Recent Algorithmic Developments for the Markov Decision/Game Process

Yinyu Ye

Frontiers in Operations Research/Operations Management

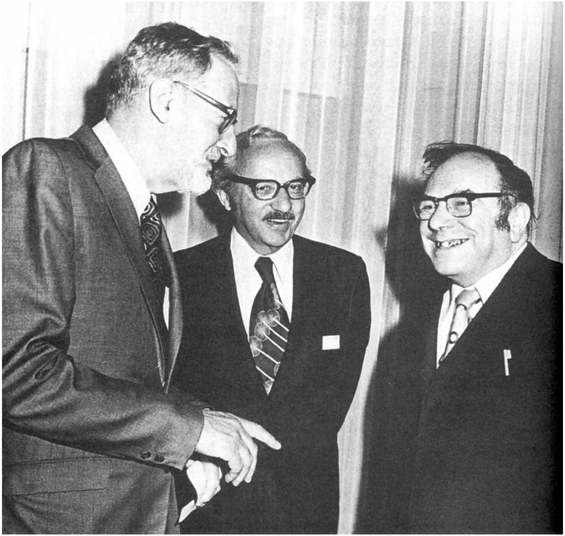
[1]Department of Management Science and Engineering and
Institute for Computational and Mathematical Engineering
Stanford University, Stanford

January 4, 2018

Table of Contents

Table of Contents

Linear Programming (LP)



Algebra of Linear Programming

$$\begin{array}{lll} \text{minimize}_{\mathbf{x}} & \mathbf{c}^T\mathbf{x} & \\ \text{subject to} & A\mathbf{x} & = \mathbf{b}, \\ & \mathbf{x} & \geq \mathbf{0}, \end{array}$$

where given constraint matrix A is an $m \times n$ matrix, the right-hand $\mathbf{b}$ is an m-dimensional vector, the objective coefficients $\mathbf{c}$ is an n-dimensional vector.

Algebra of Linear Programming

$$\begin{array}{lll} \text{minimize}_{\mathbf{x}} & \mathbf{c}^T\mathbf{x} & \\ \text{subject to} & A\mathbf{x} & = \mathbf{b}, \\ & \mathbf{x} & \geq \mathbf{0}, \end{array}$$

where given constraint matrix A is an $m \times n$ matrix, the right-hand $\mathbf{b}$ is an m-dimensional vector, the objective coefficients $\mathbf{c}$ is an n-dimensional vector.

This is a Standard Primal LP form, where variables $\mathbf{x}$ is an n-dimensional vector and need to be optimally decided.

Algebra of Linear Programming

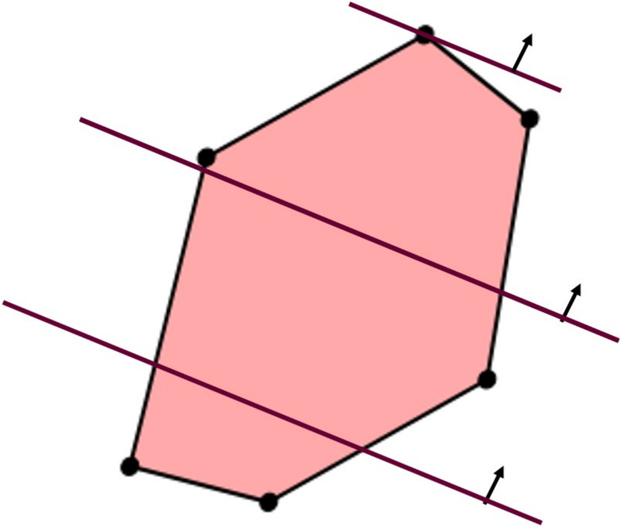
$$\begin{array}{lll} \text{minimize}_{\mathbf{x}} & \mathbf{c}^T\mathbf{x} & \\ \text{subject to} & A\mathbf{x} & = \mathbf{b}, \\ & \mathbf{x} & \geq \mathbf{0}, \end{array}$$

where given constraint matrix A is an $m \times n$ matrix, the right-hand $\mathbf{b}$ is an m-dimensional vector, the objective coefficients $\mathbf{c}$ is an n-dimensional vector.

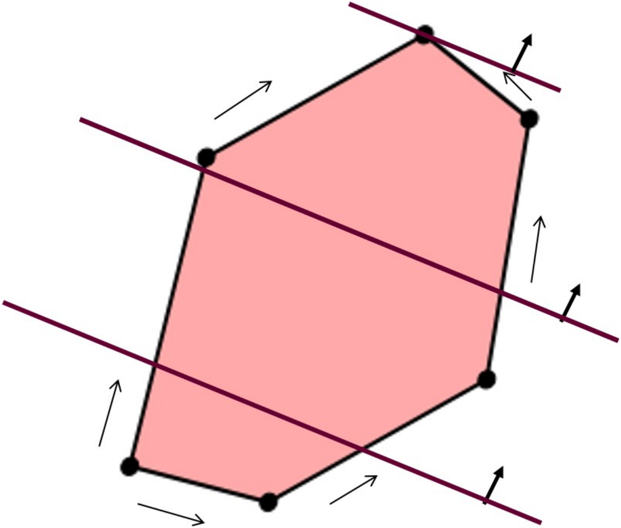
This is a Standard Primal LP form, where variables $\mathbf{x}$ is an n-dimensional vector and need to be optimally decided.

LP is a data-driven computation/decision model that is one of the most used computational problems.

Geometry of Linear Programming



LP Algorithms: the Simplex Method



LP Algorithms: the Interior-Point Method

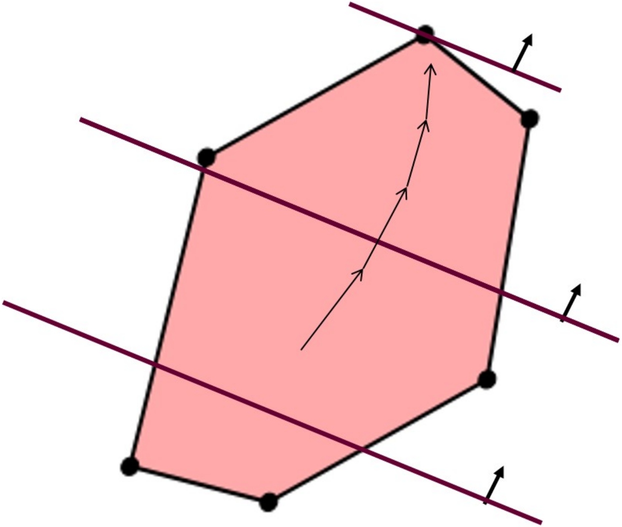


Table of Contents

The Markov Decision/Game Process

- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.

The Markov Decision/Game Process

- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
- Markov game processes (MGPs) provide a mathematical framework for modeling sequential decision-making of two-person turn-based zero-sum game.

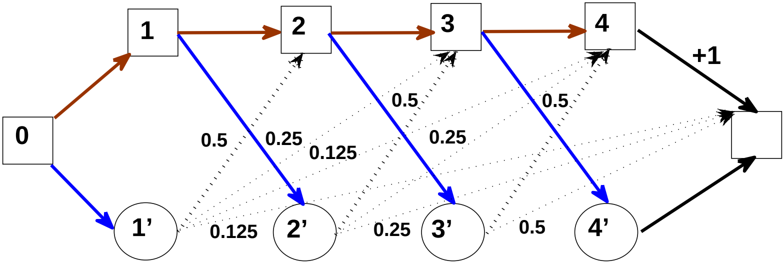
The Markov Decision/Game Process

- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
- Markov game processes (MGPs) provide a mathematical framework for modeling sequential decision-making of two-person turn-based zero-sum game.
- MDGPs are useful for studying a wide range of optimization/game problems solved via dynamic programming, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).

The Markov Decision/Game Process

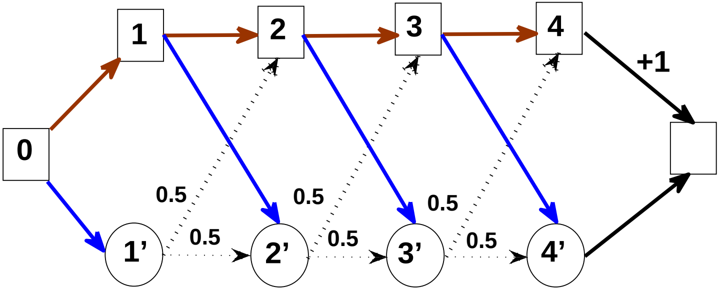
- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
- Markov game processes (MGPs) provide a mathematical framework for modeling sequential decision-making of two-person turn-based zero-sum game.
- MDGPs are useful for studying a wide range of optimization/game problems solved via dynamic programming, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).
- Modern applications include dynamic planning under uncertainty, reinforcement learning, social networking, and almost all other stochastic dynamic/sequential decision/game problems in Mathematical, Physical, Management and Social Sciences.

An MDGP Toy Example I



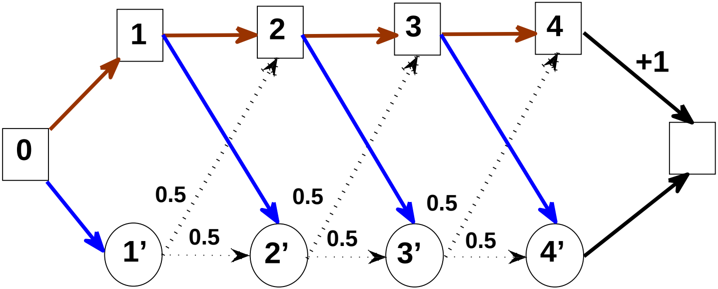
Actions are in red, blue and black; and all actions have zero cost except the state 4 to the termination state (Melekopoglou and Condon 1990). Which actions to take from every state to minimize the total cost?

Toy Example: Simplified Representation



Actions are in red, blue and black; and all actions have immediate zero cost except the state 4 to the termination state.

Toy Example: Game Setting



States $\{0, 1, 2\}$ minimize, while States $\{3, 4\}$ maximize.

The Markov Decision Process/Game continued

- At each time step, the process is in some state $i = 1, ..., m$, and the decision maker chooses an action $j \in \mathcal{A}_i$ that is available in state i, and giving the decision maker an immediate corresponding cost c_j.

The Markov Decision Process/Game continued

- At each time step, the process is in some state $i = 1, ..., m$, and the decision maker chooses an action $j \in \mathcal{A}_i$ that is available in state i, and giving the decision maker an immediate corresponding cost c_j.
- The process responds at the next time step by randomly moving into a new state i'. The probability that the process enters i' is influenced by the chosen action in state i. Specifically, it is given by the state transition distribution probability $\mathbf{p}_j \in \mathbf{R}^m$.

The Markov Decision Process/Game continued

- At each time step, the process is in some state $i = 1, ..., m$, and the decision maker chooses an action $j \in \mathcal{A}_i$ that is available in state i, and giving the decision maker an immediate corresponding cost c_j.
- The process responds at the next time step by randomly moving into a new state i'. The probability that the process enters i' is influenced by the chosen action in state i. Specifically, it is given by the state transition distribution probability $\mathbf{p}_j \in \mathbf{R}^m$.
- But given state/action j, the distribution is conditionally independent of all previous states and actions; in other words, the state transitions of an MDP possess the Markov property.

MDP Stationary Policy and Cost-to-Go Value

- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will always choose; which also lead to a cost-to-go value for each state

MDP Stationary Policy and Cost-to-Go Value

- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will always choose; which also lead to a cost-to-go value for each state
- The MDP is to find a stationary policy to minimize/maximize the expected discounted sum over the infinite horizon with a discount factor $0 \leq \gamma < 1$.

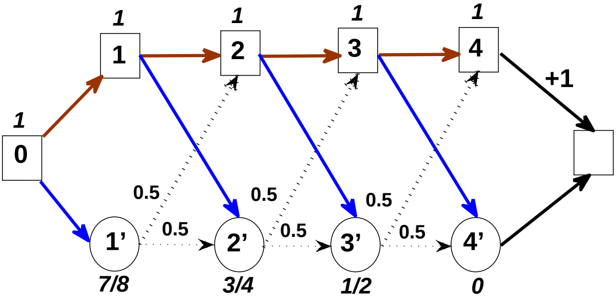
MDP Stationary Policy and Cost-to-Go Value

- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will always choose; which also lead to a cost-to-go value for each state
- The MDP is to find a stationary policy to minimize/maximize the expected discounted sum over the infinite horizon with a discount factor $0 \leq \gamma < 1$.
- If the states are partitioned into two sets, one is to minimize and the other is to maximize the discounted sum, then the process becomes a two-person turn-based zero-sum stochastic game.

MDP Stationary Policy and Cost-to-Go Value

- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will always choose; which also lead to a cost-to-go value for each state
- The MDP is to find a stationary policy to minimize/maximize the expected discounted sum over the infinite horizon with a discount factor $0 \leq \gamma < 1$.
- If the states are partitioned into two sets, one is to minimize and the other is to maximize the discounted sum, then the process becomes a two-person turn-based zero-sum stochastic game.
- Typically, discount factor $\gamma = \frac{1}{1+\rho}$ where ρ is the interest rate, where we assume it is uniform among all actions.

The Cost-to-Go Values of the States



Cost-to-go values on each state when actions in red are taken (the current policy is not optimal).

The Optimal Cost-to-Go Value Vector

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, one entry for each state i, of a given policy.

The Optimal Cost-to-Go Value Vector

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, one entry for each state i, of a given policy.

The MDP problem entails choosing the optimal value vector $\mathbf{y}^*$ such that it is the fixed point:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i,$$

with optimal policy

$$\pi_i^* = \arg\min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i.$$

The Optimal Cost-to-Go Value Vector

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, one entry for each state i, of a given policy.

The MDP problem entails choosing the optimal value vector $\mathbf{y}^*$ such that it is the fixed point:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i,$$

with optimal policy

$$\pi_i^* = \arg\min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i.$$

In the Game setting, the fixed point becomes:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^-,$$

and

$$y_i^* = \max\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^+.$$

The Linear Programming Form of the MDP

The fixed-point vector can be formulated as

$$
\begin{array}{llll}
\text{maximize}_{\mathbf{y}} & \sum_{i=1}^{m} y_i & & \\
\text{subject to} & y_1 & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_1 \\
& \ldots & \ldots & \ldots \\
& y_i & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_i \\
& \ldots & \ldots & \ldots \\
& y_m & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_m,
\end{array}
$$

where $\mathcal{A}_i$ represents all actions available in state i, and $\mathbf{p}_j$ is the state transition probabilities to all states when action j is taken.

The Linear Programming Form of the MDP

The fixed-point vector can be formulated as

$$\begin{array}{llll} \text{maximize}_{\mathbf{y}} & \sum_{i=1}^m y_i & & \\ \\ \text{subject to} & y_1 & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_1 \\ & \ldots & \ldots & \ldots \\ & y_i & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_i \\ & \ldots & \ldots & \ldots \\ & y_m & \leq & c_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ \forall j \in \mathcal{A}_m, \end{array}$$

where $\mathcal{A}_i$ represents all actions available in state i, and $\mathbf{p}_j$ is the state transition probabilities to all states when action j is taken.

This is the Standard Dual LP form.

The Primal LP Form of the MDP

$$\begin{array}{lll} \text{minimize}_{\mathbf{x}} & \sum_{j=1}^{n} x_j & \\ \text{subject to} & \sum_{j=1}^{n}(e_{ij} - \gamma p_{ij})x_j & = 1, \ \forall i, \\ & x_j & \geq 0, \ \forall j. \end{array}$$

where $e_{ij} = 1$ when $j \in \mathcal{A}_i$ and 0 otherwise.

The Primal LP Form of the MDP

$$
\begin{array}{lll}
\text{minimize}_{\mathbf{x}} & \sum_{j=1}^{n} x_j & \\
\text{subject to} & \sum_{j=1}^{n} (e_{ij} - \gamma p_{ij}) x_j & = 1, \ \forall i, \\
& x_j & \geq 0, \ \forall j.
\end{array}
$$

where $e_{ij} = 1$ when $j \in \mathcal{A}_i$ and 0 otherwise.

Primal variable x_j represents the expected jth action flow or frequency, that is, the expected present value of the number of times action j is chosen. The cost-to-go values are the "shadow Prices" of the LP problem.

The Primal LP Form of the MDP

$$\begin{array}{lll} \text{minimize}_{\mathbf{x}} & \sum_{j=1}^{n} x_j & \\ \text{subject to} & \sum_{j=1}^{n} (e_{ij} - \gamma p_{ij}) x_j & = 1, \ \forall i, \\ & x_j & \geq 0, \ \forall j. \end{array}$$

where $e_{ij} = 1$ when $j \in \mathcal{A}_i$ and 0 otherwise.

Primal variable x_j represents the expected jth action flow or frequency, that is, the expected present value of the number of times action j is chosen. The cost-to-go values are the "shadow Prices" of the LP problem.

When discount factor γ becomes γ_j, then the MDP has a non-uniform discount factors.

Algorithmic Events of the MDP Methods

- Shapley (1953) and Bellman (1957) developed a method called the Value-Iteration (VI) method to approximate the optimal state cost-to-go values and an approximate optimal policy.

Algorithmic Events of the MDP Methods

- Shapley (1953) and Bellman (1957) developed a method called the Value-Iteration (VI) method to approximate the optimal state cost-to-go values and an approximate optimal policy.
- Another best known method is due to Howard (1960) and is known as the Policy-Iteration (PI) method, which generate an optimal policy in finite number of iterations in a distributed and decentralized way, where two key procedures are the policy evaluation and the policy improvement.

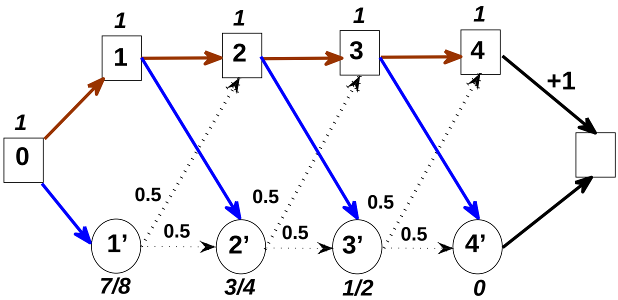
Algorithmic Events of the MDP Methods

- Shapley (1953) and Bellman (1957) developed a method called the Value-Iteration (VI) method to approximate the optimal state cost-to-go values and an approximate optimal policy.

- Another best known method is due to Howard (1960) and is known as the Policy-Iteration (PI) method, which generate an optimal policy in finite number of iterations in a distributed and decentralized way, where two key procedures are the policy evaluation and the policy improvement.

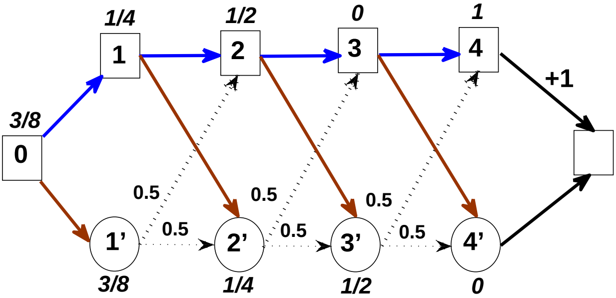
- de Ghellinck (1960), D'Epenoux (1960) and Manne (1960) showed that the MDP has an LP representation, so that it can be solved by the simplex method of Dantzig (1947) in finite number of steps, and the Ellipsoid method of Kachiyan (1979) in polynomial time.

The Policy Improvement



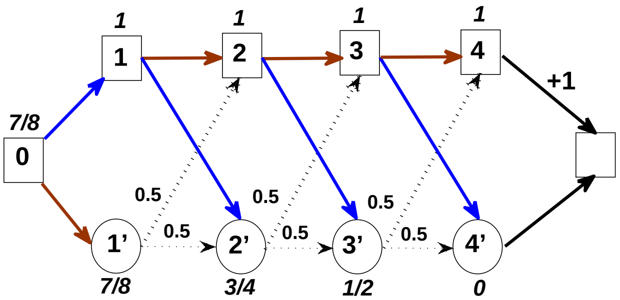
Cost-to-go values on each state when actions in red are taken (the current policy is not optimal), and each state (except State 4) would switch to blue actions.

The Policy Evaluation



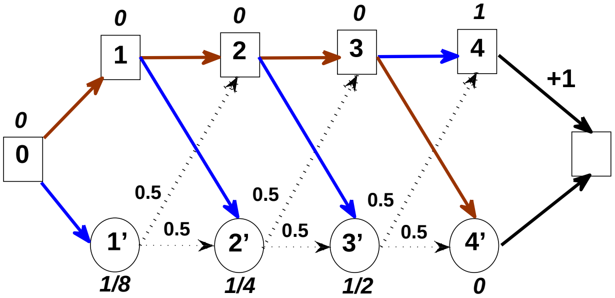
New cost-to-go values on each state when the new set of actions are taken, which are colored in red.

The Simplex or Simple Policy Iteration: index rule



New cost-to-go values on each state when actions in red are taken

The Simplex or Simple Policy Iteration: greedy rule



New cost-to-go values on each state when actions in red are taken

More Algorithmic Events of the MDP Methods

For the discounted MDP:

- Meister and Holzbaur in 1986 showed that the value iteration method generates an optimal policy in polynomial time when the discount γ is fixed, and Bertsekas (1987) and Tseng (1990) showed similar results.

More Algorithmic Events of the MDP Methods

For the discounted MDP:

- Meister and Holzbaur in 1986 showed that the value iteration method generates an optimal policy in polynomial time when the discount γ is fixed, and Bertsekas (1987) and Tseng (1990) showed similar results.
- Puterman in 1994 showed that the policy-iteration method converges no more slowly than the value iteration method, so that it is also a polynomial-time algorithm when γ is fixed.

More Algorithmic Events of the MDP Methods

For the discounted MDP:

- Meister and Holzbaur in 1986 showed that the value iteration method generates an optimal policy in polynomial time when the discount γ is fixed, and Bertsekas (1987) and Tseng (1990) showed similar results.
- Puterman in 1994 showed that the policy-iteration method converges no more slowly than the value iteration method, so that it is also a polynomial-time algorithm when γ is fixed.
- Y (2005) showed that the discounted MDP with fixed discount γ can be solved in strongly polynomial time by a combinatorial interior-point method (CIPM).

More Algorithmic Events of the MDP Methods

For the discounted MDP:

- Meister and Holzbaur in 1986 showed that the value iteration method generates an optimal policy in polynomial time when the discount γ is fixed, and Bertsekas (1987) and Tseng (1990) showed similar results.
- Puterman in 1994 showed that the policy-iteration method converges no more slowly than the value iteration method, so that it is also a polynomial-time algorithm when γ is fixed.
- Y (2005) showed that the discounted MDP with fixed discount γ can be solved in strongly polynomial time by a combinatorial interior-point method (CIPM).
- However, the PI dominates the CIPM In practice.

Polynomial vs Strongly Polynomial

- Polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem with rational data is bounded by a polynomial in m, n, and the total bits, L, of the encoded problem data.

Polynomial vs Strongly Polynomial

- Polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem with rational data is bounded by a polynomial in m, n, and the total bits, L, of the encoded problem data.
- Strongly polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem is bounded by a polynomial in m and n.

Polynomial vs Strongly Polynomial

- Polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem with rational data is bounded by a polynomial in m, n, and the total bits, L, of the encoded problem data.
- Strongly polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem is bounded by a polynomial in m and n.
- The proof of polynomial-time: when the gap between the objective value of the current policy (or BFS) and the optimal one is small than 2^{-L}, the current policy must be optimal.

Polynomial vs Strongly Polynomial

- Polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem with rational data is bounded by a polynomial in m, n, and the total bits, L, of the encoded problem data.
- Strongly polynomial-time algorithms: the computation time of an algorithm, the total number of needed basic arithmetic operations, of solving the problem is bounded by a polynomial in m and n.
- The proof of polynomial-time: when the gap between the objective value of the current policy (or BFS) and the optimal one is small than 2^{-L}, the current policy must be optimal.
- A proof of a strongly polynomial-time algorithm cannot rely on this gap argument – it has to be combinatorial.

Negative Results for the Simplex and Policy-Iteration Methods

- A negative result of Melekopoglou and Condon (1990) showed that a simple policy-iteration method, where in each iteration only the action for the state with the smallest index is updated, needs an exponential number of iterations to compute an optimal policy for a specific MDP problem regardless of discount rates.

Negative Results for the Simplex and Policy-Iteration Methods

- A negative result of Melekopoglou and Condon (1990) showed that a simple policy-iteration method, where in each iteration only the action for the state with the smallest index is updated, needs an exponential number of iterations to compute an optimal policy for a specific MDP problem regardless of discount rates.
- Recently, Fearnley (2010) showed that the policy-iteration method needs an exponential number of iterations for a undiscounted finite-horizon MDP.

Negative Results for the Simplex and Policy-Iteration Methods

- A negative result of Melekopoglou and Condon (1990) showed that a simple policy-iteration method, where in each iteration only the action for the state with the smallest index is updated, needs an exponential number of iterations to compute an optimal policy for a specific MDP problem regardless of discount rates.
- Recently, Fearnley (2010) showed that the policy-iteration method needs an exponential number of iterations for a undiscounted finite-horizon MDP.
- Friedmann, Hansen and Zwick (2011) gave an MDP example that the random pivot rule needs exponentially many steps.

Negative Results for the Simplex and Policy-Iteration Methods

- A negative result of Melekopoglou and Condon (1990) showed that a simple policy-iteration method, where in each iteration only the action for the state with the smallest index is updated, needs an exponential number of iterations to compute an optimal policy for a specific MDP problem regardless of discount rates.
- Recently, Fearnley (2010) showed that the policy-iteration method needs an exponential number of iterations for a undiscounted finite-horizon MDP.
- Friedmann, Hansen and Zwick (2011) gave an MDP example that the random pivot rule needs exponentially many steps.
- Friedman (2011) developed an MDP example that the Zadeh pivot rule needs exponentially many steps.

Complexity of the Policy Iteration and Simplex Methods

- In practice, the policy-iteration method, including the simple policy-iteration or Simplex method, has been remarkably successful and shown to be most effective and widely used.

Complexity of the Policy Iteration and Simplex Methods

- In practice, the policy-iteration method, including the simple policy-iteration or Simplex method, has been remarkably successful and shown to be most effective and widely used.

- In the past 50 years, many efforts have been made to resolve the worst-case complexity issue of the policy-iteration method or the Simplex method, and to answer the question: are they strongly polynomial-time algorithms?

Table of Contents

Complexity Theorem for MDP with Discount

- The classic simplex method (Dantzig pivoting rule) and the policy iteration method, starting from any policy, terminate in

$$\frac{m(n-m)}{1-\gamma} \cdot \log\left(\frac{m^2}{1-\gamma}\right)$$

iterations (Y MOR10).

Complexity Theorem for MDP with Discount

- The classic simplex method (Dantzig pivoting rule) and the policy iteration method, starting from any policy, terminate in

$$\frac{m(n-m)}{1-\gamma} \cdot \log\left(\frac{m^2}{1-\gamma}\right)$$

iterations (Y MOR10).

- The policy-iteration method actually terminates

$$\frac{n}{1-\gamma} \cdot \log\left(\frac{m}{1-\gamma}\right),$$

iterations with at most $O(m^2 n)$ operations per iteration (Hansen/Miltersen/Zwick ACM12).

High Level Ideas of the Proof

- Create a combinatorial event: a (non-optimal) action will never enter the (intermediate) policy again.

High Level Ideas of the Proof

- Create a combinatorial event: a (non-optimal) action will never enter the (intermediate) policy again.
- The event will happen in at most a certain polynomial number of iterations.

High Level Ideas of the Proof

- Create a combinatorial event: a (non-optimal) action will never enter the (intermediate) policy again.
- The event will happen in at most a certain polynomial number of iterations.
- More precisely, after $\frac{m}{1-\gamma} \cdot \log\left(\frac{m^2}{1-\gamma}\right)$ iterations, a new non-optimal action would be implicitly eliminated from appearance in any future policies generated by the simplex or policy-iteration method.

High Level Ideas of the Proof

- Create a combinatorial event: a (non-optimal) action will never enter the (intermediate) policy again.
- The event will happen in at most a certain polynomial number of iterations.
- More precisely, after $\frac{m}{1-\gamma} \cdot \log\left(\frac{m^2}{1-\gamma}\right)$ iterations, a new non-optimal action would be implicitly eliminated from appearance in any future policies generated by the simplex or policy-iteration method.
- The event then repeats for another non-optimal state-action, and there are no more than $(n-m)$ non-optimal actions to eliminate.

The Turn-Based Two-Person Zero-Sum Game

- The states are partitioned into two sets where one set is to maximize and the other is to minimize.

The Turn-Based Two-Person Zero-Sum Game

- The states are partitioned into two sets where one set is to maximize and the other is to minimize.
- It does not admit a convex programming formulation, and it is unknown if it can be solved in polynomial time in general.

The Turn-Based Two-Person Zero-Sum Game

- The states are partitioned into two sets where one set is to maximize and the other is to minimize.
- It does not admit a convex programming formulation, and it is unknown if it can be solved in polynomial time in general.
- Strategy-Iteration Method: One player continues policy iterations from the policy where the other player chooses the best-response action in every one of his or her state set.

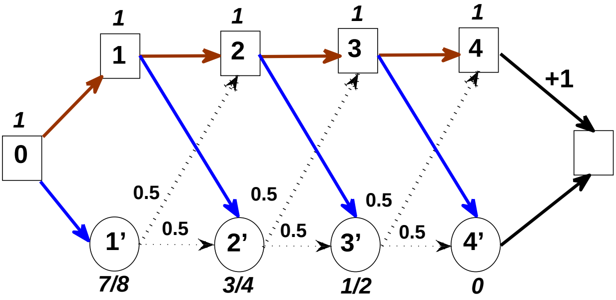
The Turn-Based Two-Person Zero-Sum Game

- The states are partitioned into two sets where one set is to maximize and the other is to minimize.
- It does not admit a convex programming formulation, and it is unknown if it can be solved in polynomial time in general.
- Strategy-Iteration Method: One player continues policy iterations from the policy where the other player chooses the best-response action in every one of his or her state set.
- Hansen/Miltersen/Zwick ACM12 proved that the strategy iteration method also terminates

$$\frac{n}{1-\gamma} \cdot \log\left(\frac{m}{1-\gamma}\right)$$

iterations – the first strongly polynomial time algorithm when the discount factor is fixed.

Strategy-Iteration Method



Recall that states $\{0, 1, 2\}$ minimize, while States $\{3, 4\}$ maximize.

Robust MDP with Discount

- In real MDP applications, the state transition distribution may be uncertain or unknown in advance.

Robust MDP with Discount

- In real MDP applications, the state transition distribution may be uncertain or unknown in advance.
- The robust MDP problem would assume that when the decision maker plays an action, an adversary state would alway choose a worst distribution to maximize the expected cost-to-go value of the decision maker.

Robust MDP with Discount

- In real MDP applications, the state transition distribution may be uncertain or unknown in advance.
- The robust MDP problem would assume that when the decision maker plays an action, an adversary state would alway choose a worst distribution to maximize the expected cost-to-go value of the decision maker.
- This can be exactly formulated as the a Turn-Based Two-Person Zero-Sum Game, so that the strategy iteration method also terminates

$$\frac{n}{1-\gamma} \cdot \log\left(\frac{n+m}{1-\gamma}\right)$$

iterations.

Deterministic MDP with Discount

- Every probability distribution contains exactly one 1 and 0 everywhere else, where the primal LP problem resembles the generalized cycle flow problem.

Deterministic MDP with Discount

- Every probability distribution contains exactly one 1 and 0 everywhere else, where the primal LP problem resembles the generalized cycle flow problem.
- Theorem: The simplex method for deterministic MDP with a uniform discount factor, regardless the factor value, terminates in $O(m^3 n^2 \log^2 m)$ iterations (Post/Y MOR2016).

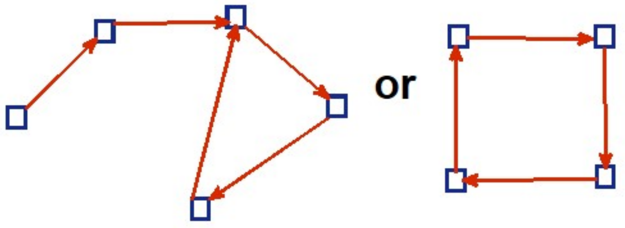
Deterministic MDP with Discount

- Every probability distribution contains exactly one 1 and 0 everywhere else, where the primal LP problem resembles the generalized cycle flow problem.
- Theorem: The simplex method for deterministic MDP with a uniform discount factor, regardless the factor value, terminates in $O(m^3 n^2 \log^2 m)$ iterations (Post/Y MOR2016).
- Theorem: The simplex method for deterministic MDP with non-uniform discount factors, regardless factor values, terminates in $O(m^5 n^3 \log^2 m)$ iterations (Post/Y MOR2016).

Deterministic MDP with Discount

- Every probability distribution contains exactly one 1 and 0 everywhere else, where the primal LP problem resembles the generalized cycle flow problem.
- Theorem: The simplex method for deterministic MDP with a uniform discount factor, regardless the factor value, terminates in $O(m^3 n^2 \log^2 m)$ iterations (Post/Y MOR2016).
- Theorem: The simplex method for deterministic MDP with non-uniform discount factors, regardless factor values, terminates in $O(m^5 n^3 \log^2 m)$ iterations (Post/Y MOR2016).
- Hansen/Miltersen/Zwick 15 were able to reduce a factor m from the bound.

High Level Ideas of the Proof I



- Each chosen action can be either a path-edge or cycle-edge of a policy, and the expected edge flow value is small when it is a path-edge and large when it is a cycle edge.

High Level Ideas of the Proof II

- There two types of pivots: the newly chosen action is either on a path or on a cycle of the new policy.

High Level Ideas of the Proof II

- There two types of pivots: the newly chosen action is either on a path or on a cycle of the new policy.
- In every $m^2 n \log(m)$ consecutive pivot steps, there must be at least one step that is a cycle pivot.

High Level Ideas of the Proof II

- There two types of pivots: the newly chosen action is either on a path or on a cycle of the new policy.
- In every $m^2 n \log(m)$ consecutive pivot steps, there must be at least one step that is a cycle pivot.
- For the uniform discount case, after every $m \log(m)$ cycle pivot steps, there is a non-optimal action that would never be chosen again, and there are at most n actions for such "implicit elimination".

High Level Ideas of the Proof II

- There two types of pivots: the newly chosen action is either on a path or on a cycle of the new policy.
- In every $m^2 n \log(m)$ consecutive pivot steps, there must be at least one step that is a cycle pivot.
- For the uniform discount case, after every $m \log(m)$ cycle pivot steps, there is a non-optimal action that would never be chosen again, and there are at most n actions for such "implicit elimination".
- For the non-uniform discount case, there are n different edge flow value layers...

Table of Contents

The Value-Iteration Method (VI)

Let $\mathbf{y}^0 \in \mathbf{R}^m$ represent the initial cost-to-go values of the m states.

The Value-Iteration Method (VI)

Let $\mathbf{y}^0 \in \mathbf{R}^m$ represent the initial cost-to-go values of the m states.

The VI for MDP:

$$y_i^{k+1} = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k,\ \forall j \in \mathcal{A}_i\},\ \forall i.$$

The Value-Iteration Method (VI)

Let $\mathbf{y}^0 \in \mathbf{R}^m$ represent the initial cost-to-go values of the m states.

The VI for MDP:

$$y_i^{k+1} = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k, \ \forall j \in \mathcal{A}_i\}, \ \forall i.$$

The VI for MGP

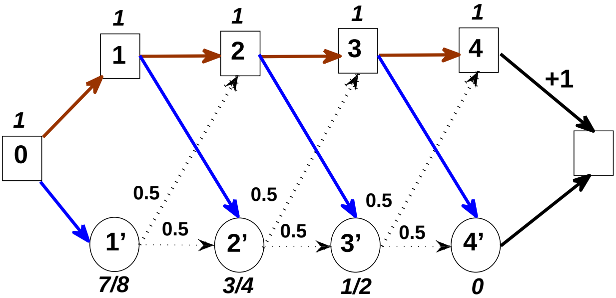
$$y_i^{k+1} = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^-,$$

and

$$y_i^{k+1} = \max\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^+.$$

The values inside the parenthesis are the so-called Q-values.

Value-Iteration Method of MDP



Assume discount $\gamma = 0.9$ and start the value vector $(1, 1, 1, 1, 1, 0)$. The next would be $(0.788, 0.675, 0.45, 0, 1, 0)$ for MDP, and $(0.788, 0.675, 0.45, 0.9, 1, 0)$ for MGP.

Sample Value-Iteration

- Rather than compute each quantity $\mathbf{p}_j^T \mathbf{y}^k$ exactly, we approximate it by sampling, that is, we construct a sparser sample distribution $\hat{\mathbf{p}}_j$ for the evaluation. (Thus, the method does not need to know $\mathbf{p}_j$ exactly).

Sample Value-Iteration

- Rather than compute each quantity $\mathbf{p}_j^T \mathbf{y}^k$ exactly, we approximate it by sampling, that is, we construct a sparser sample distribution $\hat{\mathbf{p}}_j$ for the evaluation. (Thus, the method does not need to know $\mathbf{p}_j$ exactly).
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T \mathbf{y}^k$ takes $O(m)$ up to operations.

Sample Value-Iteration

- Rather than compute each quantity $\mathbf{p}_j^T \mathbf{y}^k$ exactly, we approximate it by sampling, that is, we construct a sparser sample distribution $\hat{\mathbf{p}}_j$ for the evaluation. (Thus, the method does not need to know $\mathbf{p}_j$ exactly).
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T \mathbf{y}^k$ takes $O(m)$ up to operations.
- We analyze this performance using Hoeffdings inequality and classic results on contraction properties of value iteration. Moreover, we improve the final result using Variance Reduction and Monotone Iteration.

Sample Value-Iteration

- Rather than compute each quantity $\mathbf{p}_j^T \mathbf{y}^k$ exactly, we approximate it by sampling, that is, we construct a sparser sample distribution $\hat{\mathbf{p}}_j$ for the evaluation. (Thus, the method does not need to know $\mathbf{p}_j$ exactly).
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T \mathbf{y}^k$ takes $O(m)$ up to operations.
- We analyze this performance using Hoeffdings inequality and classic results on contraction properties of value iteration. Moreover, we improve the final result using Variance Reduction and Monotone Iteration.
- Variance Reduction enables us to update the Q-values so that the needed number of samples is decreased from iteration to iteration.

Sample Value-Iteration Results

Two results are developed (Sidford, Wang, Wu and Y [2017]):

- Knowing $\mathbf{p}_j$:

$$O\left((mn + \frac{n}{(1-\gamma)^3}) \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta})\right)$$

to compute an ϵ-optimal policy with probability at least $1-\delta$.

Sample Value-Iteration Results

Two results are developed (Sidford, Wang, Wu and Y [2017]):

- Knowing $\mathbf{p}_j$:

$$O\left((mn + \frac{n}{(1-\gamma)^3})\log(\frac{1}{\epsilon})\log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

- Sampling:

$$O\left(\frac{n}{(1-\gamma)^4\epsilon^2}\log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

Sample Value-Iteration Results

Two results are developed (Sidford, Wang, Wu and Y [2017]):

- Knowing $\mathbf{p}_j$:

$$O\left((mn + \frac{n}{(1-\gamma)^3})\log(\frac{1}{\epsilon})\log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

- Sampling:

$$O\left(\frac{n}{(1-\gamma)^4\epsilon^2}\log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

- Sample lower bound: $O\left(\frac{n}{(1-\gamma)^3\epsilon^2}\right)$.

Table of Contents

More Results and Extensions

- The Simplex Method result was extended to general non-degenerate LP by Kitahara and Mizuno 2012,

More Results and Extensions

- The Simplex Method result was extended to general non-degenerate LP by Kitahara and Mizuno 2012,
- Renewed exciting research work on the simplex method, e.g., Feinberg/Huang 2013, Lee/Epelman/Romeijn/Smith 2013, Scherrer 2014, Fearnley/Savani 2014, Adler/Papadimitriou/Rubinstein 2014, etc.

More Results and Extensions

- The Simplex Method result was extended to general non-degenerate LP by Kitahara and Mizuno 2012,
- Renewed exciting research work on the simplex method, e.g., Feinberg/Huang 2013, Lee/Epelman/Romeijn/Smith 2013, Scherrer 2014, Fearnley/Savani 2014, Adler/Papadimitriou/Rubinstein 2014, etc.
- Lin, Sidford, Wang, Wu and Y 2018 on approximate PI method to achieve the optimal sample complexity.

More Results and Extensions

- The Simplex Method result was extended to general non-degenerate LP by Kitahara and Mizuno 2012,
- Renewed exciting research work on the simplex method, e.g., Feinberg/Huang 2013, Lee/Epelman/Romeijn/Smith 2013, Scherrer 2014, Fearnley/Savani 2014, Adler/Papadimitriou/Rubinstein 2014, etc.
- Lin, Sidford, Wang, Wu and Y 2018 on approximate PI method to achieve the optimal sample complexity.
- Lin, Sidford, Wang, Wu and Y 2018 on approximate PI method for solving Ergodic MDP where the dependence on γ is removed.

More Results and Extensions

- The Simplex Method result was extended to general non-degenerate LP by Kitahara and Mizuno 2012,
- Renewed exciting research work on the simplex method, e.g., Feinberg/Huang 2013, Lee/Epelman/Romeijn/Smith 2013, Scherrer 2014, Fearnley/Savani 2014, Adler/Papadimitriou/Rubinstein 2014, etc.
- Lin, Sidford, Wang, Wu and Y 2018 on approximate PI method to achieve the optimal sample complexity.
- Lin, Sidford, Wang, Wu and Y 2018 on approximate PI method for solving Ergodic MDP where the dependence on γ is removed.
- All results are extended to the Markov Game Process.

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.
- Multi-updates or pivots work better than a single-update does; policy iteration vs. simplex.

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.
- Multi-updates or pivots work better than a single-update does; policy iteration vs. simplex.
- Tatonnement and decentralized iterative process such as PI works under the Markov property.

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.
- Multi-updates or pivots work better than a single-update does; policy iteration vs. simplex.
- Tatonnement and decentralized iterative process such as PI works under the Markov property.
- Greedy or Steepest descent such as VI works when there is a discount!

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.
- Multi-updates or pivots work better than a single-update does; policy iteration vs. simplex.
- Tatonnement and decentralized iterative process such as PI works under the Markov property.
- Greedy or Steepest descent such as VI works when there is a discount!
- Dynamic sampling over actions in each iteration to deal with a large number of actions in each state?

Remarks and Open Problems

- The performance of the simplex method is very sensitive to the pivoting rule.
- Multi-updates or pivots work better than a single-update does; policy iteration vs. simplex.
- Tatonnement and decentralized iterative process such as PI works under the Markov property.
- Greedy or Steepest descent such as VI works when there is a discount!
- Dynamic sampling over actions in each iteration to deal with a large number of actions in each state?
- Dimension reduction to reduce the number of states?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?
- Is the policy iteration method that is strongly polynomial for the deterministic MDP?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?
- Is the policy iteration method that is strongly polynomial for the deterministic MDP?
- Is there a simplex-type method that is (strongly) polynomial for the deterministic MGP (independent of γ)?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?
- Is the policy iteration method that is strongly polynomial for the deterministic MDP?
- Is there a simplex-type method that is (strongly) polynomial for the deterministic MGP (independent of γ)?
- Is there an MDP algorithm whose running time is PTAS for the general MGP?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?
- Is the policy iteration method that is strongly polynomial for the deterministic MDP?
- Is there a simplex-type method that is (strongly) polynomial for the deterministic MGP (independent of γ)?
- Is there an MDP algorithm whose running time is PTAS for the general MGP?
- Is there a strongly polynomial-time algorithm for MDP regardless the discount factor?

Remarks and Open Problems continued

- A Online MDP algorithms where available actions together with cost/reward come sequentially (e.g., Agrawal, Wang, and Y 2010)?
- Is the policy iteration method that is strongly polynomial for the deterministic MDP?
- Is there a simplex-type method that is (strongly) polynomial for the deterministic MGP (independent of γ)?
- Is there an MDP algorithm whose running time is PTAS for the general MGP?
- Is there a strongly polynomial-time algorithm for MDP regardless the discount factor?
- Is there a strongly polynomial-time algorithm for LP?