivations and Objectives-Objectives-

Since similar situations require similar actions we propose to use CLUSTERING (grouping). Therefore:

- find optimal and near optimal policies
- make computation more efficient
- search for policies on-line
- Incorporate robustness to get stable policies

Outline

- The agent needs to take an action given the actual state and the reward from taking that action.
- The action taken does not need to increase the immediate reward but the expected cumulative reward.
- The rewards are used to evaluate how well the action taken will help achieve the already set goal.

Reinforcement Learning

- The agent needs to take an action given the actual state and the reward from taking that action.
- The action taken does not need to increase the immediate reward but the expected cumulative reward.
- The rewards are used to evaluate how well the action taken will help achieve the already set goal.

Reinforcement Learning

- The agent needs to take an action given the actual state and the reward from taking that action.
- The action taken does not need to increase the immediate reward but the expected cumulative reward.
- The rewards are used to evaluate how well the action taken will help achieve the already set goal.

Markov Decision Processes

An MDP is defined by:

- S, a state space.
- A, an action space.
- $\mathcal{P}$, a transition probability distribution, where: $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$.
- $g^a_{ss'}$ immediate reward/cost.
- π, the policy that the agent needs to find, is a mapping from S to A.

Markov Decision Processes

An MDP is defined by:

- S, a state space.
- A, an action space.
- $\mathcal{P}$, a transition probability distribution, where: $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$.
- $g^a_{ss'}$ immediate reward/cost.
- π, the policy that the agent needs to find, is a mapping from S to A.

Markov Decision Processes

An MDP is defined by:

- S, a state space.
- A, an action space.
- $\mathcal{P}$, a transition probability distribution, where: $\mathcal{P}^a_{ss'} = Pr\left\{s_{t+1} = s' | s_t = s, a_t = a\right\}$.
- $g^a_{ss'}$ immediate reward/cost.
- π, the policy that the agent needs to find, is a mapping from S to A.

Markov Decision Processes

An MDP is defined by:

- S, a state space.
- A, an action space.
- $\mathcal{P}$, a transition probability distribution, where: $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$.
- $g^a_{ss'}$ immediate reward/cost.
- π, the policy that the agent needs to find, is a mapping from S to A.

An MDP is defined by:

- S, a state space.
- A, an action space.
- $\mathcal{P}$, a transition probability distribution, where: $\mathcal{P}^a_{ss'} = Pr\left\{s_{t+1} = s' | s_t = s, a_t = a\right\}$.
- $g^a_{ss'}$ immediate reward/cost.
- π, the policy that the agent needs to find, is a mapping from S to A.

Value Functions

To evaluate how good a particular policy is, the agent needs to evaluate $\mathcal{J}_\pi(s)$ and $Q^\pi(s, a)$.

- The state-value function:

$$\mathcal{J}_\pi(s) = E_\pi \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} g_{s_k s_{k+1}}^{\pi(s_k)} | s_t = s \right\}$$

  Where $(0 < \gamma < 1)$.

- the action-value function:

$$Q_\pi(s, a) = E_\pi \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} g_{s_k s_{k+1}}^{\pi(s_k)} | s_t = s, a_t = a \right\}$$

  $\mathbb{E}_\pi \{\}$ is the expected value achieved following the policy π.

Value Functions

To evaluate how good a particular policy is, the agent needs to evaluate $\mathcal{J}_\pi(s)$ and $Q^\pi(s, a)$.

- The state-value function:

$$\mathcal{J}_\pi(s) = E_\pi \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} g_{s_k s_{k+1}}^{\pi(s_k)} | s_t = s \right\}$$

  Where $(0 < \gamma < 1)$.

- the action-value function:

$$Q_\pi(s, a) = E_\pi \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} g_{s_k s_{k+1}}^{\pi(s_k)} | s_t = s, a_t = a \right\}$$

  $\mathbb{E}_\pi \{\}$ is the expected value achieved following the policy π.

Value Functions -continued-

- The value functions $\mathcal{J}_\pi$ satisfies a recursive relationship:

$$\mathcal{J}_\pi(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s'} P^a_{ss'} \left[g^a_{ss'} + \gamma \mathcal{J}_\pi(s') \right]$$

- The optimal policy π^* is the one that maximizes $\mathcal{J}^\pi$

$$\mathcal{J}^*(s) = \max_\pi \mathcal{J}^\pi(s)$$

- The optimal action-value function Q^* is defined by:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

Value Functions -continued-

- The value functions $\mathcal{J}_\pi$ satisfies a recursive relationship:

$$\mathcal{J}_\pi(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s'} P^a_{ss'} \left[g^a_{ss'} + \gamma \mathcal{J}_\pi(s') \right]$$

- The optimal policy π^* is the one that maximizes $\mathcal{J}^\pi$

$$\mathcal{J}^*(s) = \max_\pi \mathcal{J}^\pi(s)$$

- The optimal action-value function Q^* is defined by:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

Value Functions -continued-

- The value functions $\mathcal{J}_\pi$ satisfies a recursive relationship:

$$\mathcal{J}_\pi(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s'} P^a_{ss'} \left[g^a_{ss'} + \gamma \mathcal{J}_\pi(s') \right]$$

- The optimal policy π^* is the one that maximizes $\mathcal{J}^\pi$

$$\mathcal{J}^*(s) = \max_\pi \mathcal{J}^\pi(s)$$

- The optimal action-value function Q^* is defined by:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

The value function $\mathcal{J}_\pi$ has the following properties:

- $\mathcal{J}_\pi$ is the unique solution to:

$$\mathcal{J}_\pi = \mathcal{T}_\pi \mathcal{J}_\pi$$

- The operator $\mathcal{T}_\pi$ is defined by:

$$\mathcal{T}_\pi \mathcal{J} = g_\pi + \gamma \mathcal{P}_\pi \mathcal{J}$$

- The optimal value function $\mathcal{J}^* = \min_\pi \mathcal{J}_\pi$ is the unique solution to Bellman's equation:

$$\mathcal{J}^* = \mathcal{T} \mathcal{J}^*$$

- where the operator $\mathcal{T}$ is:

$$\mathcal{T} \mathcal{J} = \min_\pi \mathcal{T}_\pi \mathcal{J}$$

The value function $\mathcal{J}_\pi$ has the following properties:

- $\mathcal{J}_\pi$ is the unique solution to:

$$\mathcal{J}_\pi = \mathcal{T}_\pi \mathcal{J}_\pi$$

- The operator $\mathcal{T}_\pi$ is defined by:

$$\mathcal{T}_\pi \mathcal{J} = g_\pi + \gamma \mathcal{P}_\pi \mathcal{J}$$

- The optimal value function $\mathcal{J}^* = \min_\pi \mathcal{J}_\pi$ is the unique solution to Bellman's equation:

$$\mathcal{J}^* = \mathcal{T} \mathcal{J}^*$$

- where the operator $\mathcal{T}$ is:

$$\mathcal{T} \mathcal{J} = \min_\pi \mathcal{T}_\pi \mathcal{J}$$

The value function $\mathcal{J}_\pi$ has the following properties:

- $\mathcal{J}_\pi$ is the unique solution to:

$$\mathcal{J}_\pi = \mathcal{T}_\pi \mathcal{J}_\pi$$

- The operator $\mathcal{T}_\pi$ is defined by:

$$\mathcal{T}_\pi \mathcal{J} = g_\pi + \gamma \mathcal{P}_\pi \mathcal{J}$$

- The optimal value function $\mathcal{J}^* = \min_\pi \mathcal{J}_\pi$ is the unique solution to Bellman's equation:

$$\mathcal{J}^* = \mathcal{T} \mathcal{J}^*$$

- where the operator $\mathcal{T}$ is:

$$\mathcal{T} \mathcal{J} = \min_\pi \mathcal{T}_\pi \mathcal{J}$$

The value function $\mathcal{J}_\pi$ has the following properties:

- $\mathcal{J}_\pi$ is the unique solution to:

$$\mathcal{J}_\pi = \mathcal{T}_\pi \mathcal{J}_\pi$$

- The operator $\mathcal{T}_\pi$ is defined by:

$$\mathcal{T}_\pi \mathcal{J} = g_\pi + \gamma \mathcal{P}_\pi \mathcal{J}$$

- The optimal value function $\mathcal{J}^* = \min_\pi \mathcal{J}_\pi$ is the unique solution to Bellman's equation:

$$\mathcal{J}^* = \mathcal{T}\mathcal{J}^*$$

- where the operator $\mathcal{T}$ is:

$$\mathcal{T}\mathcal{J} = \min_\pi \mathcal{T}_\pi \mathcal{J}$$

- This approach aims to decrease the complexity of the problem by reducing the number of states
- Many real life problems have states that are similar or closed to each other
- This similarity yields the same or similar value function
- $\Rightarrow$ calculate the value function just for one of the similar states
- To determine similar states we will use clustering

- This approach aims to decrease the complexity of the problem by reducing the number of states
- Many real life problems have states that are similar or closed to each other
- This similarity yields the same or similar value function
- $\Rightarrow$ calculate the value function just for one of the similar states
- To determine similar states we will use clustering

- This approach aims to decrease the complexity of the problem by reducing the number of states
- Many real life problems have states that are similar or closed to each other
- This similarity yields the same or similar value function
- $\Rightarrow$ calculate the value function just for one of the similar states
- To determine similar states we will use clustering

- This approach aims to decrease the complexity of the problem by reducing the number of states
- Many real life problems have states that are similar or closed to each other
- This similarity yields the same or similar value function
- $\Rightarrow$ calculate the value function just for one of the similar states
- To determine similar states we will use clustering

- This approach aims to decrease the complexity of the problem by reducing the number of states
- Many real life problems have states that are similar or closed to each other
- This similarity yields the same or similar value function
- $\Rightarrow$ calculate the value function just for one of the similar states
- To determine similar states we will use clustering

- We need to construct a subset $\overline{S}$ using clustering. The value function will be calculated for the elements in $\overline{S}$ only
- $\{S_1, S_2, \ldots, S_l\}$ be a partition of S
- $\overline{s}_i$, an element of S_i, be the **state representative** for the cluster S_i
- $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_l\}$

- We need to construct a subset $\overline{S}$ using clustering. The value function will be calculated for the elements in $\overline{S}$ only
- $\{S_1, S_2, \ldots, S_l\}$ be a partition of S
- $\overline{s}_i$, an element of S_i, be the **state representative** for the cluster S_i
- $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_l\}$

- We need to construct a subset $\overline{S}$ using clustering. The value function will be calculated for the elements in $\overline{S}$ only
- $\{S_1, S_2, \ldots, S_l\}$ be a partition of S
- $\overline{s}_i$, an element of S_i, be the **state representative** for the cluster S_i
- $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_l\}$

- We need to construct a subset $\overline{S}$ using clustering. The value function will be calculated for the elements in $\overline{S}$ only
- $\{S_1, S_2, \ldots, S_l\}$ be a partition of S
- $\overline{s}_i$, an element of S_i, be the **state representative** for the cluster S_i
- $\overline{S} = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_l\}$

Reduced States' Set

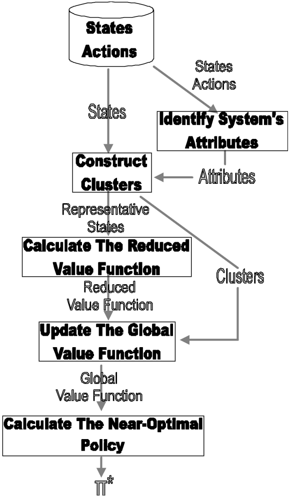


Figure: Reduced States' set diagram

- Consider an example of a city evacuation plan.
- There are few possible exits from the city.

- The Objective:
  Draft a policy that will guide the evacuees to get out of the city.

- Consider an example of a city evacuation plan.
- There are few possible exits from the city.
- The Objective:
  Draft a policy that will guide the evacuees to get out of the city.

- Consider an example of a city evacuation plan.
- There are few possible exits from the city.

- The Objective:
  Draft a policy that will guide the evacuees to get out of the city.

- To simplify the modeling, we suppose that there are 256 blocks (16 columns and 16 rows).
- At each intersection we can go on all 4 directions.
- The only available exits from the city are the north-west and south-east blocks.

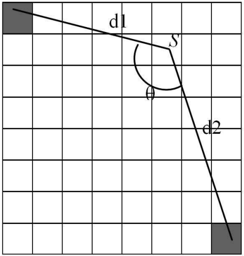


Figure: Simplified city representation with two exists

- To simplify the modeling, we suppose that there are 256 blocks (16 columns and 16 rows).
- At each intersection we can go on all 4 directions.
- The only available exits from the city are the north-west and south-east blocks.

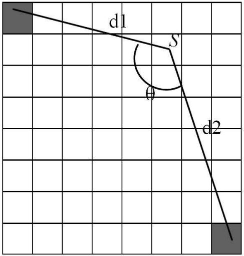


Figure: Simplified city representation with two exists

- To simplify the modeling, we suppose that there are 256 blocks (16 columns and 16 rows).
- At each intersection we can go on all 4 directions.
- The only available exits from the city are the north-west and south-east blocks.

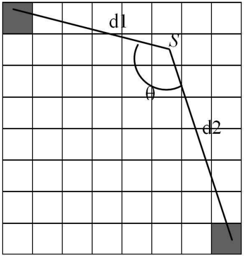


Figure: Simplified city representation with two exists

- 254 states and 2 terminal ones.
- The reward to go from one block to another is -1.
- The reward to get to one of the terminal states is 0.
- 3 attributes: d_1, d_2, and θ.

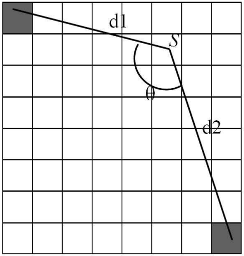


Figure: Attributes

- 254 states and 2 terminal ones.
- The reward to go from one block to another is -1.
- The reward to get to one of the terminal states is 0.
- 3 attributes: d_1, d_2, and θ.

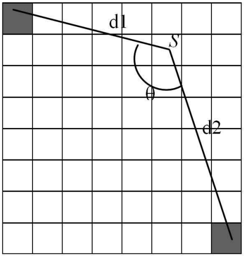


Figure: Attributes

- 254 states and 2 terminal ones.
- The reward to go from one block to another is -1.
- The reward to get to one of the terminal states is 0.
- 3 attributes: d_1, d_2, and θ.

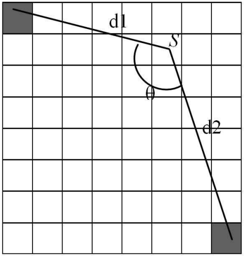


Figure: Attributes

- 254 states and 2 terminal ones.
- The reward to go from one block to another is -1.
- The reward to get to one of the terminal states is 0.
- 3 attributes: d_1, d_2, and θ.

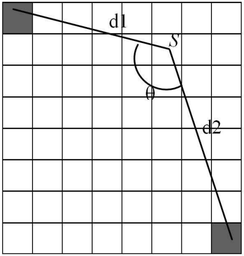


Figure: Attributes

- The accuracy rate is introduced to compare the different policies obtained:
$A_r = \frac{\text{number of accurate states}}{\text{number of states (256)}}$

	Optimal	Reduced	Reduced
# of clusters	256	73	56
Time (sec)	211	70	33
# of accurate states	256	256	196
A_r	1	1	.77

Table: Comparing Computation Time and Accuracy

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

Curse of dimensionality

- Consider finite sets with fixed cardinality for the states and actions.
- Need huge amount of resources to find an optimal policy.
- Need off-line algorithms.
- However
  In real world problems, the environment might change and include new states or actions.
- A policy that was optimal for a previous environment might not be optimal anymore.
- Use the reduced states' set method in an on-line framework.

On-line Reinforcement Learning using Kernels

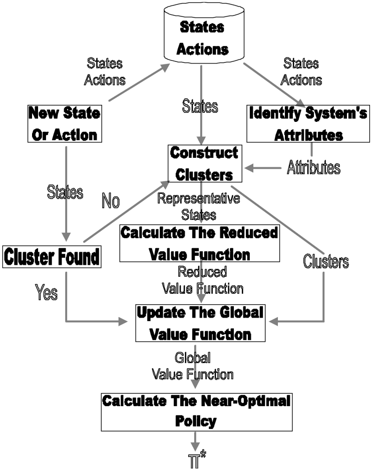


Figure: On-line Reduced States' set diagram

Consider the same city example with the following changes:

- Some blocks in the city might be inaccessible or some of the actions might become impossible to take.
- Due to the congestion, one of the emergency exits might be temporarily disabled.
- The emergency services might succeed in increasing the number of emergency exits.
- To make the problem feasible, we assume that during the whole horizon, there is at least one terminal state.
- The reward to get to an inaccessible block is $-\infty$.

Consider the same city example with the following changes:

- Some blocks in the city might be inaccessible or some of the actions might become impossible to take.
- Due to the congestion, one of the emergency exits might be temporarily disabled.
- The emergency services might succeed in increasing the number of emergency exits.
- To make the problem feasible, we assume that during the whole horizon, there is at least one terminal state.
- The reward to get to an inaccessible block is $-\infty$.

Consider the same city example with the following changes:

- Some blocks in the city might be inaccessible or some of the actions might become impossible to take.
- Due to the congestion, one of the emergency exits might be temporarily disabled.
- The emergency services might succeed in increasing the number of emergency exits.
- To make the problem feasible, we assume that during the whole horizon, there is at least one terminal state.
- The reward to get to an inaccessible block is $-\infty$.

Consider the same city example with the following changes:

- Some blocks in the city might be inaccessible or some of the actions might become impossible to take.
- Due to the congestion, one of the emergency exits might be temporarily disabled.
- The emergency services might succeed in increasing the number of emergency exits.
- To make the problem feasible, we assume that during the whole horizon, there is at least one terminal state.
- The reward to get to an inaccessible block is $-\infty$.

Consider the same city example with the following changes:

- Some blocks in the city might be inaccessible or some of the actions might become impossible to take.
- Due to the congestion, one of the emergency exits might be temporarily disabled.
- The emergency services might succeed in increasing the number of emergency exits.
- To make the problem feasible, we assume that during the whole horizon, there is at least one terminal state.
- The reward to get to an inaccessible block is $-\infty$.

Events	Optimal	Reduced			Reduced		
	time (sec)	time (sec)	clusters	A_r	time (sec)	clusters	A_r
- 1 block	220	75	74	1	36	57	.769
- 1 block	220	80	75	1	1	57	.762
- 1 exit	350	170	75	1	62	57	.773
- 1 block	345	1	75	1	1	57	.754
+ 1 block	347	1	75	1	1	57	.754
+ 1 exit	219	76	75	1	35	57	.773
+ 1 block	222	72	74	1	33	56	.773
+ 1 block	221	73	74	1	1	56	.766
Total	2144	548	-	-	170	-	-

Table: Comparing Computation Time and Accuracy for On line Reduced states' set

- We used a Modified Asynchronous Policy Iteration:
- The value iterations are executed according to the following updates:

$$\mathcal{J}_{k+1}(s) = \begin{cases} (\mathcal{T}_{\pi_k}\mathcal{J}_k)(s), \text{ if } s \in \overline{S}, \\ \mathcal{J}_{k+1}(s_i), \text{ s.t. } s \in S_i, \text{ otherwise,} \end{cases}$$

  the policy is kept unchanged by setting $\pi_{k+1} = \pi_k$
- The policy iterations are executed according to the following updates:

$$\pi_{k+1}(s) = \begin{cases} \text{argmin}_{a \in A(s)} \sum_{s'=0}^{|S|} \mathcal{P}^a_{ss'} \left(g^a_{ss'} + \mathcal{J}_k(s') \right), \text{ if } s \in \overline{S}, \\ \pi_{k+1}(s_i), \text{ s.t. } s \in S_i, \text{ otherwise,} \end{cases}$$

  the value function estimates are kept unchanged by setting $\mathcal{J}_{k+1} = \mathcal{J}_k$

- We used a Modified Asynchronous Policy Iteration:
- The value iterations are executed according to the following updates:

$$\mathcal{J}_{k+1}(s) = \begin{cases} (\mathcal{T}_{\pi_k}\mathcal{J}_k)(s), \text{ if } s \in \overline{S}, \\ \mathcal{J}_{k+1}(s_i), \text{ s.t. } s \in S_i, \text{ otherwise,} \end{cases}$$

  the policy is kept unchanged by setting $\pi_{k+1} = \pi_k$
- The policy iterations are executed according to the following updates:

$$\pi_{k+1}(s) = \begin{cases} \text{argmin}_{a \in A(s)} \sum_{s'=0}^{|S|} \mathcal{P}^a_{ss'} \left(g^a_{ss'} + \mathcal{J}_k(s') \right), \text{ if } s \in \overline{S}, \\ \pi_{k+1}(s_i), \text{ s.t. } s \in S_i, \text{ otherwise,} \end{cases}$$

  the value function estimates are kept unchanged by setting $\mathcal{J}_{k+1} = \mathcal{J}_k$

- We used a Modified Asynchronous Policy Iteration:
- The value iterations are executed according to the following updates:

$$\mathcal{J}_{k+1}(s) = \begin{cases} (\mathcal{T}_{\pi_k}\mathcal{J}_k)(s), & \text{if } s \in \overline{S}, \\ \mathcal{J}_{k+1}(s_i), \text{ s.t. } s \in S_i, & \text{otherwise,} \end{cases}$$

  the policy is kept unchanged by setting $\pi_{k+1} = \pi_k$
- The policy iterations are executed according to the following updates:

$$\pi_{k+1}(s) = \begin{cases} \text{argmin}_{a \in A(s)} \sum_{s'=0}^{|S|} \mathcal{P}^a_{ss'} \left(g^a_{ss'} + \mathcal{J}_k(s') \right), & \text{if } s \in \overline{S}, \\ \pi_{k+1}(s_i), \text{ s.t. } s \in S_i, & \text{otherwise,} \end{cases}$$

  the value function estimates are kept unchanged by setting $\mathcal{J}_{k+1} = \mathcal{J}_k$

- Let $s \in \overline{S}_i$ and $s \notin \overline{S}$, let $s' \in \overline{S}_j$.
- Let $\mathcal{P}_{ss'}^{\pi_k(s)} > 0$ and $s'' \in \overline{S}_j$ such that:

  1. $\mathcal{P}_{\bar{s}_i s''}^{\pi_k(\bar{s}_i)} > 0$ and $\left| \mathcal{P}_{ss'}^{\pi_k(s)} - \mathcal{P}_{\bar{s}_i s''}^{\pi_k(\bar{s}_i)} \right| \leq \epsilon_1$
  2. $\left| g_{ss'}^{\pi_k(s)} - g_{\bar{s}_i s''}^{\pi_k(\bar{s}_i)} \right| \leq \epsilon_2$

- then $\exists\ \epsilon \geq 0$ such that:
  $|(\mathcal{T}_{\pi_k} \mathcal{J}_k)(s) - \mathcal{J}_{k+1}(s)| \leq \epsilon\ \forall\ s \notin \overline{S}$.

- Let $s \in \overline{S}_i$ and $s \notin \overline{S}$, let $s' \in \overline{S}_j$.
- Let $\mathcal{P}_{ss'}^{\pi_k(s)} > 0$ and $s'' \in \overline{S}_j$ such that:
  1. $\mathcal{P}_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} > 0$ and $\left| \mathcal{P}_{ss'}^{\pi_k(s)} - \mathcal{P}_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} \right| \leq \epsilon_1$
  2. $\left| g_{ss'}^{\pi_k(s)} - g_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} \right| \leq \epsilon_2$
- then $\exists\ \epsilon \geq 0$ such that: $|(\mathcal{T}_{\pi_k} \mathcal{J}_k)(s) - \mathcal{J}_{k+1}(s)| \leq \epsilon\ \forall\ s \notin \overline{S}$.

- Let $s \in \overline{S}_i$ and $s \notin \overline{S}$, let $s' \in \overline{S}_j$.
- Let $\mathcal{P}_{ss'}^{\pi_k(s)} > 0$ and $s'' \in \overline{S}_j$ such that:
  1. $\mathcal{P}_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} > 0$ and $\left| \mathcal{P}_{ss'}^{\pi_k(s)} - \mathcal{P}_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} \right| \leq \epsilon_1$
  2. $\left| g_{ss'}^{\pi_k(s)} - g_{\overline{s}_i s''}^{\pi_k(\overline{s}_i)} \right| \leq \epsilon_2$
- then $\exists\ \epsilon \geq 0$ such that:
  $|(\mathcal{T}_{\pi_k}\mathcal{J}_k)(s) - \mathcal{J}_{k+1}(s)| \leq \epsilon\ \forall\ s \notin \overline{S}$.

- Conduct a set of experiments on the class of shortest path problems.
- Compare the time it takes to find an optimal policy using our algorithm and without reducing the states' set cardinality.
- Define the following two percentages:

$$A = \frac{Clustering\ Time + Reduced\ Time}{Optimal\ Time} \times 100,$$

$$B = \frac{Reduced\ Time}{Optimal\ Time} \times 100.$$

- Conduct a set of experiments on the class of shortest path problems.
- Compare the time it takes to find an optimal policy using our algorithm and without reducing the states' set cardinality.
- Define the following two percentages:

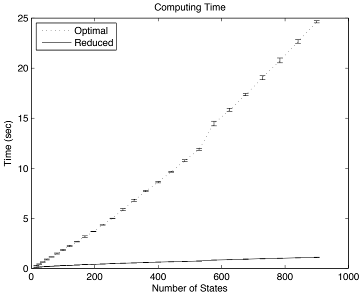
$$A = \frac{Clustering\ Time + Reduced\ Time}{Optimal\ Time} \times 100,$$

$$B = \frac{Reduced\ Time}{Optimal\ Time} \times 100.$$

- Conduct a set of experiments on the class of shortest path problems.
- Compare the time it takes to find an optimal policy using our algorithm and without reducing the states' set cardinality.
- Define the following two percentages:

$$A = \frac{Clustering\ Time + Reduced\ Time}{Optimal\ Time} \times 100,$$

$$B = \frac{Reduced\ Time}{Optimal\ Time} \times 100.$$



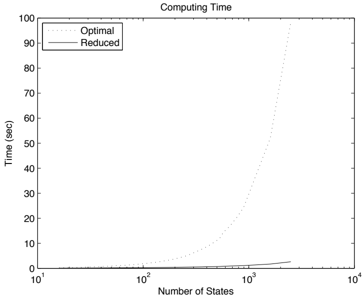


Figure: Optimal Vs Reduced Computation Times

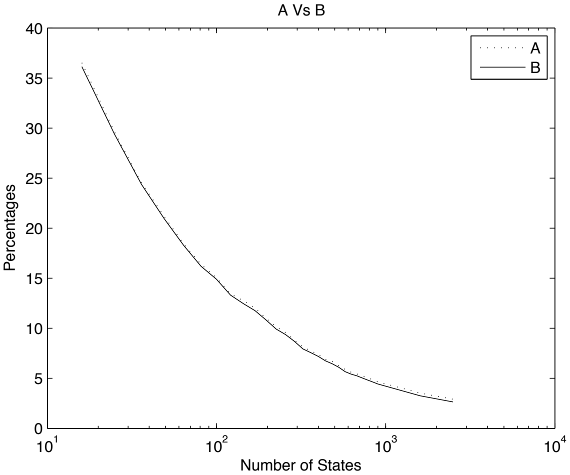


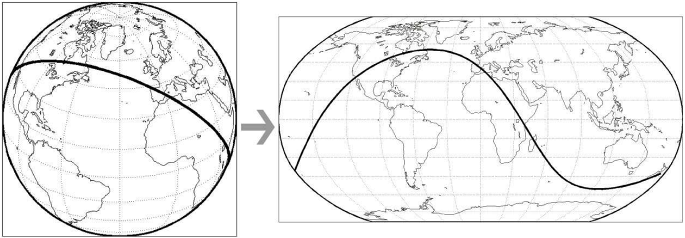
Figure: The Impact of the Clustering Time

- Kernel methods play a major role in Machine Learning.
- They provide a simple framework for manipulating nonlinear relationships.
- Instantaneous adaptation of former linear algorithms.
- Require modest computational resources.

- A kernel is a continuous symmetric real-valued function defined on compact subsets of $\mathbb{R}^n$, $k : (x, y) \mapsto k(x, y)$.
- A Mercer kernel is a nonnegative definite kernel.
- The domain of a Mercer kernel is called the input space.
- The quantity $k(x, y)$ can be used to represent measures of angle and measures of distance.
- Angles and distances are between inputs mapped in a higher dimensional Hilbert space.
- The Hilbert space is called the feature space.

- Mercer's theorem suggests a particular decomposition of Mercer kernels.
- A Mercer kernel can be expressed as a dot product between two inputs mapped in the feature space, $k(x, y) = \langle \phi(x) \cdot \phi(y) \rangle$.
- Explicit knowledge of the map ϕ and the feature space is not required. The only thing of importance is the kernel itself.
- With k we can calculate angles and distances between elements in the feature space without knowing the map ϕ.

- Kernel methods simplify the representation of nonlinear patterns in the input space.
- The intersection between hyperplanes and the manifold has a non-trivial reciprocal image in the input space.
- Instead of searching complex patterns in the input space, we use kernel methods.



Objectives

- Use Linear approximate dynamic programming to speed up the algorithm.
- Incorporate robustness to get stable policies.
- Use kernels to define the concept of the neighborhood of a state.
- Combine all the above techniques to generate an on-line robust RL algorithm.

RORLK: Algorithm Diagram

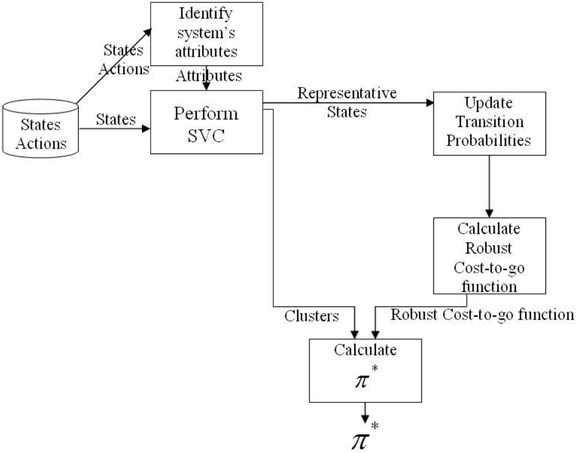


Figure: Robust On-line RL using Kernels diagram

RORLK: Algorithm Diagram

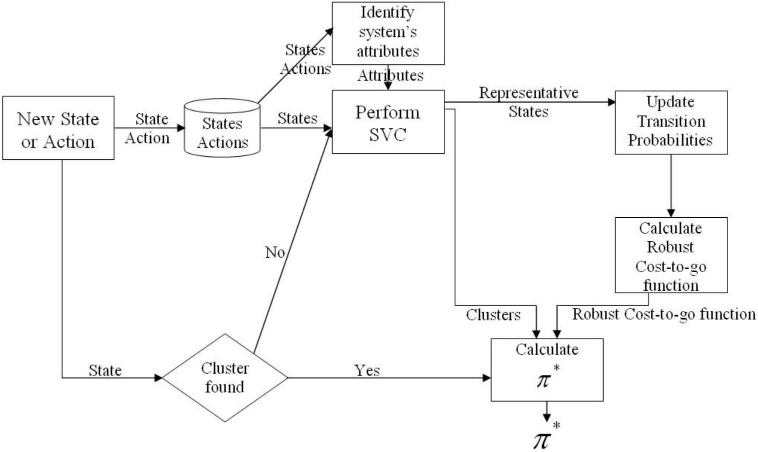
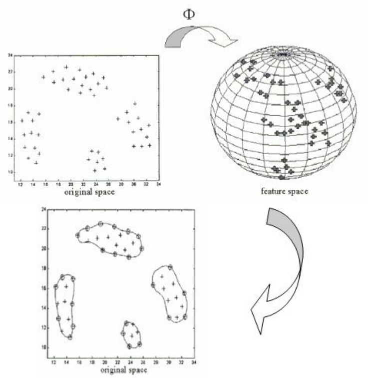


Figure: Robust On-line RL using Kernels diagram

- This approach aims to decrease the complexity of the problem by reducing the number of states.
- Many real life problems have states that are similar or closed to each other.
- This similarity yields the same or similar value function.
- $\Rightarrow$ calculate the value function just for one of the similar states.
- To determine similar states we will use clustering.

- Clustering is a part of data mining that consists of grouping a set of data according to various attributes.
- Many methods have been developed.
- The effectiveness of each approach depends on the nature of the data.
- Ben-Hur (2001) derived SVC from SVM.

- Map data points to a feature space.
- Search for the smallest sphere enclosing all data points.
- Map the sphere back and generate clusters.



- $\{X_i\}_{1 \leq i \leq n}$ a data set of n points in the input space.
- $X_i \in \mathbb{R}^d$ and d is the number of attributes.
- We need to find the smallest sphere with radius R.
- The optimization formulation is the following:

$$
\begin{array}{ll}
min_{R,a,\xi_i} & R^2 + C\sum_i \xi_i \\
\text{Subject to:} & \\
 & \|\Phi(X_i) - a\|^2 \leq R^2 + \xi_i, \quad i = 1 \cdots n \\
 & \xi_i \geq 0
\end{array}
$$

- To solve this optimization problem, we right the Lagrangian:

$$L(R, a, \mu, \beta, \xi) = R^2 - \sum_i \left(R^2 + \xi_i - \|\Phi(X_i) - a\|^2\right) \beta_i - \sum_i \xi_i \mu_i + C \sum_i \xi_i$$

  Where $\beta_i \geq 0$ and $\mu_i \geq 0$ are the Lagrange multipliers.

- Using the Karush-Kuhn-Tucker complementary slackness conditions, the Lagrangian becomes:

$$L(\beta) = \sum_i \Phi(X_i)^2 \beta_i - \sum_{ij} \beta_i \beta_j \Phi(X_i) \cdot \Phi(X_j)$$

  With the constraints $0 \leq \beta_i \leq C$.

- Replacing the dot product with the Gaussian kernel function:

  $K\left(X_i, X_j\right) = e^{-q\left\|X_i - X_j\right\|^2}$

  Where q is the width parameter.

- we get:

$$\begin{array}{ll} max_{\beta} & \sum_i K(X_i, X_i)\beta_i - \sum_{ij} \beta_i \beta_j K(X_i, X_j) \\ \text{Subject to:} & \\ & 0 \leq \beta_i \leq C, \quad i = 1 \cdots n \\ & \sum_i \beta_i = 1 \end{array}$$

- The distance of each image in the feature space from the sphere center *a*:

$$
\begin{aligned}
R^2(X) &= \|\Phi(X_i) - a\|^2 \\
&= K(X, X) - 2\sum_i \beta_i K(X_i, X) + \sum_{ij} \beta_i \beta_j K(X_i, X_j)
\end{aligned}
$$

- The radius of the sphere is as follows:

$$R_{Sphere} = \frac{\sum_{X_i \text{ is } SV} R(X_i)}{Number\ of\ SVs}$$

- clusters are defined as the connected components of the graph induced by A, where A is the adjacency matrix defined by:

$$A_{ij} = \begin{cases} 1, & \text{if for all } y \in (X_i, X_j), \; R(y) \leq R_{sphere} \\ 0, & \text{Otherwise} \end{cases}$$

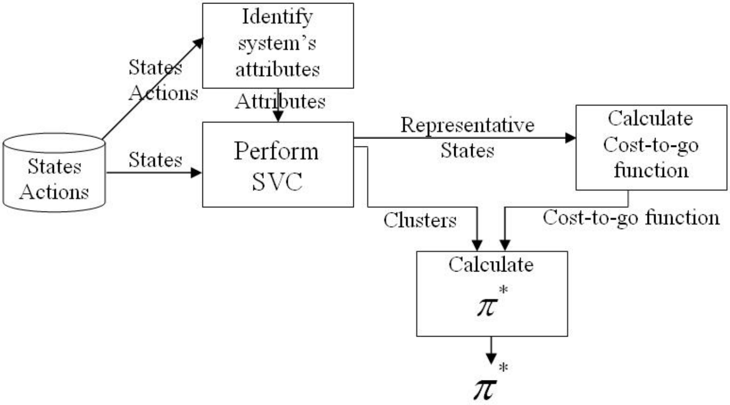


Figure: Reduced States' set diagram

Bellman's equation can be solved using the following Linear Dynamic Programming (LDP):

$$\max \quad c^T \mathcal{J}$$

$$\textit{subject to}: \; g_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \mathcal{J}(s') \geq \mathcal{J}(s), \; \forall s \in S, \forall a \in A(s)$$

Using the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ we can reduced the LDP to:

$$\max \sum_{i=1}^{l} \overline{c}_i \mathcal{J}(\overline{s}_i)$$

$$\textit{subject to}: \; g^a_{\overline{s}_j} + \gamma \sum_{i=1}^{l} |S_i| \, \mathcal{P}^a_{s\overline{s}_i} \mathcal{J}(\overline{s}_i) \geq \mathcal{J}(\overline{s}_j), \forall j = 1 \cdots l, \forall a \in A(\overline{s}_j)$$

where $\overline{c}_i = \sum_{s \in S_i} c_s$

$\mathcal{J}^*$ is approximated using a linear combination of preselected basis functions:

$$\phi_k : S \mapsto \mathbb{R},\ k = 1, \ldots, K$$

$\Rightarrow$ Generate a weight vector $\tilde{r} \in \mathbb{R}^K$, such that:

$$\mathcal{J}^*(s) \approx \sum_{k=0}^{K} \phi_k(s)\tilde{r}$$

Using the vector $\tilde{r}$ we get the Linear Approximate DP (LADP):
$\max_{r} C^T \Phi r$
subject to

$$g^a_{ss''} + \gamma \sum_{s' \in S} \mathcal{P}^a_{ss'} (\Phi r)(s') \geq (\Phi r)(s), \ \forall s \in S, \forall a \in A(s)$$

where $\Phi = \begin{bmatrix} \phi_1(1) & \cdots & \phi_K(1) \\ \vdots & & \vdots \\ \phi_1(|S|) & \cdots & \phi_K(|S|) \end{bmatrix}$

Using the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ we can reduced the LADP to:

max $\overline{c}^T \overline{\Phi} r$

$$\textit{subject to } g^a_{\overline{s}_j} + \gamma \sum_{i=1}^{l} |S_i| \, \mathcal{P}^a_{\overline{s}_j \overline{s}_i} \left(\overline{\Phi} r\right)(\overline{s}_i) \geq \left(\overline{\Phi} r\right)(\overline{s}_j), \forall j = 1 \cdots l, \forall a \in A(\overline{s}_j),$$

$$\text{where } \overline{\Phi} = \begin{bmatrix} \phi_1(\overline{s}_1) & \cdots & \phi_K(\overline{s}_1) \\ \vdots & & \vdots \\ \phi_1(\overline{s}_l) & \cdots & \phi_K(\overline{s}_l) \end{bmatrix}.$$

LADP Using Kernels

- The LADP yields good approximation to the value function
- How to choose the basis function and K
- We propose to use the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ and set:

$$\phi_k(s) = \kappa(s, \bar{s}_k), \; \forall \; k = 1 \ldots l$$

- The approximation will be done using the system's parameters
- $K = l$ is the number of clusters

LADP Using Kernels

- The LADP yields good approximation to the value function
- How to choose the basis function and K
- We propose to use the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ and set:

$$\phi_k(s) = \kappa(s, \bar{s}_k), \ \forall \ k = 1 \ldots l$$

- The approximation will be done using the system's parameters
- $K = l$ is the number of clusters

LADP Using Kernels

- The LADP yields good approximation to the value function
- How to choose the basis function and K
- We propose to use the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ and set:

$$\phi_k(s) = \kappa(s, \bar{s}_k), \ \forall \ k = 1 \ldots l$$

- The approximation will be done using the system's parameters
- $K = l$ is the number of clusters

LADP Using Kernels

- The LADP yields good approximation to the value function
- How to choose the basis function and K
- We propose to use the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ and set:

$$\phi_k(s) = \kappa(s, \overline{s}_k), \ \forall\ k = 1 \ldots l$$

- The approximation will be done using the system's parameters
- $K = l$ is the number of clusters

LADP Using Kernels

- The LADP yields good approximation to the value function
- How to choose the basis function and K
- We propose to use the partitioning $S = \{S_1, S_2, \ldots, S_l\}$ and set:

$$\phi_k(s) = \kappa(s, \bar{s}_k), \; \forall \, k = 1 \ldots l$$

- The approximation will be done using the system's parameters
- $K = l$ is the number of clusters

The LADP becomes Kernelized LADP:
max $c^T \mathcal{K} r$

$$\textit{subject to } g_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a (\mathcal{K} r)(s') \geq (\mathcal{K} r)(s),\ \forall s \in S, \forall a \in A(s),$$

where

$$\mathcal{K} = \begin{bmatrix} \kappa(s_1, \bar{s}_1) & \cdots & \kappa(s_1, \bar{s}_l) \\ \vdots & & \vdots \\ \kappa(s_{|S|}, \bar{s}_1) & \cdots & \kappa(s_{|S|}, \bar{s}_l) \end{bmatrix}$$

The RLADP becomes Kernelized RLADP:

max $\overline{c}^T \overline{\mathcal{K}} r$

$$\textit{subject to } g^a_{\overline{s}_j} + \gamma \sum_{i=1}^{l} |S_i| \, \mathcal{P}^a_{\overline{s}_j \overline{s}_i} \left(\overline{\mathcal{K}} r\right)(\overline{s}_i) \geq \left(\overline{\mathcal{K}} r\right)(\overline{s}_j), \forall j = 1 \cdots l, \forall a \in A(\overline{s}_j)$$

where

$$\overline{\mathcal{K}} = \begin{bmatrix} \kappa(\overline{s}_1, \overline{s}_1) & \cdots & \kappa(\overline{s}_1, \overline{s}_l) \\ \vdots & & \vdots \\ \kappa(\overline{s}_l, \overline{s}_1) & \cdots & \kappa(\overline{s}_l, \overline{s}_l) \end{bmatrix}$$

- We used the same shortest path problems used in section
- The reduce linear dynamic programming did not find the optimal policies all the time
- We introduce the reduction in this experiments:

$$R_r = \frac{|S| - |\overline{S}|}{|S|}$$

Experiments and Results

- We used the same shortest path problems used in section
- The reduce linear dynamic programming did not find the optimal policies all the time
- We introduce the reduction in this experiments:

$$R_r = \frac{|S| - |\overline{S}|}{|S|}$$

- We used the same shortest path problems used in section
- The reduce linear dynamic programming did not find the optimal policies all the time
- We introduce the reduction in this experiments:

$$R_r = \frac{|S| - |\overline{S}|}{|S|}$$

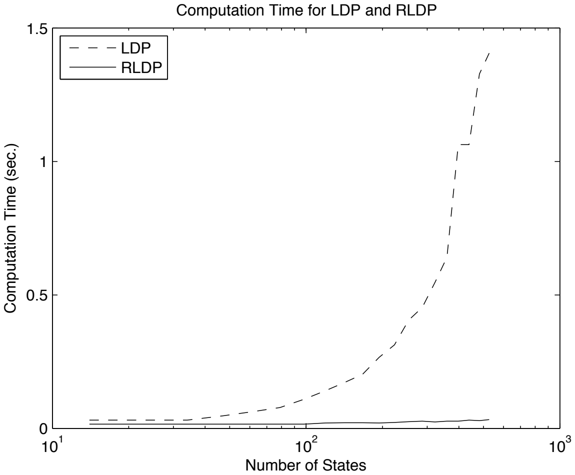


Figure: Computation Time -LDP Vs RLDP-

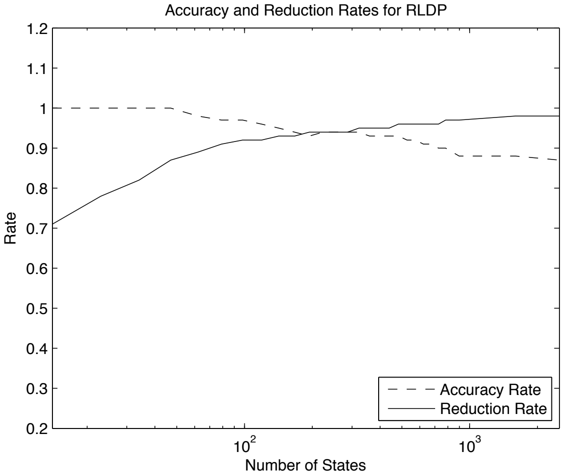


Figure: Accuracy and Reduction Rates -LDP Vs RLDP-

- Most developed algorithms ignore the uncertainty within the transition probability matrices.
- **However**, the DP optimal solution is sensitive to perturbation in transition matrices and in the cost/reward.
- **Therefore**, we need to account for the uncertainty.
- Different approaches have been developed.

- Most developed algorithms ignore the uncertainty within the transition probability matrices.
- **However**, the DP optimal solution is sensitive to perturbation in transition matrices and in the cost/reward.
- **Therefore**, we need to account for the uncertainty.
- Different approaches have been developed.

- Most developed algorithms ignore the uncertainty within the transition probability matrices.
- **However**, the DP optimal solution is sensitive to perturbation in transition matrices and in the cost/reward.
- **Therefore**, we need to account for the uncertainty.
- Different approaches have been developed.

- Most developed algorithms ignore the uncertainty within the transition probability matrices.
- **However**, the DP optimal solution is sensitive to perturbation in transition matrices and in the cost/reward.
- **Therefore**, we need to account for the uncertainty.
- Different approaches have been developed.

- Uncertainty within the transition probability matrices has been studied extensively
- But not the uncertainty within the reward/cost values
- We have augmented the LDP to make the policies robust with respect to the reward/cost values
- We have used Bertsimas and Sim's robust formulation to mitigate the effect of uncertainty within the transition probability matrices

- Uncertainty within the transition probability matrices has been studied extensively
- But not the uncertainty within the reward/cost values
- We have augmented the LDP to make the policies robust with respect to the reward/cost values
- We have used Bertsimas and Sim's robust formulation to mitigate the effect of uncertainty within the transition probability matrices

- Uncertainty within the transition probability matrices has been studied extensively
- But not the uncertainty within the reward/cost values
- We have augmented the LDP to make the policies robust with respect to the reward/cost values
- We have used Bertsimas and Sim's robust formulation to mitigate the effect of uncertainty within the transition probability matrices

- Uncertainty within the transition probability matrices has been studied extensively
- But not the uncertainty within the reward/cost values
- We have augmented the LDP to make the policies robust with respect to the reward/cost values
- We have used Bertsimas and Sim's robust formulation to mitigate the effect of uncertainty within the transition probability matrices

Consider the following linear optimization problem:

$$\max \; c^T x$$

$$\textit{subject to}: \; Ax \leq b$$

- Each element b_i of b can take a finite number of values $\{b_i^1, \ldots, b_i^q\}$
- In practice, a convex combination of the $\{b_i^1, \ldots, b_i^q\}$ is used as the expected value for b_i
- We assume that probability$(b_i = b_i^w) = \frac{1}{q}, \; \forall \; w = 1 \cdots q$

Consider the following linear optimization problem:

$$\max \; c^T x$$

$$\textit{subject to}: \; Ax \leq b$$

- Each element b_i of b can take a finite number of values $\{b_i^1, \ldots, b_i^q\}$
- In practice, a convex combination of the $\{b_i^1, \ldots, b_i^q\}$ is used as the expected value for b_i
- We assume that probability$(b_i = b_i^w) = \frac{1}{q}, \; \forall \; w = 1 \cdots q$

Consider the following linear optimization problem:

$$\max \; c^T x$$

$$\textit{subject to}: \; Ax \leq b$$

- Each element b_i of b can take a finite number of values $\{b_i^1, \ldots, b_i^q\}$
- In practice, a convex combination of the $\{b_i^1, \ldots, b_i^q\}$ is used as the expected value for b_i
- We assume that probability$(b_i = b_i^w) = \frac{1}{q}, \; \forall \; w = 1 \cdots q$

- For each realization b_i^w we associate the decision variable x_i^w.
- Hence, $E\left\{\max\ c^T x^w | Ax^w \leq b^w\right\}$ is equivalent to:

$$\max \sum_i \frac{c_i}{q} \sum_{v=1}^{q} x_i^v$$

$$\textit{subject to}: \sum_{ij, j\neq i} \frac{a_{ij}}{q} \sum_{v=1}^{q} x_j^v + a_{ii} x_i^w \leq b_i^w,\ \forall\ i, w$$

- The linear dynamic formulation becomes:

$$\max \sum_s \frac{c_s}{q} \sum_{v=1}^{q} \mathcal{J}(s^v)$$

$$\textit{subject to}: \ g_{s^w}^a + \gamma \sum_{s' \in S, s' \neq s^w} \frac{\mathcal{P}_{s^w s'}^a}{q} \sum_{v=1}^{q} \mathcal{J}(s'^v) \geq (1 - \gamma \mathcal{P}_{s^w s^w}^a) \mathcal{J}(s^w), \forall s^w, a, w$$

- For each realization b_i^w we associate the decision variable x_i^w.
- Hence, $E\left\{\max\ c^T x^w | Ax^w \le b^w\right\}$ is equivalent to:

$$\max \sum_i \frac{c_i}{q} \sum_{v=1}^{q} x_i^v$$

$$subject\ to: \sum_{ij,j\neq i} \frac{a_{ij}}{q} \sum_{v=1}^{q} x_j^v + a_{ii} x_i^w \le b_i^w,\ \forall\ i, w$$

- The linear dynamic formulation becomes:

$$\max \sum_s \frac{c_s}{q} \sum_{v=1}^{q} \mathcal{J}(s^v)$$

$$subject\ to: g_{s^w}^a + \gamma \sum_{s' \in S, s' \neq s^w} \frac{\mathcal{P}_{s^w s'}^a}{q} \sum_{v=1}^{q} \mathcal{J}(s'^v) \ge (1 - \gamma \mathcal{P}_{s^w s^w}^a) \mathcal{J}(s^w), \forall s^w, a, w$$

- For each realization b_i^w we associate the decision variable x_i^w.
- Hence, $E\left\{\max\ c^T x^w | Ax^w \leq b^w\right\}$ is equivalent to:

$$\max \sum_i \frac{c_i}{q} \sum_{v=1}^{q} x_i^v$$

$$subject\ to: \sum_{ij, j\neq i} \frac{a_{ij}}{q} \sum_{v=1}^{q} x_j^v + a_{ii} x_i^w \leq b_i^w,\ \forall\ i, w$$

- The linear dynamic formulation becomes:

$$\max \sum_s \frac{c_s}{q} \sum_{v=1}^{q} \mathcal{J}(s^v)$$

$$subject\ to: \ g_{s^w}^a + \gamma \sum_{s' \in S, s' \neq s^w} \frac{\mathcal{P}_{s^w s'}^a}{q} \sum_{v=1}^{q} \mathcal{J}(s'^v) \geq (1 - \gamma \mathcal{P}_{s^w s^w}^a) \mathcal{J}(s^w), \forall s^w, a, w$$

Robust Linear Dynamic Programming

- Use Bertsimas and Sim's formulation to account for uncertainty within the transition probability matrices
- We assume that each entry $\mathcal{P}^a_{ss'}$ is modeled as a symmetric and bounded random variable $\tilde{\mathcal{P}}^a_{ss'}$, such that:

$$\tilde{\mathcal{P}}^a_{ss'} \in \left[\mathcal{P}^a_{ss'} - \hat{\mathcal{P}}^a_{ss'}, \mathcal{P}^a_{ss'} + \hat{\mathcal{P}}^a_{ij}\right]$$

- We set $\alpha^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\delta_{s^w s'^{w'}} - \gamma \mathcal{P}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$ and $\hat{\alpha}^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\gamma \hat{\mathcal{P}}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$

- Use Bertsimas and Sim's formulation to account for uncertainty within the transition probability matrices
- We assume that each entry $\mathcal{P}^a_{ss'}$ is modeled as a symmetric and bounded random variable $\tilde{\mathcal{P}}^a_{ss'}$, such that:

$$\tilde{\mathcal{P}}^a_{ss'} \in \left[\mathcal{P}^a_{ss'} - \hat{\mathcal{P}}^a_{ss'}, \mathcal{P}^a_{ss'} + \hat{\mathcal{P}}^a_{ij}\right]$$

- We set $\alpha^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\delta_{s^w s'^{w'}} - \gamma \mathcal{P}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$ and $\hat{\alpha}^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\gamma \hat{\mathcal{P}}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$

- Use Bertsimas and Sim's formulation to account for uncertainty within the transition probability matrices
- We assume that each entry $\mathcal{P}^a_{ss'}$ is modeled as a symmetric and bounded random variable $\tilde{\mathcal{P}}^a_{ss'}$, such that:

$$\tilde{\mathcal{P}}^a_{ss'} \in \left[\mathcal{P}^a_{ss'} - \hat{\mathcal{P}}^a_{ss'}, \mathcal{P}^a_{ss'} + \hat{\mathcal{P}}^a_{ij}\right]$$

- We set $\alpha^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\delta_{s^w s'^{w'}} - \gamma \mathcal{P}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$ and $\hat{\alpha}^a_{s^w s'^{w'}} = (1 - \delta_{s^w s'^{w'}}) \frac{\gamma \hat{\mathcal{P}}^a_{s^w s'^{w'}}}{q + (1-q)\delta_{s^w s'^{w'}}}$

The Robust Linear Dynamic Programming is the following:

$$
\begin{aligned}
\max \quad & c_\omega^T \mathcal{J}_\omega && (1)\\
\textit{subject to}: & \sum_{j=1}^{|S|}\sum_{v=1}^{q} \alpha^a_{s^w s^v_j}\mathcal{J}(s^v_j) + z^{aw}_s \Gamma^{aw}_s + \sum_{j=1}^{|S|}\sum_{v=1}^{q} p_{s^w s^v_j} \le g^a_{s^w}, \ \forall\ s^w, a && (2)\\
& z^{aw}_s + p_{s^w s^v_j} \ge \hat{\alpha}^a_{s^w s^v_j} y^v_j, \ \forall\ s^w, a, j, v && (3)\\
& -y^v_j \le \mathcal{J}(s^v_j) \le y^v_j, \ \forall\ j, v && (4)\\
& p_{s^w s^v_j} \ge 0, \ \forall\ s^w, j, v && (5)\\
& z \ge 0 && (6)\\
& y \ge 0. && (7)
\end{aligned}
$$

The Reduced Robust Linear Dynamic Programming is the following:

$$\max \quad \sum_{i=1}^{I} \frac{\overline{c}_i}{q} \sum_{v=1}^{q} \mathcal{J}(\overline{s}_i) \tag{8}$$

$$\textit{subject to}: \sum_{j=1}^{I} \sum_{v=1}^{q} |\overline{S}_j| \alpha^{a}_{\overline{s}_i^w \overline{s}_j^v} \mathcal{J}(\overline{s}_j^v) + z^{aw}_{\overline{s}_i} \Gamma^{aw}_{\overline{s}_i} + \sum_{j=1}^{I} \sum_{v=1}^{q} p_{s^w s_j^v} \leq g^{a}_{\overline{s}_i^w}, \ \forall\ \overline{s}_i^w, a \tag{9}$$

$$z^{aw}_{\overline{s}_i} + p_{\overline{s}_i^w \overline{s}_j^v} \geq \hat{\alpha}^{a}_{\overline{s}_i^w \overline{s}_j^v} y_j^v, \ \forall\ \overline{s}_i^w, a, j, v \tag{10}$$

$$-y_j^v \leq \mathcal{J}(\overline{s}_j^v) \leq y_j^v, \ \forall\ j, v \tag{11}$$

$$p_{\overline{s}_i^w \overline{s}_j^v} \geq 0, \ \forall\ \overline{s}_i^w, j, v \tag{12}$$

$$z \geq 0 \tag{13}$$

$$y \geq 0. \tag{14}$$

- An option is the right to engage in a future transaction on some underlying security for a certain prescribed price known as the exercise or strike price
- There are two basic types of options: American and European options
- There are two kinds of American options: call options and put options

- An option is the right to engage in a future transaction on some underlying security for a certain prescribed price known as the exercise or strike price
- There are two basic types of options: American and European options
- There are two kinds of American options: call options and put options

- An option is the right to engage in a future transaction on some underlying security for a certain prescribed price known as the exercise or strike price
- There are two basic types of options: American and European options
- There are two kinds of American options: call options and put options

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- K is the strike price
- T is the expiration date
- x_t is the price of the underlying asset at t
- r is the risk free interest rate
- σ is the volatility
- The intrinsic value of a put option for the holder is:

$$f(x_t) = \begin{cases} K - x_t, & \text{if } x_t \leq K \text{ and } t \leq T \\ 0, & \text{otherwise} \end{cases}$$

- American options can be exercised at any time before maturity
- It is important to set exercising strategies that maximize the profit
- American options can be implemented as optimal stopping problems (OSPs)
- OSPs can be written in the form of a Bellman equation (DP)

- American options can be exercised at any time before maturity
- It is important to set exercising strategies that maximize the profit
- American options can be implemented as optimal stopping problems (OSPs)
- OSPs can be written in the form of a Bellman equation (DP)

- American options can be exercised at any time before maturity
- It is important to set exercising strategies that maximize the profit
- American options can be implemented as optimal stopping problems (OSPs)
- OSPs can be written in the form of a Bellman equation (DP)

Example: Strategy for American Options

- American options can be exercised at any time before maturity
- It is important to set exercising strategies that maximize the profit
- American options can be implemented as optimal stopping problems (OSPs)
- OSPs can be written in the form of a Bellman equation (DP)

- The value function is the expected value of the intrinsic value under the risk neutral assumption:

$$\mathcal{J}(x_t) = \max_{t \in [0,T]} E_{\pi} \left\{ e^{-rt} f(x_t) \right\}$$

- The evolution of x_t is simulated using the Binomial Options Pricing Model (BOPM)

$$x_{t+1} = \begin{cases} ux_t, & \text{with probability } p \\ dx_t, & \text{with probability } 1-p \end{cases}.$$

- The value function is the expected value of the intrinsic value under the risk neutral assumption:

$$\mathcal{J}(x_t) = \max_{t \in [0,T]} E_\pi \left\{ e^{-rt} f(x_t) \right\}$$

- The evolution of x_t is simulated using the Binomial Options Pricing Model (BOPM)

$$x_{t+1} = \begin{cases} ux_t, & \text{with probability } p \\ dx_t, & \text{with probability } 1-p \end{cases} .$$

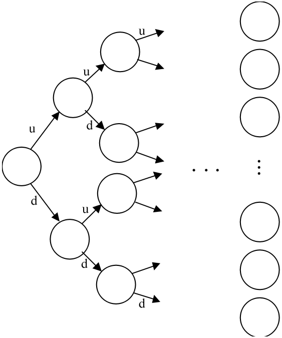


Figure: The Binomial Options Pricing Model

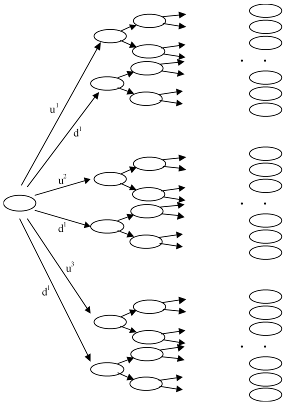


Figure: The Robust Binomial Options Pricing Model

The parameters used are:

	$K = 1$	$x_0 = .5$	$e^{-r} = .99$	
	Low Volatility	Moderate Volatility	High Volatility	Average
u	$\frac{13}{12}$	$\frac{8}{6}$	$\frac{9}{5}$	1.406
d	$\frac{12}{13}$	$\frac{6}{8}$	$\frac{5}{9}$	.711

Table: Parameters Used for the Exercising Strategy

The intrinsic value for this example is:

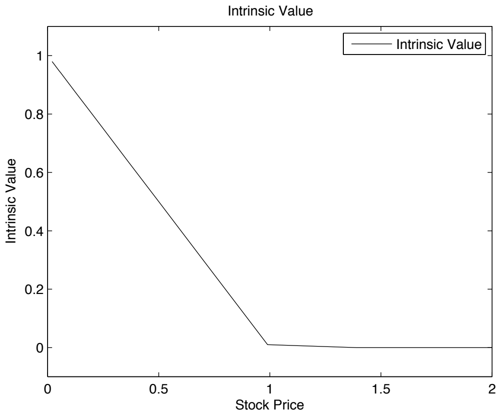


Figure: The Intrinsic Value for the Put Option

Time	Low Volatility	Moderate Volatility	High Volatility	Average
1	0.393	0.281	0.154	0.253
2	0.393	0.281	0.154	0.253
3	0.393	0.281	.278	0.253
4	0.393	0.281	.278	0.356
5	0.426	0.375	.278	0.356
6	0.426	0.375	.278	0.356
7	0.461	0.375	.278	0.356
8	0.5	0.5	0.5	0.5
9	0.7	0.7	0.7	0.7
10	1	1	1	1

Table: Exercising Strategies -Robust Vs Average-

The intrinsic value for this example is:

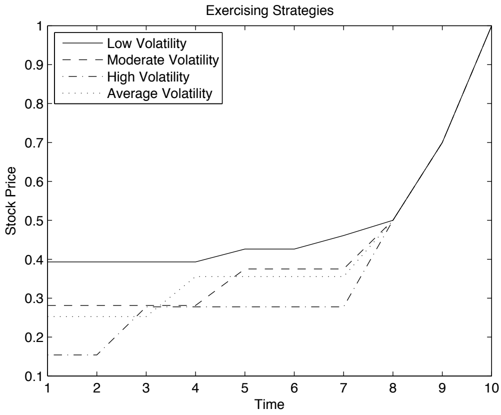


Figure: Exercising Strategies for the Robust Formulation and the Average

- This work was motivated by the “curse of dimensionality” in reinforcement learning and dynamic programming
- Coped with this problem using clustering
- Conducted a mathematical analysis to support the results obtained
- Developed an on line dynamic programming procedure
- Presented a novel LP algorithm to mitigate uncertainties within the transition probability matrices and the cost/reward

- This work was motivated by the “curse of dimensionality” in reinforcement learning and dynamic programming
- Coped with this problem using clustering
- Conducted a mathematical analysis to support the results obtained
- Developed an on line dynamic programming procedure
- Presented a novel LP algorithm to mitigate uncertainties within the transition probability matrices and the cost/reward

- This work was motivated by the “curse of dimensionality” in reinforcement learning and dynamic programming
- Coped with this problem using clustering
- Conducted a mathematical analysis to support the results obtained
- Developed an on line dynamic programming procedure
- Presented a novel LP algorithm to mitigate uncertainties within the transition probability matrices and the cost/reward

- This work was motivated by the “curse of dimensionality” in reinforcement learning and dynamic programming
- Coped with this problem using clustering
- Conducted a mathematical analysis to support the results obtained
- Developed an on line dynamic programming procedure
- Presented a novel LP algorithm to mitigate uncertainties within the transition probability matrices and the cost/reward

- This work was motivated by the “curse of dimensionality” in reinforcement learning and dynamic programming
- Coped with this problem using clustering
- Conducted a mathematical analysis to support the results obtained
- Developed an on line dynamic programming procedure
- Presented a novel LP algorithm to mitigate uncertainties within the transition probability matrices and the cost/reward

- Develop a linear dynamic programming that uses multicriteria optimization (state relevant weight)
- Find a good trade off between the reduction and the accuracy rates
- Develop the concept presented in this talk for an infinite horizon and for continuous states' space

- Develop a linear dynamic programming that uses multicriteria optimization (state relevant weight)
- Find a good trade off between the reduction and the accuracy rates
- Develop the concept presented in this talk for an infinite horizon and for continuous states' space

- Develop a linear dynamic programming that uses multicriteria optimization (state relevant weight)
- Find a good trade off between the reduction and the accuracy rates
- Develop the concept presented in this talk for an infinite horizon and for continuous states' space

Questions

Thank you