

Knowledge Transfer in Markov Decision Processes

Caitlin Phillips

`cphill@cs.mcgill.ca`

Supervised by Prakash Panangaden, Joelle Pineau, and Doina Precup

McGill University

What is a Markov Decision Process (MDP)?

A way to formulate problems in Machine Learning, which captures the stochastic nature of real-life situations.

Mathematically:

What is a Markov Decision Process (MDP)?

A way to formulate problems in Machine Learning, which captures the stochastic nature of real-life situations.

Mathematically:

- A set of states $s_i \in S$

What is a Markov Decision Process (MDP)?

A way to formulate problems in Machine Learning, which captures the stochastic nature of real-life situations.

Mathematically:

- A set of states $s_i \in S$
- A set of actions $a_j \in A$

What is a Markov Decision Process (MDP)?

A way to formulate problems in Machine Learning, which captures the stochastic nature of real-life situations.

Mathematically:

- A set of states $s_i \in S$
- A set of actions $a_j \in A$
- A probability function $P : S \times A \times S \mapsto [0, 1]$ denoted by $P_{ss'}^a$

What is a Markov Decision Process (MDP)?

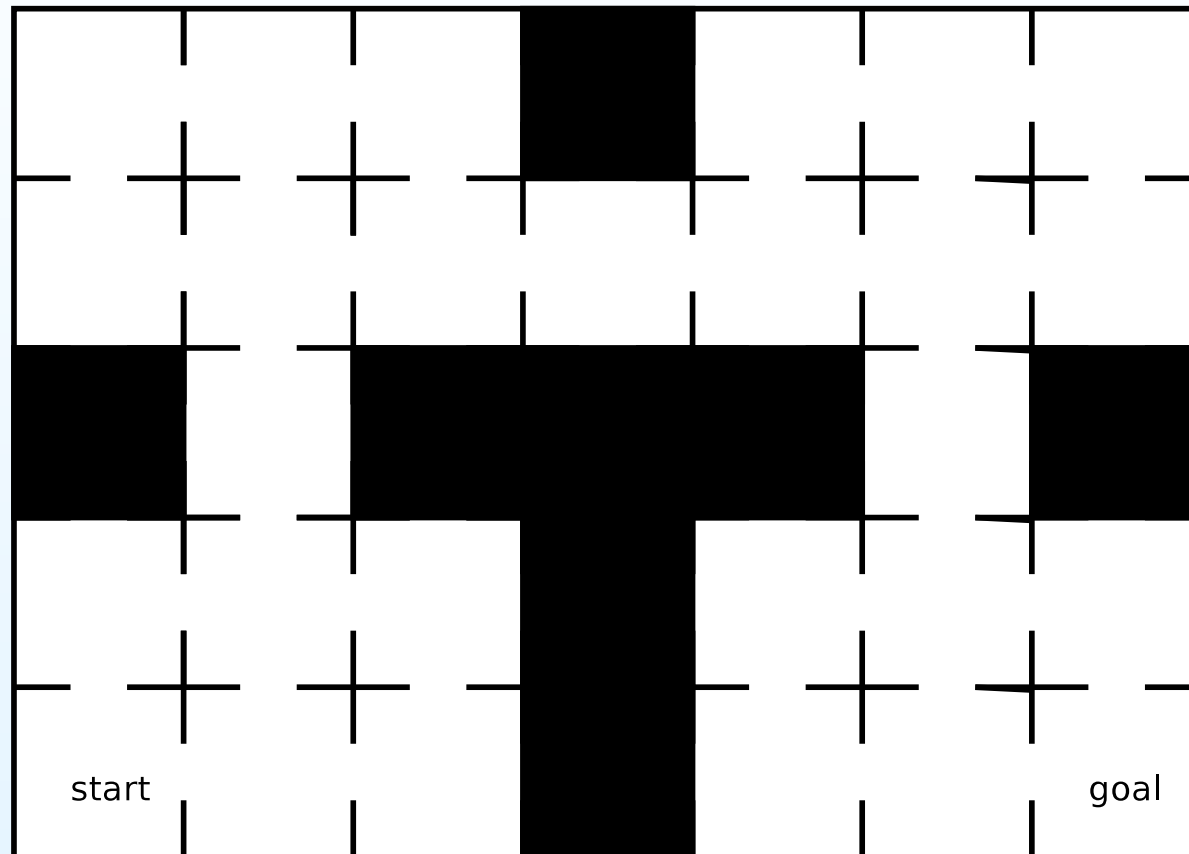
A way to formulate problems in Machine Learning, which captures the stochastic nature of real-life situations.

Mathematically:

- A set of states $s_i \in S$
- A set of actions $a_j \in A$
- A probability function $P : S \times A \times S \mapsto [0, 1]$ denoted by $P_{ss'}^a$
- A reward function $R : S \times A \mapsto \mathbb{R}$ denoted by R_s^a

A Simple Example

The goal of an MDP is to maximize the reward over time.



Solving MDPs

- The optimal value function V^* of an MDP:

$$V^*(s) = \max_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

where $\gamma \in (0, 1)$ is a discount factor.

Solving MDPs

- The optimal value function V^* of an MDP:

$$V^*(s) = \max_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

where $\gamma \in (0, 1)$ is a discount factor.

- Policy induced by the value function:

$$\pi^*(s) = \operatorname{argmax}_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

Solving MDPs

- The optimal value function V^* of an MDP:

$$V^*(s) = \max_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

where $\gamma \in (0, 1)$ is a discount factor.

- Policy induced by the value function:

$$\pi^*(s) = \operatorname{argmax}_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

- Problem: On larger MDPs, it can take a long time to find V^* .

Solving MDPs

- The optimal value function V^* of an MDP:

$$V^*(s) = \max_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

where $\gamma \in (0, 1)$ is a discount factor.

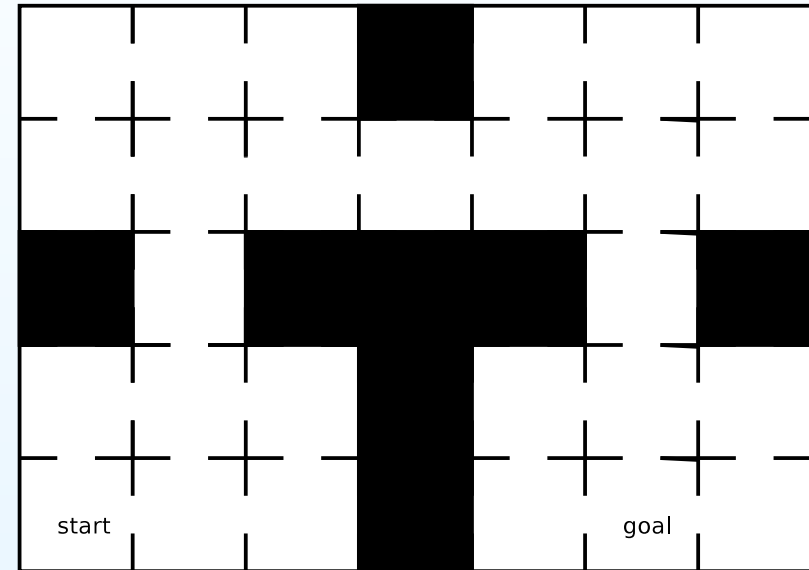
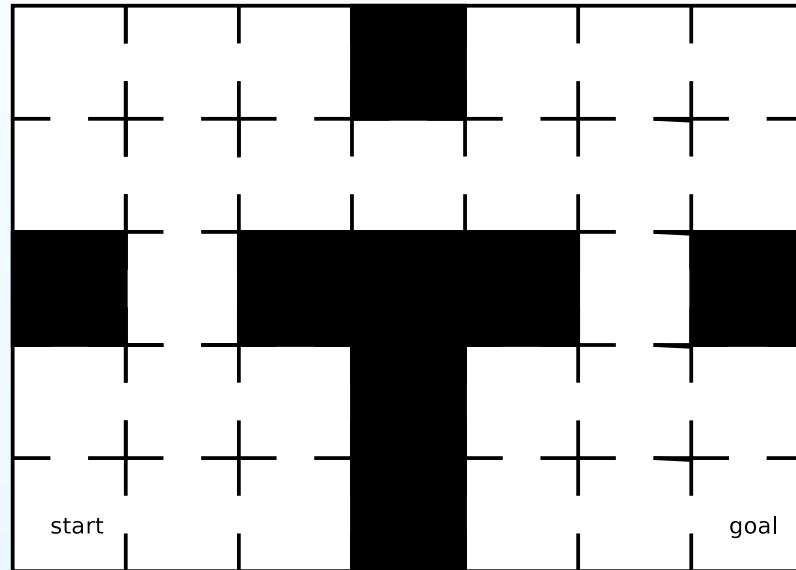
- Policy induced by the value function:

$$\pi^*(s) = \operatorname{argmax}_{a \in A} (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s'))$$

- Problem: On larger MDPs, it can take a long time to find V^* .
- Suppose we have already solved a similar problem, can we use that knowledge to help solve this one?

Motivation

How well will a policy learned on the first MDP work on the second one?:



Policy Transfer

-

$$M_1 = (S_1, A, \{P_{ss'}^a\}, \{R_s^a\})$$

$$M_2 = (S_2, A, \{Q_{tt'}^a\}, \{R_t^a\})$$

2 MDPs which share the same set of actions.

Policy Transfer

-

$$M_1 = (S_1, A, \{P_{ss'}^a\}, \{R_s^a\})$$

$$M_2 = (S_2, A, \{Q_{tt'}^a\}, \{R_t^a\})$$

2 MDPs which share the same set of actions.

- Suppose there is a mapping $\rho : S_1 \mapsto S_2$ which maps each state in M_1 to a corresponding one in M_2 .

Policy Transfer

-

$$M_1 = (S_1, A, \{P_{ss'}^a\}, \{R_s^a\})$$

$$M_2 = (S_2, A, \{Q_{tt'}^a\}, \{R_t^a\})$$

2 MDPs which share the same set of actions.

- Suppose there is a mapping $\rho : S_1 \mapsto S_2$ which maps each state in M_1 to a corresponding one in M_2 .
- Then a policy π_1 on M_1 naturally induces a policy on M_2 as follows:

$$\pi_1(s, a) = \pi_2(\rho(s), a)$$

Policy Transfer

-

$$M_1 = (S_1, A, \{P_{ss'}^a\}, \{R_s^a\})$$

$$M_2 = (S_2, A, \{Q_{tt'}^a\}, \{R_t^a\})$$

2 MDPs which share the same set of actions.

- Suppose there is a mapping $\rho : S_1 \mapsto S_2$ which maps each state in M_1 to a corresponding one in M_2 .
- Then a policy π_1 on M_1 naturally induces a policy on M_2 as follows:

$$\pi_1(s, a) = \pi_2(\rho(s), a)$$

- How do we choose M_1 and M_2 ?

Policy Transfer

-

$$M_1 = (S_1, A, \{P_{ss'}^a\}, \{R_s^a\})$$

$$M_2 = (S_2, A, \{Q_{tt'}^a\}, \{R_t^a\})$$

2 MDPs which share the same set of actions.

- Suppose there is a mapping $\rho : S_1 \mapsto S_2$ which maps each state in M_1 to a corresponding one in M_2 .
- Then a policy π_1 on M_1 naturally induces a policy on M_2 as follows:

$$\pi_1(s, a) = \pi_2(\rho(s), a)$$

- How do we choose M_1 and M_2 ?
- And once we do, how close to optimal will π_2 be?

Finding a Similar MDP

To measure distance between MDPs, look at reward and probability distribution in corresponding states.

Bisimulation Metric:

$$d(s, t) = \max_{a \in A} (|R_s^a - R_t^a| + \gamma T_K(d)(P_s^a, Q_t^a))$$

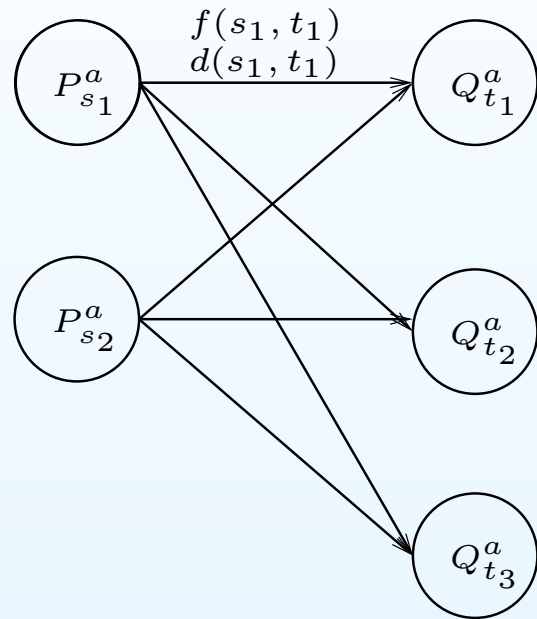
where $T_K(d)$ is a probability metric known as the Kantorovich.

Translation: How different are the rewards, and what are the chances of transitioning to “close” states from these ones?

Ferns, N. et al., Metrics For Finite Markov Decision Processes, 2004

The Kantorovich Metric

Calculating the Kantorovich distance amounts to solving a minimum cost flow (a linear program):



$d(s_i, t_j)$: Cost of shipping one unit from s_i to t_j

$f(s_i, t_j)$: $d(s_i, t_j) \times$ [no. units shipped along $\overrightarrow{s_i t_j}$]

Supply Nodes

Demand Nodes

Goal: Find the minimum cost of shipping the contents of the Supply nodes to the Demand nodes, according to their needs.

Very expensive to solve!!

Speeding Up the Calculation

To speed up the calculation of the Kantorovich, we approximated the probability distributions by empirical distributions.

Ferns, N. et al., Methods for Computing State Similarity in Markov Decision Processes,
2006

Speeding Up the Calculation

To speed up the calculation of the Kantorovich, we approximated the probability distributions by empirical distributions.

- Given P_s^a , take n random samples (X_1, X_2, \dots, X_n) for state s under action a .

Speeding Up the Calculation

To speed up the calculation of the Kantorovich, we approximated the probability distributions by empirical distributions.

- Given P_s^a , take n random samples (X_1, X_2, \dots, X_n) for state s under action a .
- Similarly, for Q_t^a take n random samples (Y_1, Y_2, \dots, Y_n) for state t under action a .

Speeding Up the Calculation

To speed up the calculation of the Kantorovich, we approximated the probability distributions by empirical distributions.

- Given P_s^a , take n random samples (X_1, X_2, \dots, X_n) for state s under action a .
- Similarly, for Q_t^a take n random samples (Y_1, Y_2, \dots, Y_n) for state t under action a .
- The Kantorovich distance is then efficiently approximated by

$$T_K(d)(P_s^a, Q_t^a) = \min_{\sigma} \frac{1}{n} \sum_{k=1}^n d(X_k, Y_{\sigma(k)}) \text{ where } \sigma \text{ is some}$$

permutation of the indices. (This is still a linear program, but it is an easier one!)

The Similarity Calculation

The goal of the metrics presented above is to approximate the difference in value functions when a policy π_1 is transferred from one MDP to another. This depends on two things:

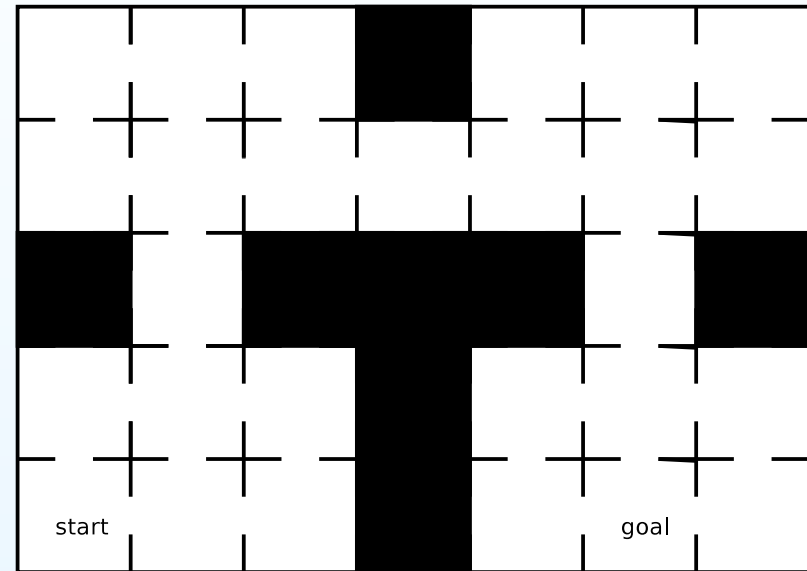
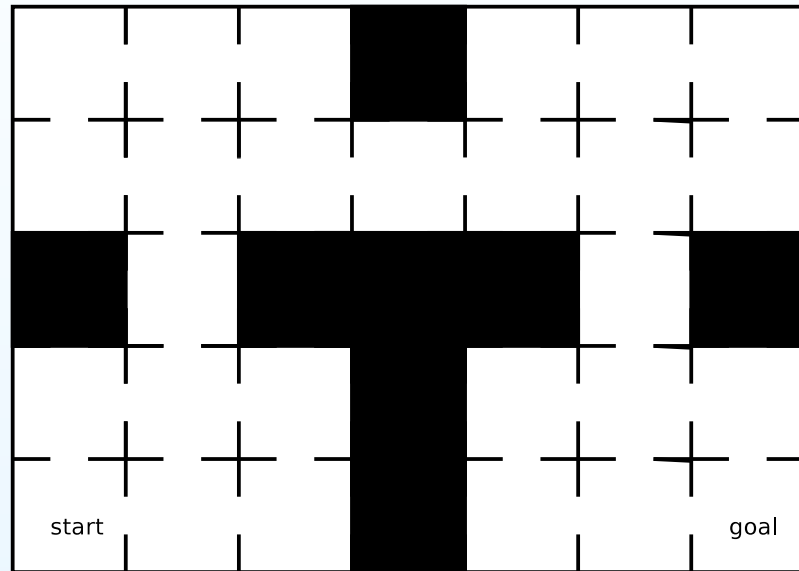
1. How close to optimal was π_1 to begin with?
2. How similar are the MDPs?

Our bound:

$$\|V^{\pi_2} - V_2^*\| \leq \frac{2}{1 - \gamma} \max_{s \in S_1} d(s, \rho(s)) + \frac{1 + \gamma}{1 - \gamma} \|V^{\pi_1} - V_1^*\|$$

Experimental Results

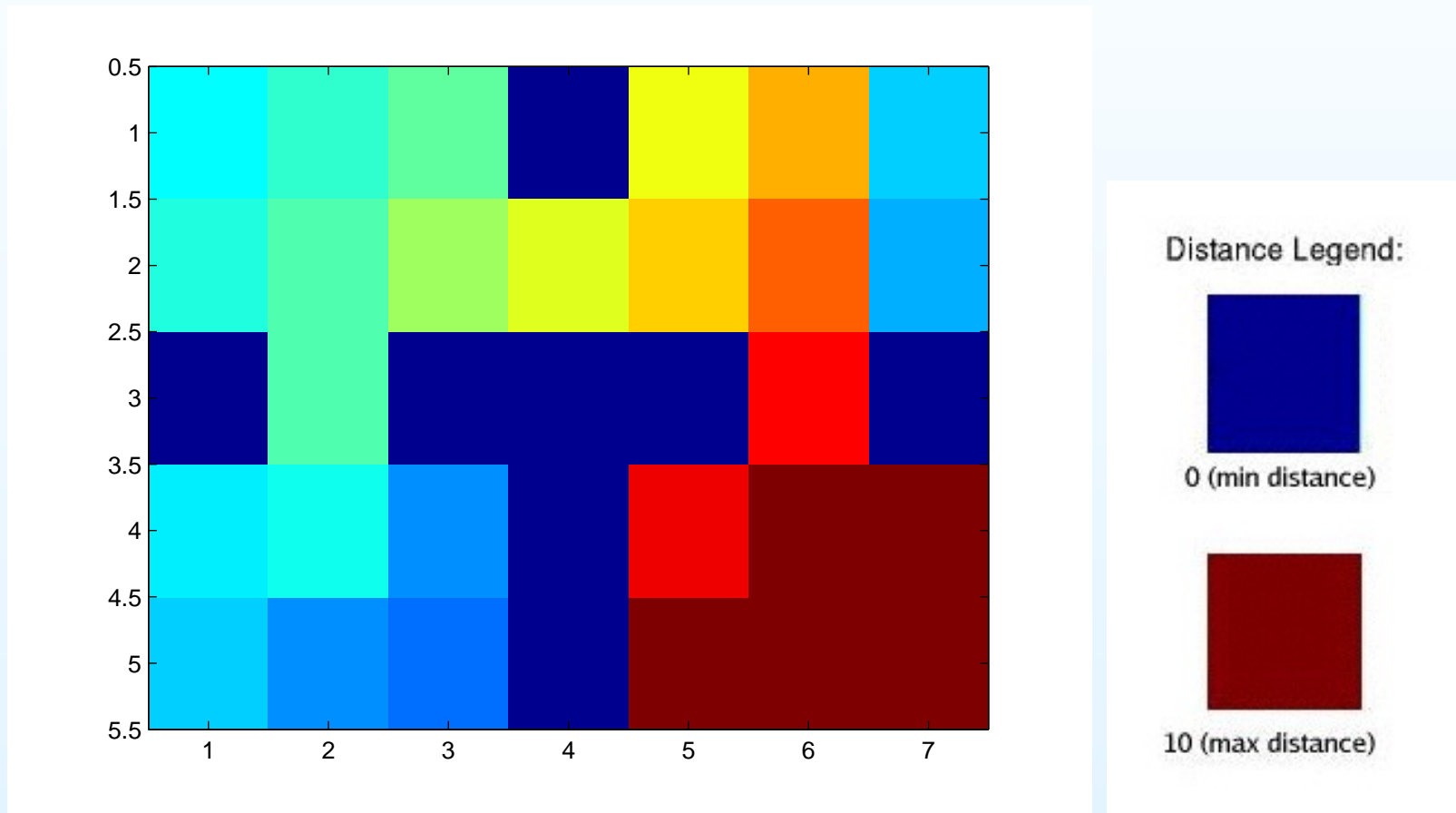
Our similar MDPs:



Actions: move North, move South, move East, move West

Experimental Results

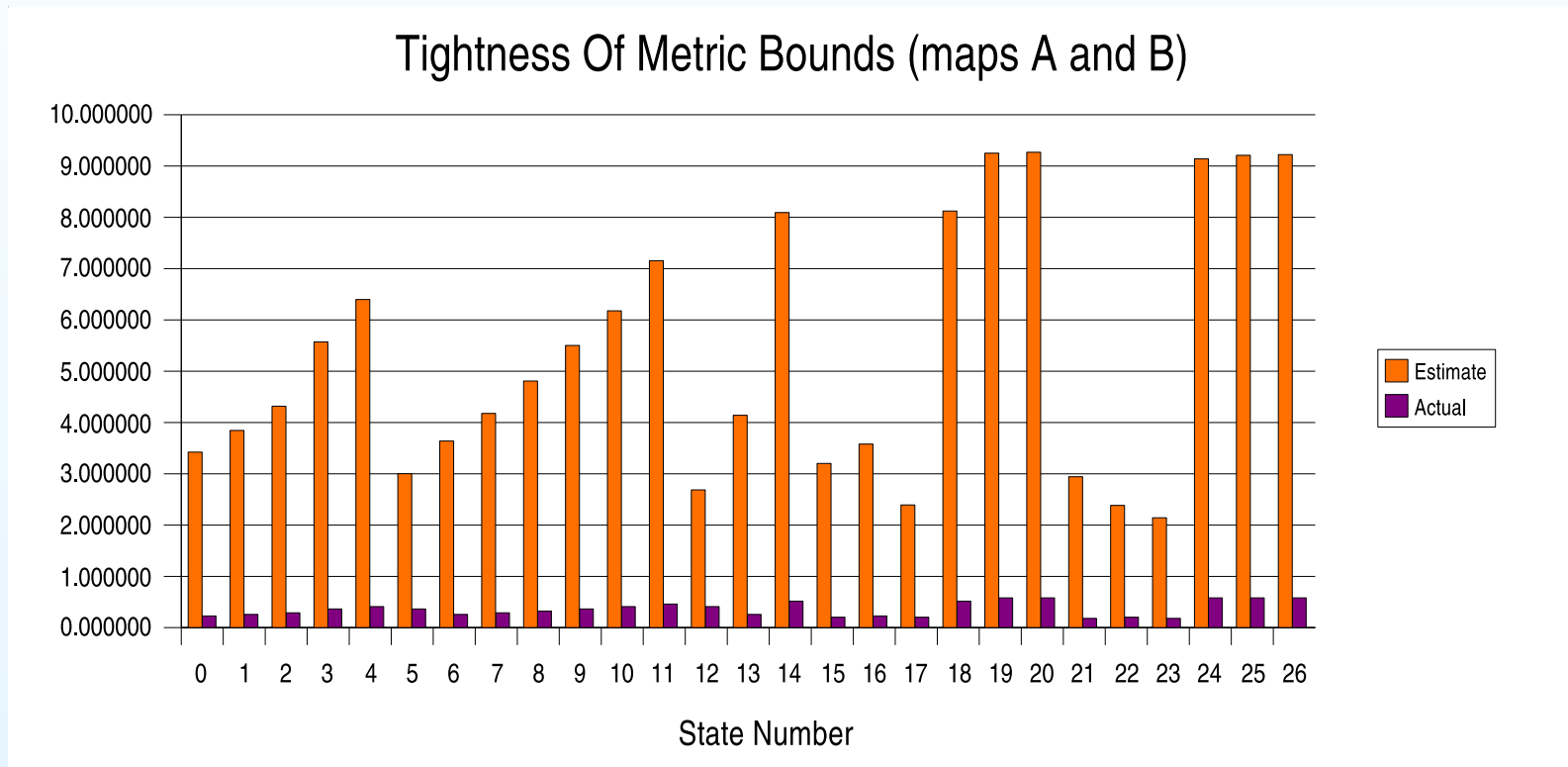
Metric distances between corresponding states in the two MDPs:



Distance between states increases as the goal approaches.

Experimental Results

Estimated distances compared to actual difference in value functions:



The metric overestimates the difference in value functions.

Why the Big Discrepancy?

- The metric is computed independent of the policy we are trying to transfer

Why the Big Discrepancy?

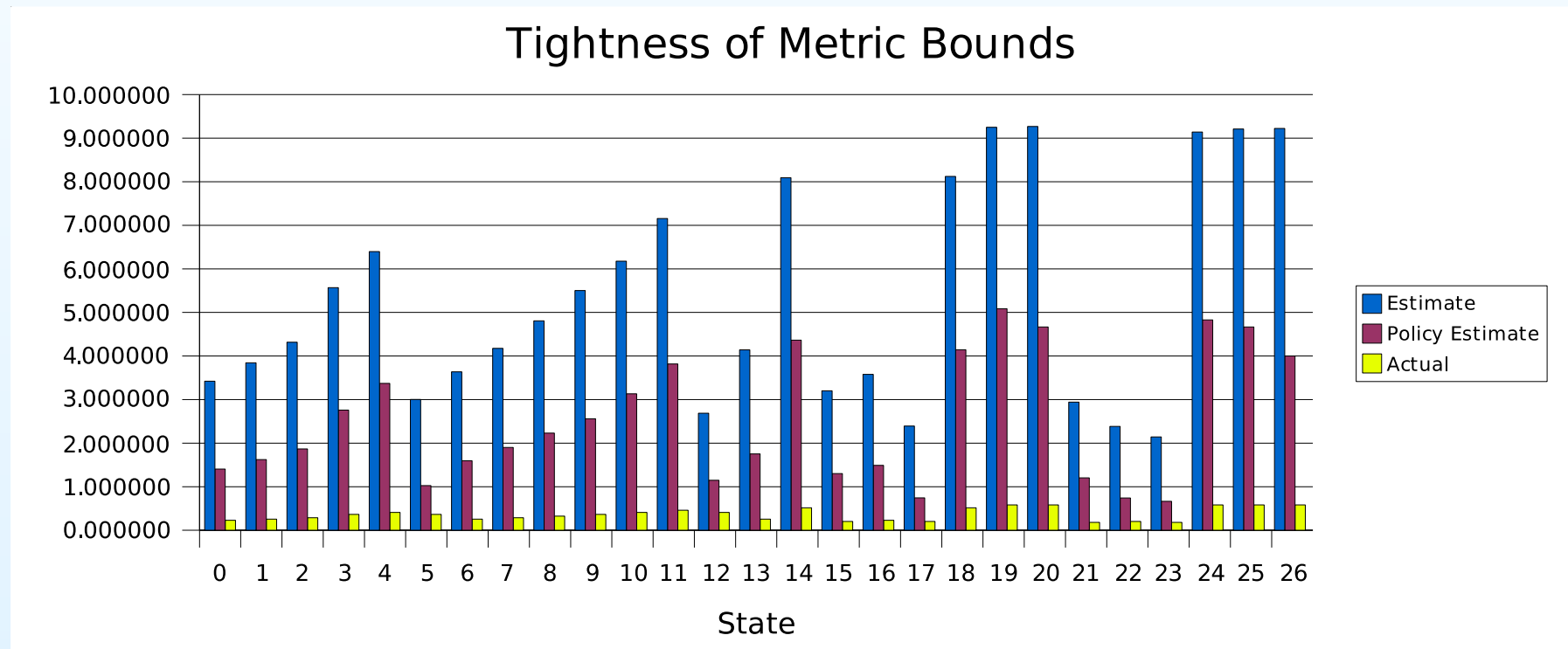
- The metric is computed independent of the policy we are trying to transfer
- It assumes the policy which maximizes the difference in value functions

Why the Big Discrepancy?

- The metric is computed independent of the policy we are trying to transfer
- It assumes the policy which maximizes the difference in value functions
- What if we incorporate the policy we are trying to transfer into the distance measure?

The Fixed Policy Distance Measure:

$$d(s_k, t_j) = \sum_{a \in A} |\pi_1(s_k, a) * (R_{s_k}^a - R_{t_j}^a)| + \gamma * \sum_{a \in A} (\pi_1(s_k, a) * T_k(d)(P_{s_k}^a, Q_{t_j}^a))$$



Pros and Cons of Fixed Policy Metric

- Pros:

Pros and Cons of Fixed Policy Metric

- Pros:
 - Under a reasonable policy, the estimated distance appears to be much closer to the actual difference in value functions

Pros and Cons of Fixed Policy Metric

- Pros:
 - Under a reasonable policy, the estimated distance appears to be much closer to the actual difference in value functions
- Cons:

Pros and Cons of Fixed Policy Metric

- Pros:
 - Under a reasonable policy, the estimated distance appears to be much closer to the actual difference in value functions
- Cons:
 - What is a reasonable policy?

Pros and Cons of Fixed Policy Metric

- Pros:
 - Under a reasonable policy, the estimated distance appears to be much closer to the actual difference in value functions
- Cons:
 - What is a reasonable policy?
 - **The new measure is no longer a metric**

Summary

What We Did

- Used Bisimulation to measure the distance between MDPs

Summary

What We Did

- Used Bisimulation to measure the distance between MDPs
- Transferred a policy by mapping states from one MDP to another

Summary

What We Did

- Used Bisimulation to measure the distance between MDPs
- Transferred a policy by mapping states from one MDP to another
- Bounded the suboptimality of the new policy

Summary

What We Did

- Used Bisimulation to measure the distance between MDPs
- Transferred a policy by mapping states from one MDP to another
- Bounded the suboptimality of the new policy
- Determined through experiments (many of which are not shown here) that the bisimulation method was too pessimistic for our purposes

Summary

What We Did

- Used Bisimulation to measure the distance between MDPs
- Transferred a policy by mapping states from one MDP to another
- Bounded the suboptimality of the new policy
- Determined through experiments (many of which are not shown here) that the bisimulation method was too pessimistic for our purposes

What Next?

- Use the original metric, but with larger time-steps
- Modify the Fixed-Policy measure so that it retains some of the properties of a metric