

Lower bounds for Howard's algorithm for finding Minimum Mean-Cost Cycles

Thomas Dueholm Hansen¹ Uri Zwick²

¹ Center for Algorithmic Game Theory
Department of Computer Science
Aarhus University, Denmark

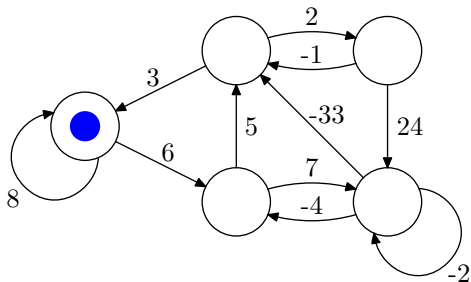
² School of Computer Science
Tel Aviv University, Israel

December 16, 2010



(Deterministic) Markov decision processes

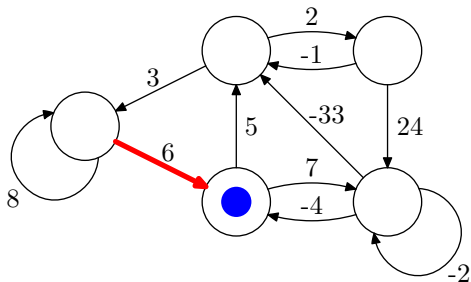
Howard's algorithm is used to solve Markov decision processes (MDPs):



- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

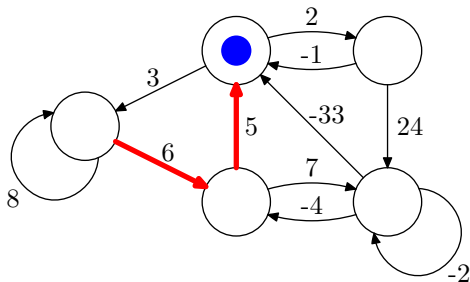


Observed costs: 6

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

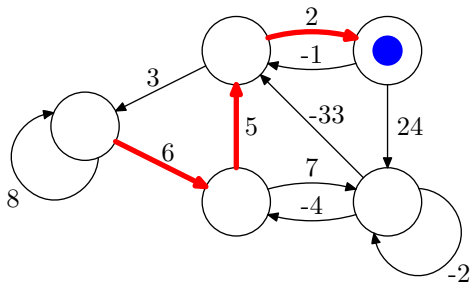


Observed costs: 6 , 5

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

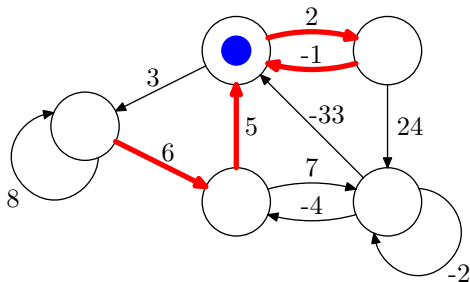


Observed costs: 6 , 5 , 2

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

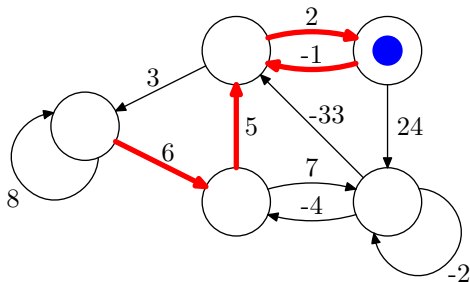


Observed costs: 6 , 5 , 2 , - 1

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

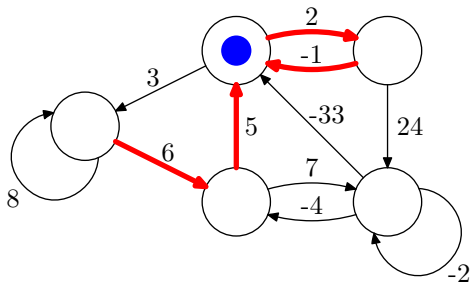


Observed costs: 6 , 5 , 2 , - 1 , 2

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

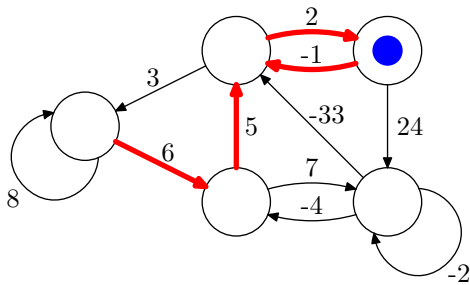


Observed costs: 6 , 5 , 2 , -1 , 2 , -1

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):

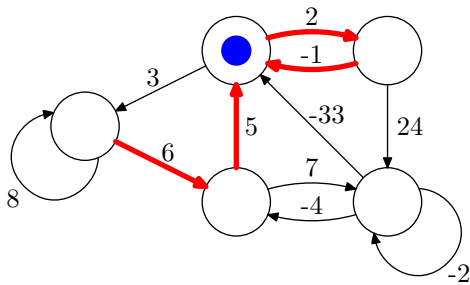


Observed costs: 6 , 5 , 2 , -1 , 2 , -1 , 2

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

(Deterministic) Markov decision processes

Howard's algorithm is used to solve Markov decision processes (MDPs):



Observed costs: 6 , 5 , 2 , -1 , 2 , -1 , 2 , -1 , ...

- Directed, weighted graph $G = (V, E, c)$, $c : E \rightarrow \mathbb{R}$.
- Vertices are called *states* and edges are called *actions*.
- **Token** passed along edges. **Controller** picks edges.
- Goal: Minimize observed edge-weights (costs).

- Let c_0, c_1, c_2, \dots be observed costs.
- Limiting average:

$$\limsup_{k \rightarrow \infty} \frac{1}{k} \sum_{t=0}^{k-1} c_t$$

- Goal: Minimize the limiting average of costs over infinite horizon.

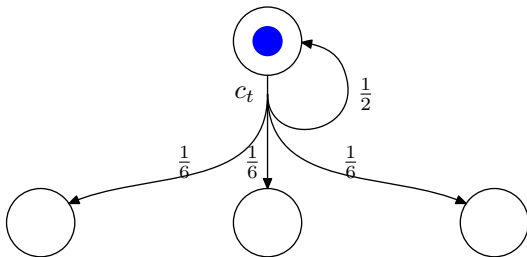
- Let c_0, c_1, c_2, \dots be observed costs.
- Limiting average:

$$\limsup_{k \rightarrow \infty} \frac{1}{k} \sum_{t=0}^{k-1} c_t$$

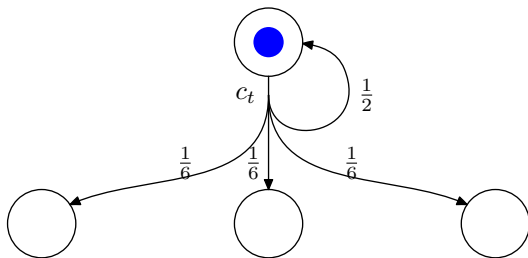
- Goal: Minimize the limiting average of costs over infinite horizon.
- Alternatively, discounted sum of costs:

$$\sum_{t=0}^{\infty} \gamma^t c_t \quad , \quad \gamma < 1$$

- For general MDPs transitions are made according to probability distributions over states:



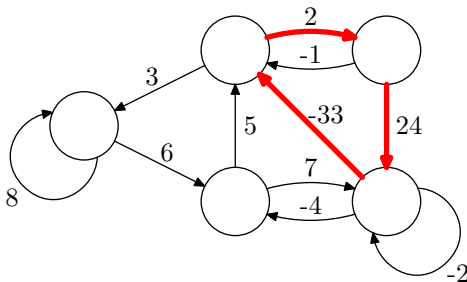
- For general MDPs transitions are made according to probability distributions over states:



- We focus on the case where all transitions are deterministic.

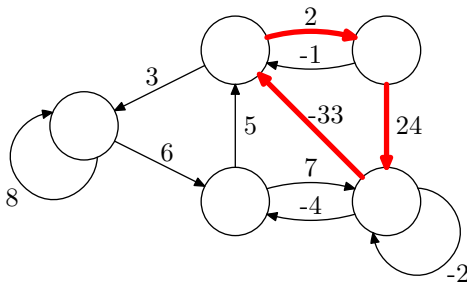
The minimum mean-cost cycle problem

- When solving a deterministic MDP the minimum mean-cost cycle (MMCC) of a weighted directed graph is found:



The minimum mean-cost cycle problem

- When solving a deterministic MDP the minimum mean-cost cycle (MMCC) of a weighted directed graph is found:



- Solving the MMCC problem is, for instance, used as a subroutine in the min-cost flow algorithm of Goldberg and Tarjan (1989).

Let G be a weighted directed graph with n vertices and m edges.

- Karp (1978): $O(mn)$ time algorithm for solving the MMCC problem.

Let G be a weighted directed graph with n vertices and m edges.

- Karp (1978): $O(mn)$ time algorithm for solving the MMCC problem.
- Young, Tarjan and Orlin (1991): $O(mn + n^2 \log n)$ time algorithm, but much faster in practice.

Let G be a weighted directed graph with n vertices and m edges.

- Karp (1978): $O(mn)$ time algorithm for solving the MMCC problem.
- Young, Tarjan and Orlin (1991): $O(mn + n^2 \log n)$ time algorithm, but much faster in practice.
- Dasdan (2004): Howard's algorithm is usually almost as fast as the algorithm of Young et al. according to experimental results.

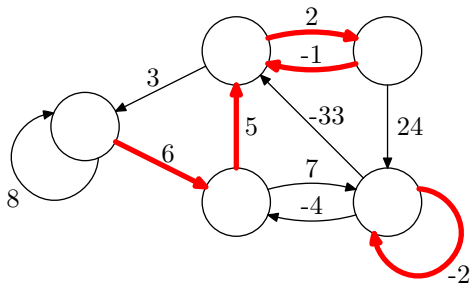
Let G be a weighted directed graph with n vertices and m edges.

- Karp (1978): $O(mn)$ time algorithm for solving the MMCC problem.
- Young, Tarjan and Orlin (1991): $O(mn + n^2 \log n)$ time algorithm, but much faster in practice.
- Dasdan (2004): Howard's algorithm is usually almost as fast as the algorithm of Young et al. according to experimental results.

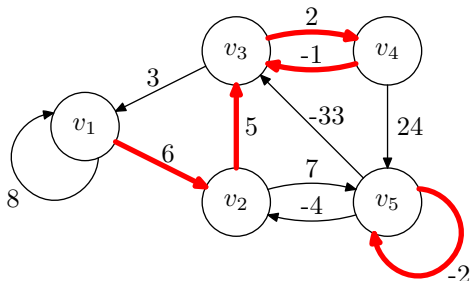
Howard's algorithm works in iterations, each iteration taking $O(m)$ time (for deterministic MDPs).

- **Main result:** Howard's algorithm may require $\Omega(n^2)$ iterations on deterministic MDPs.

Policies and values



- A (positional) policy π is a choice of an edge from each vertex.



- A (positional) policy π is a choice of an edge from each vertex.
- The value, $val_{\pi}(v)$, of a vertex $v \in V$ for a policy π , is the average of costs of the cycle reached from v . For instance:

$$val_{\pi}(v_1) = \frac{2 - 1}{2} = \frac{1}{2} \quad \text{and} \quad val_{\pi}(v_5) = -2$$



$$val_{\pi}(v_1) = val_{\pi}(v_2) = val_{\pi}(v_3) = val_{\pi}(v_4) = \frac{7 - 1}{2} = 3$$

$$pot_{\pi}(v_1) = (6 - 3) + (5 - 3) = 3 + 2 = 5$$

- The potential, $pot_{\pi}(v)$, of a vertex $v \in V$ for a policy π , is the sum of costs on the path to the vertex of lowest index on the cycle reached from v minus for each step the value of v .

- Let $VAL_{\pi}(v) = (val_{\pi}(v), pot_{\pi}(v))$ be the valuation of v .
- Let $VAL_{\pi}(u) < VAL_{\pi}(v)$ iff either
 - $val_{\pi}(u) < val_{\pi}(v)$, or
 - $val_{\pi}(u) = val_{\pi}(v)$ and $pot_{\pi}(u) < pot_{\pi}(v)$.

- Let $VAL_{\pi}(v) = (val_{\pi}(v), pot_{\pi}(v))$ be the valuation of v .
- Let $VAL_{\pi}(u) < VAL_{\pi}(v)$ iff either
 - $val_{\pi}(u) < val_{\pi}(v)$, or
 - $val_{\pi}(u) = val_{\pi}(v)$ and $pot_{\pi}(u) < pot_{\pi}(v)$.
- Let VAL_{π} be the vector of valuations of all vertices.
- Valuation vectors are compared component-wise:
 - $VAL_{\pi} \leq VAL_{\pi'}$ iff $VAL_{\pi}(v) \leq VAL_{\pi'}(v)$ for all vertices $v \in V$.
 - $VAL_{\pi} < VAL_{\pi'}$ iff $VAL_{\pi} \leq VAL_{\pi'}$ and $VAL_{\pi} \neq VAL_{\pi'}$.

- Let $VAL_{\pi}(v) = (val_{\pi}(v), pot_{\pi}(v))$ be the valuation of v .
- Let $VAL_{\pi}(u) < VAL_{\pi}(v)$ iff either
 - $val_{\pi}(u) < val_{\pi}(v)$, or
 - $val_{\pi}(u) = val_{\pi}(v)$ and $pot_{\pi}(u) < pot_{\pi}(v)$.
- Let VAL_{π} be the vector of valuations of all vertices.
- Valuation vectors are compared component-wise:
 - $VAL_{\pi} \leq VAL_{\pi'}$ iff $VAL_{\pi}(v) \leq VAL_{\pi'}(v)$ for all vertices $v \in V$.
 - $VAL_{\pi} < VAL_{\pi'}$ iff $VAL_{\pi} \leq VAL_{\pi'}$ and $VAL_{\pi} \neq VAL_{\pi'}$.
- π is *optimal* if for all π' , $VAL_{\pi} \leq VAL_{\pi'}$.

- Let $VAL_{\pi}(v) = (val_{\pi}(v), pot_{\pi}(v))$ be the valuation of v .
- Let $VAL_{\pi}(u) < VAL_{\pi}(v)$ iff either
 - $val_{\pi}(u) < val_{\pi}(v)$, or
 - $val_{\pi}(u) = val_{\pi}(v)$ and $pot_{\pi}(u) < pot_{\pi}(v)$.
- Let VAL_{π} be the vector of valuations of all vertices.
- Valuation vectors are compared component-wise:
 - $VAL_{\pi} \leq VAL_{\pi'}$ iff $VAL_{\pi}(v) \leq VAL_{\pi'}(v)$ for all vertices $v \in V$.
 - $VAL_{\pi} < VAL_{\pi'}$ iff $VAL_{\pi} \leq VAL_{\pi'}$ and $VAL_{\pi} \neq VAL_{\pi'}$.
- π is *optimal* if for all π' , $VAL_{\pi} \leq VAL_{\pi'}$.
- Bellman (1957): Optimal positional policies are guaranteed to exist.

- Let $VAL_{\pi}(v) = (val_{\pi}(v), pot_{\pi}(v))$ be the valuation of v .
- Let $VAL_{\pi}(u) < VAL_{\pi}(v)$ iff either
 - $val_{\pi}(u) < val_{\pi}(v)$, or
 - $val_{\pi}(u) = val_{\pi}(v)$ and $pot_{\pi}(u) < pot_{\pi}(v)$.
- Let VAL_{π} be the vector of valuations of all vertices.
- Valuation vectors are compared component-wise:
 - $VAL_{\pi} \leq VAL_{\pi'}$ iff $VAL_{\pi}(v) \leq VAL_{\pi'}(v)$ for all vertices $v \in V$.
 - $VAL_{\pi} < VAL_{\pi'}$ iff $VAL_{\pi} \leq VAL_{\pi'}$ and $VAL_{\pi} \neq VAL_{\pi'}$.
- π is *optimal* if for all π' , $VAL_{\pi} \leq VAL_{\pi'}$.
- Bellman (1957): Optimal positional policies are guaranteed to exist.
- Goal: Compute optimal policy.

- Let π be a policy and $\pi[e]$ be obtained from π by switching to the edge e . If $VAL_{\pi[e]} < VAL_{\pi}$, e is an improving switch.

- Let π be a policy and $\pi[e]$ be obtained from π by switching to the edge e . If $VAL_{\pi[e]} < VAL_{\pi}$, e is an improving switch.
- Define the appraisal of an edge (u, v) for a policy π as:

$$A_{\pi}(u, v) = (val_{\pi}(v), c(u, v) - val_{\pi}(v) + pot_{\pi}(v))$$

- An edge (u, v) is an improving switch w.r.t. π iff $A_{\pi}(u, v) < A_{\pi}(u, \pi(u))$.

Lemma (Howard (1960))

Let π' be obtained from π by jointly performing any non-empty set of improving switches. Then $VAL_{\pi'} < VAL_{\pi}$.

Lemma (Howard (1960))

A policy π is optimal iff there are no improving switches.

Function POLICY-ITERATION(π)

while \exists *improving switch w.r.t.* π **do**

 | Update π by performing improving switches

return π

Function POLICY-ITERATION(π)

while \exists *improving switch w.r.t. π* **do**

 Update π by performing improving switches

return π

- Howard's algorithm: Perform as many improving switches as possible. More precisely, for all $u \in V$:

$$\pi(u) \leftarrow \operatorname{argmin}_{v:(u,v) \in E} A_{\pi}(u, v).$$

Bounds for Howard's policy iteration algorithm for MDPs with n states and m actions:

- Mansour and Singh (1999): $O(2^n/n)$ iterations (two actions per state).

Bounds for Howard's policy iteration algorithm for MDPs with n states and m actions:

- Mansour and Singh (1999): $O(2^n/n)$ iterations (two actions per state).
- Madani (2008, personal communication): $2n - O(1)$ lower bound for deterministic MDPs, $m = \Theta(n^2)$.

Bounds for Howard's policy iteration algorithm for MDPs with n states and m actions:

- Mansour and Singh (1999): $O(2^n/n)$ iterations (two actions per state).
- Madani (2008, personal communication): $2n - O(1)$ lower bound for deterministic MDPs, $m = \Theta(n^2)$.
- Fearnley (2010): $2^{\Omega(n)}$ iterations for general MDPs, $m = \Theta(n)$.

Bounds for Howard's policy iteration algorithm for MDPs with n states and m actions:

- Mansour and Singh (1999): $O(2^n/n)$ iterations (two actions per state).
- Madani (2008, personal communication): $2n - O(1)$ lower bound for deterministic MDPs, $m = \Theta(n^2)$.
- Fearnley (2010): $2^{\Omega(n)}$ iterations for general MDPs, $m = \Theta(n)$.

Bounds for discounted MDPs, discount factor $\gamma < 1$:

- Ye (2010): $O(\frac{mn}{1-\gamma} \log \frac{n}{1-\gamma})$ iterations.

Bounds for Howard's policy iteration algorithm for MDPs with n states and m actions:

- Mansour and Singh (1999): $O(2^n/n)$ iterations (two actions per state).
- Madani (2008, personal communication): $2n - O(1)$ lower bound for deterministic MDPs, $m = \Theta(n^2)$.
- Fearnley (2010): $2^{\Omega(n)}$ iterations for general MDPs, $m = \Theta(n)$.

Bounds for discounted MDPs, discount factor $\gamma < 1$:

- Ye (2010): $O(\frac{mn}{1-\gamma} \log \frac{n}{1-\gamma})$ iterations.
- Hansen, Miltersen and Zwick (2010): $O(\frac{m}{1-\gamma} \log \frac{n}{1-\gamma})$ iterations.

Theorem (Hansen and Zwick (2010))

For any integer n , there exists a weighted directed graph with n vertices on which Howard's algorithm performs $\Omega(n^2)$ iterations.

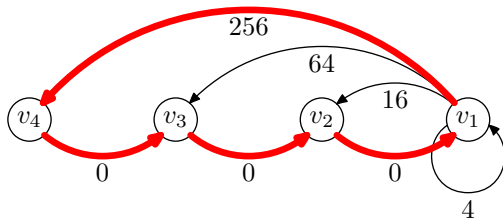
Theorem (Hansen and Zwick (2010))

For any integer n , there exists a weighted directed graph with n vertices on which Howard's algorithm performs $\Omega(n^2)$ iterations.

Main idea:

- Use cycles of very large cost to make Howard's algorithm favor going through many edges (recall that when computing potentials the value was subtracted at every step).

Delaying discovery of cheap cycles



$$\text{val}_{\pi}(v_1) = \frac{256}{4} = 64$$

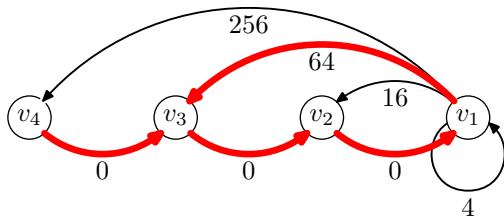
$$A_{\pi}(v_1, v_1)_2 = 4 - 64 = -60$$

$$A_{\pi}(v_1, v_2)_2 = 16 - 2 \cdot 64 = -112$$

$$A_{\pi}(v_1, v_3)_2 = 64 - 3 \cdot 64 = -128$$

$$A_{\pi}(v_1, v_4)_2 = 256 - 4 \cdot 64 = 0$$

Delaying discovery of cheap cycles



$$val_{\pi}(v_1) = \frac{64}{3} \approx 21.33$$

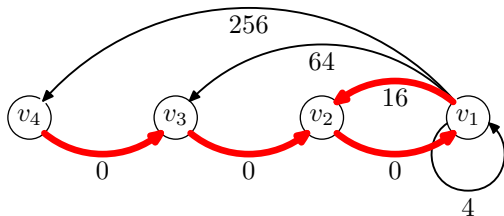
$$A_{\pi}(v_1, v_1)_2 \approx 4 - 21.33 = -17.33$$

$$A_{\pi}(v_1, v_2)_2 \approx 16 - 2 \cdot 21.33 = -26.66$$

$$A_{\pi}(v_1, v_3)_2 = 0$$

$$A_{\pi}(v_1, v_4)_2 \approx 256 - 4 \cdot 21.33 = 170.68$$

Delaying discovery of cheap cycles



$$val_{\pi}(v_1) = \frac{16}{2} = 8$$

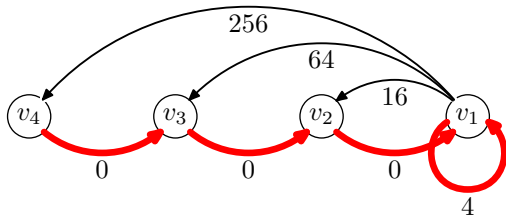
$$A_{\pi}(v_1, v_1)_2 = 4 - 8 = -4$$

$$A_{\pi}(v_1, v_2)_2 = 0$$

$$A_{\pi}(v_1, v_3)_2 = 64 - 3 \cdot 8 = 40$$

$$A_{\pi}(v_1, v_4)_2 = 256 - 4 \cdot 8 = 224$$

Delaying discovery of cheap cycles



$$val_{\pi}(v_1) = 4$$

$$A_{\pi}(v_1, v_1)_2 = 0$$

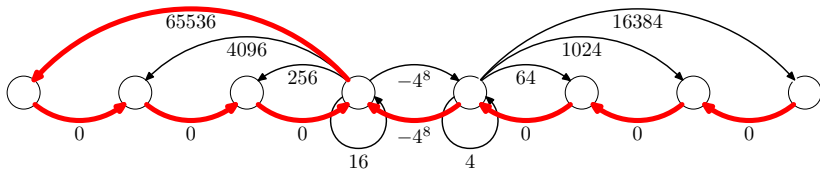
$$A_{\pi}(v_1, v_2)_2 = 16 - 2 \cdot 4 = 8$$

$$A_{\pi}(v_1, v_3)_2 = 64 - 3 \cdot 4 = 52$$

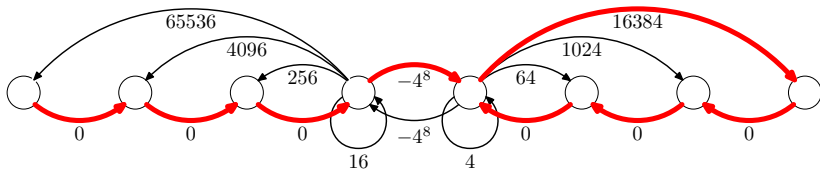
$$A_{\pi}(v_1, v_4)_2 = 256 - 4 \cdot 4 = 240$$

Delaying discovery of cheap cycles

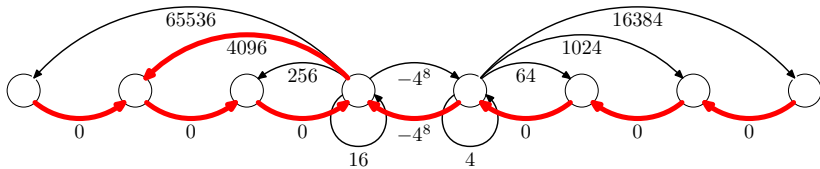
Getting alternating behaviour:



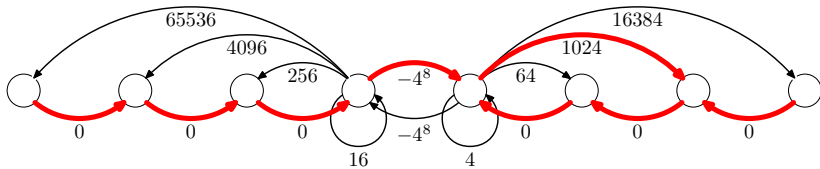
Getting alternating behaviour:



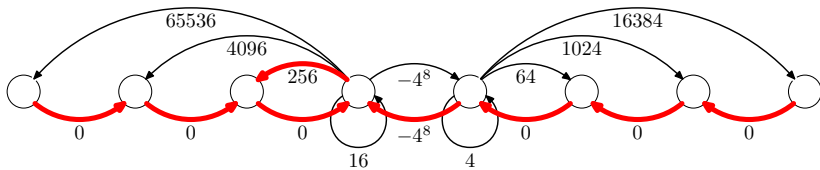
Getting alternating behaviour:



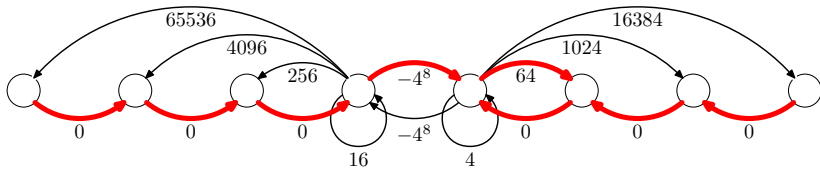
Getting alternating behaviour:



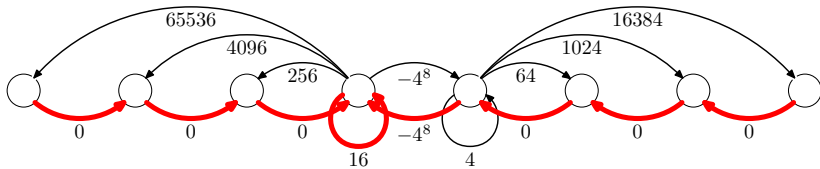
Getting alternating behaviour:



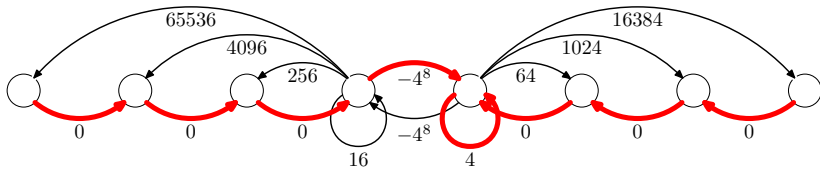
Getting alternating behaviour:



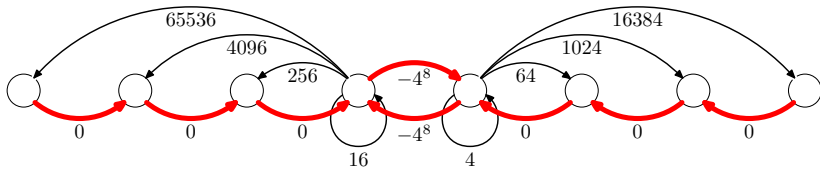
Getting alternating behaviour:



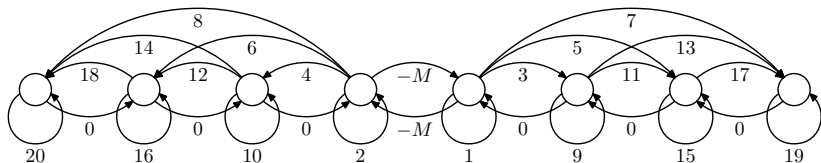
Getting alternating behaviour:



Getting alternating behaviour:

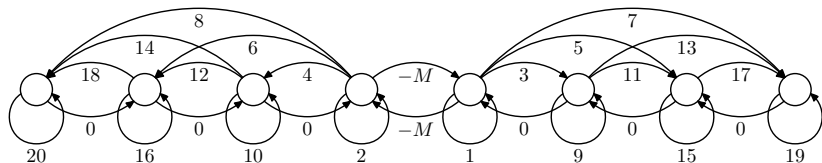


Lower bound construction



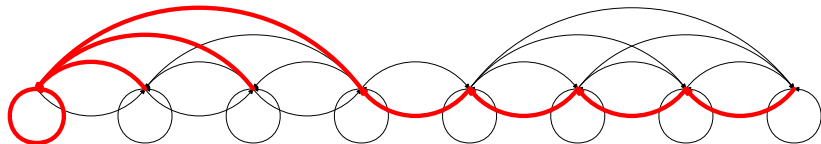
For the example $n = 4$. An edge labeled k has cost n^k , except the edge labeled $-M$ has cost $-n^N$, where $N = n^2 + n$.

Lower bound construction

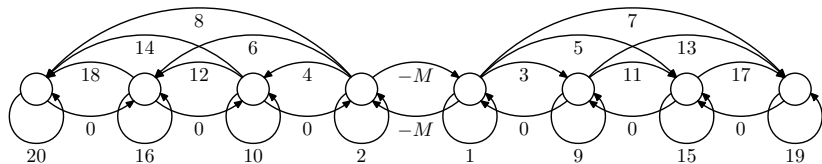


For the example $n = 4$. An edge labeled k has cost n^k , except the edge labeled $-M$ has cost $-n^N$, where $N = n^2 + n$.

Sequence of policies:



Lower bound construction



For the example $n = 4$. An edge labeled k has cost n^k , except the edge labeled $-M$ has cost $-n^N$, where $N = n^2 + n$.

Sequence of policies:

Finding good lower bound examples

- Let a graph and a sequence of policies be given.
- Every update of the policy during some iteration induces constraints on the costs.
- By solving a **linear program** it is possible to check whether there exist costs such that Howard's algorithm generates the sequence.
- To help find a lower bound, we implemented a program that searches for long sequences of policies using this technique.

- Close the gap between $\Omega(n^2)$ and $O(2^n/n)$ for bounds on the number of iterations performed by Howard's algorithm for deterministic MDPs. We conjecture:

Conjecture

The number of iterations performed by Howard's algorithm, when applied to a weighted directed graph, is at most the number of edges in the graph.

- Close the gap between $\Omega(n^2)$ and $O(2^n/n)$ for bounds on the number of iterations performed by Howard's algorithm for deterministic MDPs. We conjecture:

Conjecture

The number of iterations performed by Howard's algorithm, when applied to a weighted directed graph, is at most the number of edges in the graph.

- It is not difficult to modify our construction to make it work for discounted MDPs, giving again an $\Omega(n^2)$ lower bound. Is it possible to get a better lower bound involving the discount factor γ ?

Thank you for listening!