Route Planning under Uncertainty: The Canadian Traveller Problem

based on the paper with the same title by Evdokia Nikolova and David R. Karger (2008)

Seminar 'Optimization under Uncertainty'

Leticia Serejo Kunz

29.05.2024

Content

- What is the Canadian Traveller Problem?

- Heuristic solutions

- General approach: Markov Decision Processes

- Problem: Canadian Traveller is #P-hard

- Optimal Policy for disjoint-path graphs (MDPs)

(- Optimal policy for DAGs (dynamic programming))

- Recap

What is the Canadian Traveller Problem?

First described in 1991 by Papadimitriou & Yannakakis:

"The road map is now known, but the roads with question marks may be unsuitable for travel (say, due to snowfall), an eventuality that is revealed to us only when an adjacent node is reached."

What is the Canadian Traveller Problem?

First described in 1991 by Papadimitriou & Yannakakis:

"The road map is now known, but the roads with question marks may be unsuitable for travel (say, due to snowfall), an eventuality that is revealed to us only when an adjacent node is reached."

In this paper:

- distributions for the costs of edges are given

- upon arriving at a node we see the actual cost values of incident edges (once observed, an edge cost remains fixed)

GOAL: find an optimal *policy* for travelling from source s to destination t

What is the Canadian Traveller Problem?

In this paper:

- distributions for the cost values of the edges are given
- upon arriving at a node we see the actual cost values of incident edges (once observed, an edge cost remains fixed)

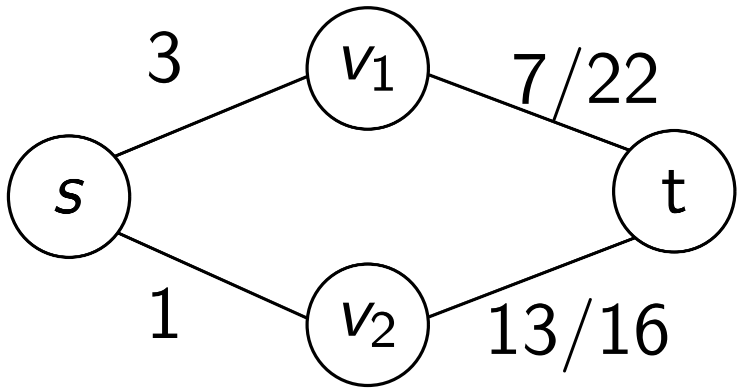
GOAL: find an optimal *policy* for reaching from source s to destination t that minimizes expected cost

What's an optimal policy?

- policy: mapping from perceived states of the environment to probablities of selecting each possible action
- optimal: if the expected cost is smaller than or equal to that of all other policies

The Canadian Traveller Problem: Example

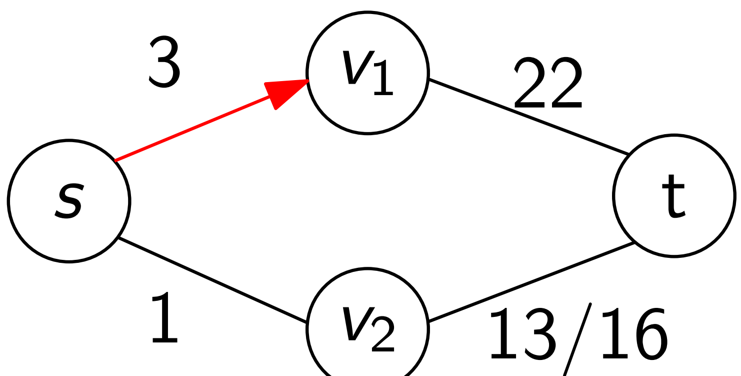
What would you do?



Notation i/j: edge can cost i or j;
both values have probablities of 50%
i and j are positive numbers

The Canadian Traveller Problem: Example

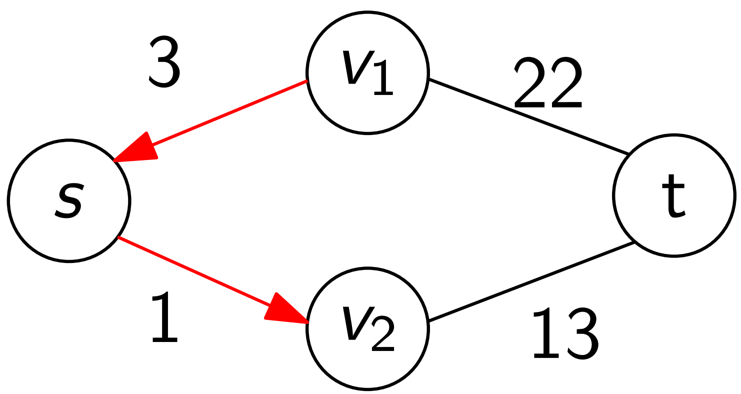
What would you do?



Notation i/j: edge can cost i or j;
both values have probablities of 50%
i and j are positive numbers

The Canadian Traveller Problem: Example

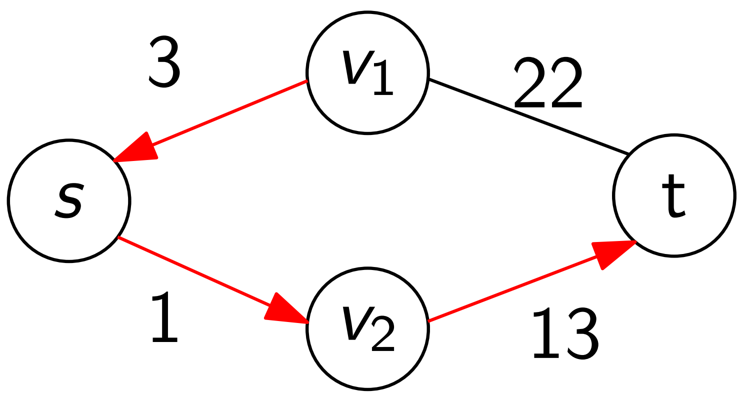
What would you do?



Notation i/j: edge can cost i or j;
both values have probablities of 50%
i and j are positive numbers

The Canadian Traveller Problem: Example

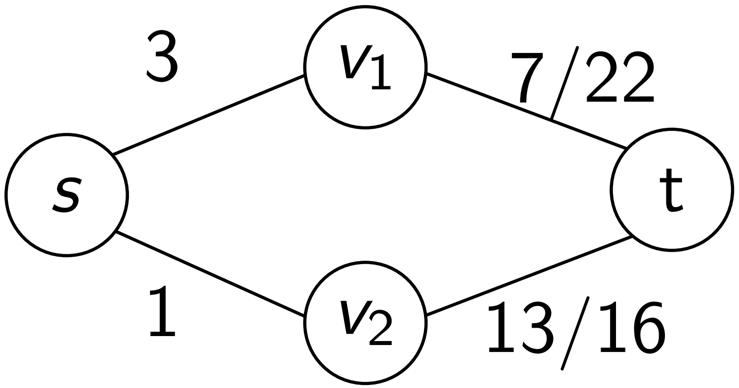
What would you do?



Notation i/j: edge can cost i or j;
both values have probablities of 50%
i and j are positive numbers

Heuristic solution: What could go wrong?

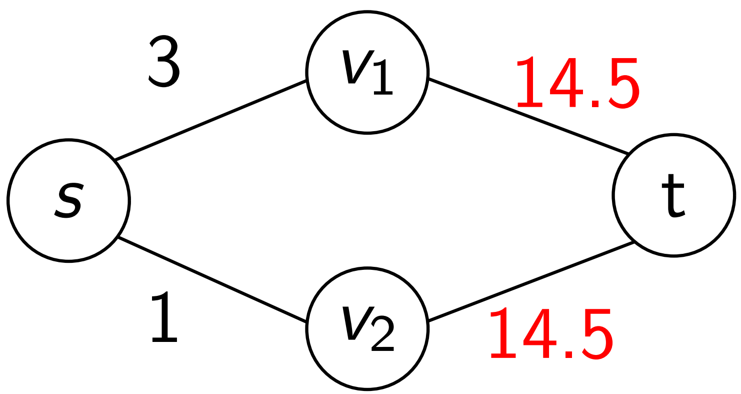
What would a heuristic solution look like?



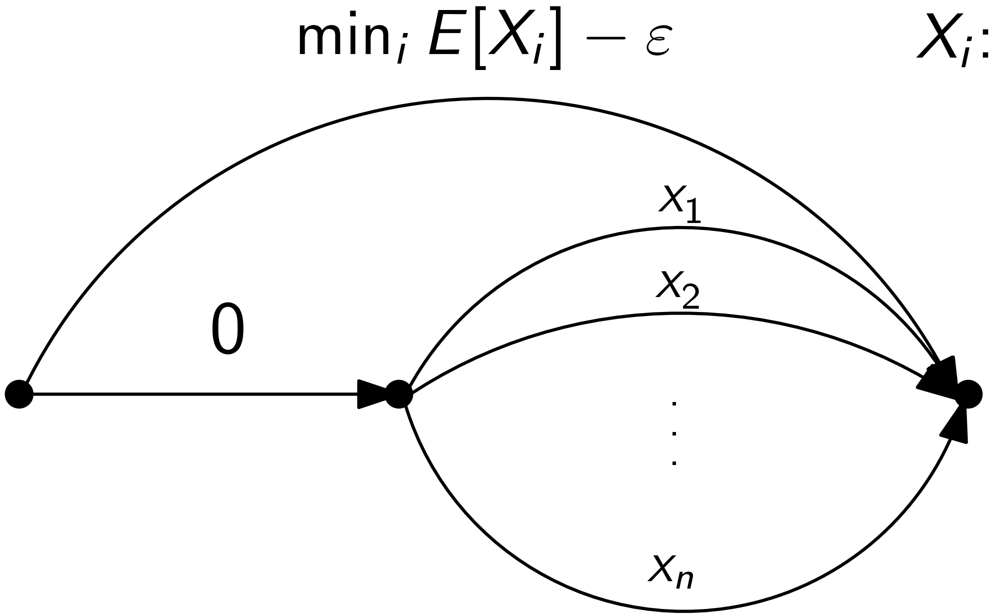
Heuristic solution: What could go wrong?

What would a heuristic solution look like?

- minimum expected distance: replaces unknown edge costs by their expectation

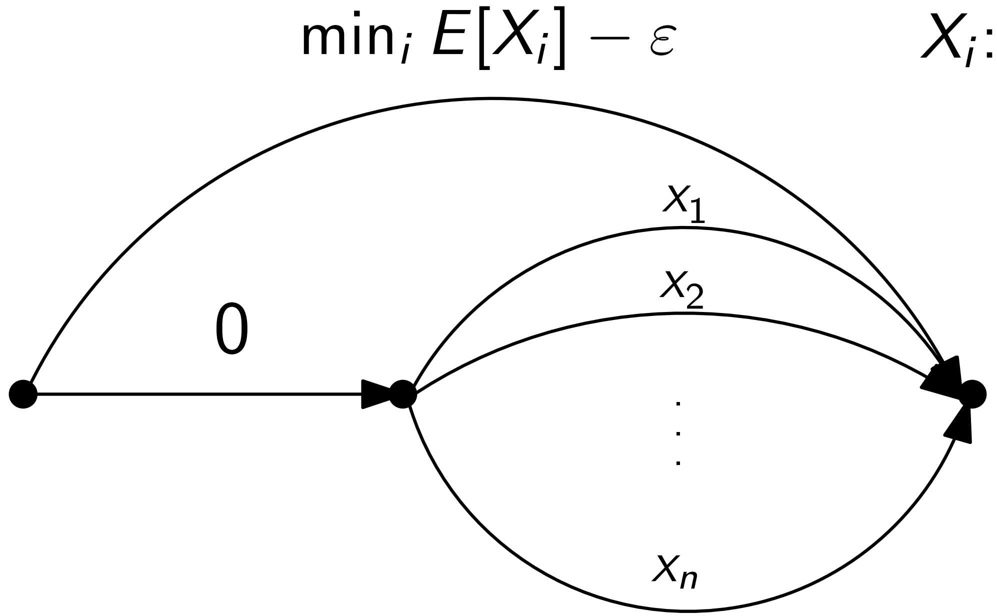


Heuristics: Minimum expected distance



X_i: random variables for edge costs
$\varepsilon > 0$

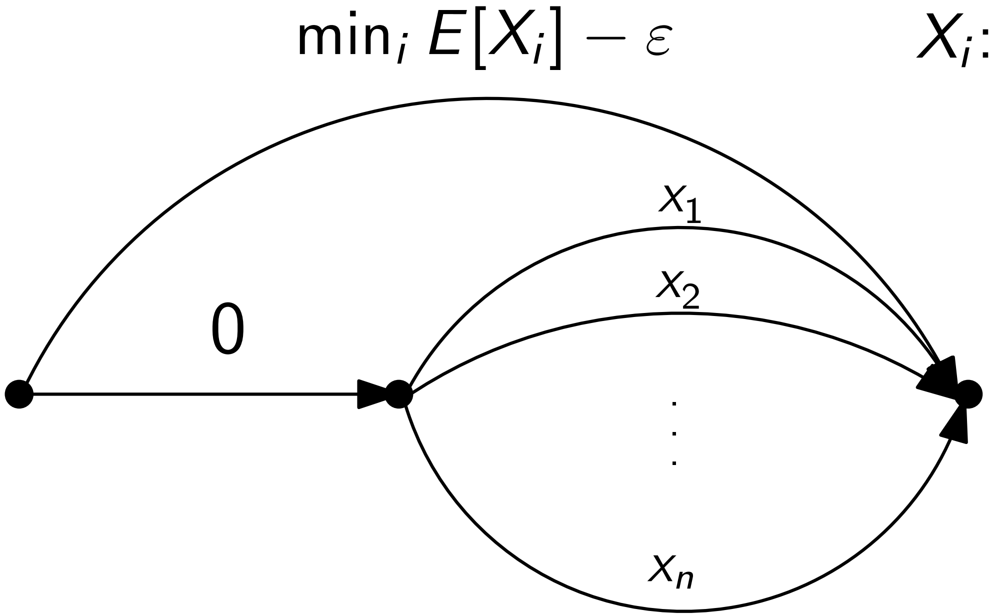
Heuristics: Minimum expected distance



X_i: random variables for edge costs
$\varepsilon > 0$

Always takes the top route
$\Rightarrow$ suboptimal if the costs of all X_i are 1 with probability $p > 0$ and 0 otherwise:

Heuristics: Minimum expected distance



X_i: random variables for edge costs
$\varepsilon > 0$

Always takes the top route
$\Rightarrow$ suboptimal if the costs of all X_i are 1 with probability $p > 0$ and 0 otherwise:

$$\lim_{\varepsilon \to 0} \frac{\min E[X_i] - \varepsilon}{E[\min X_i]} = \lim_{\varepsilon \to 0} \frac{p - \varepsilon}{p^n} = \frac{p}{p^n} = \frac{1}{p^{n-1}}$$

$\Rightarrow$ exponential gap from the optimum

Heuristics: Expected minimum distance

- $\Omega(\log|V|)$ gap from the optimal policy
- optimal on DAGs (see later)

General approach: Markov Decision Processes

What are MDPs?

- model for sequential decision making
- widely used in reinforcement learning

General approach: Markov Decision Processes

What are MDPs?
- model for sequential decision making
- widely used in reinforcement learning

Components:
- set of states (comprises current location and knowledge)
- set of actions
- probabilities of transitioning from one state to another given an action
- costs/rewards (function of the state)

General approach: Markov Decision Processes

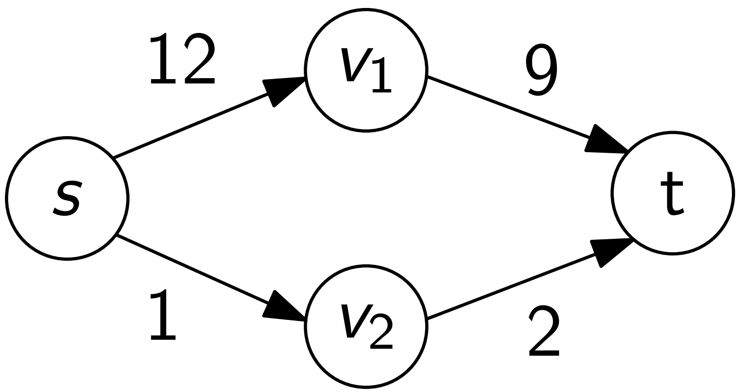
What are MDPs?
- model for sequential decision making
- widely used in reinforcement learning

Components:
- set of states (comprises current location and knowledge)
- set of actions
- probabilities of transitioning from one state to another given an action
- costs/rewards (function of the state)

Why MDPs?
The optimal policy can be found in polynomial time in the size of the MDP (the number of states and actions)

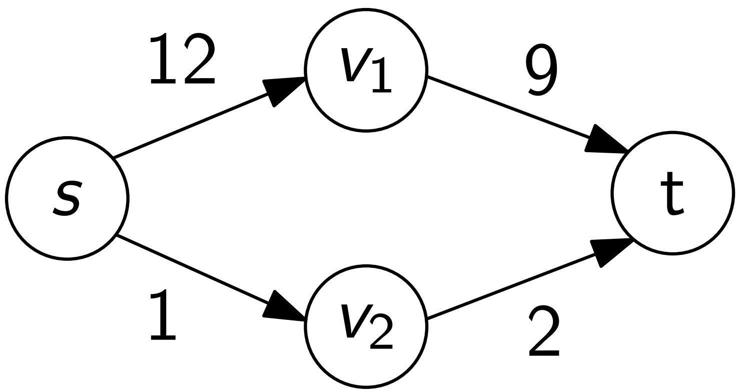
MDPs: Deterministic example

Graph:

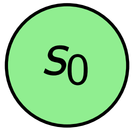


MDPs: Deterministic example

Graph:

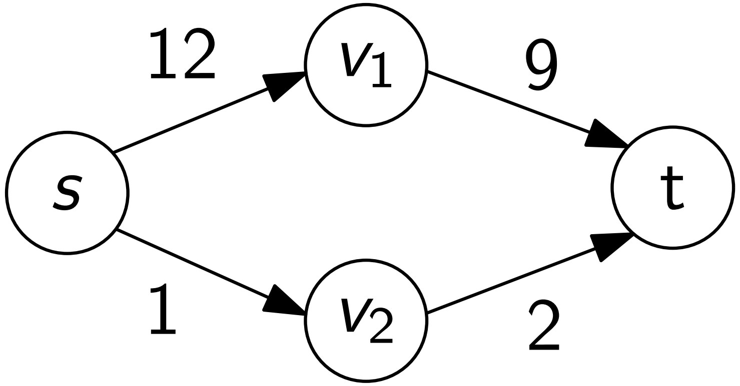


resulting MDP:

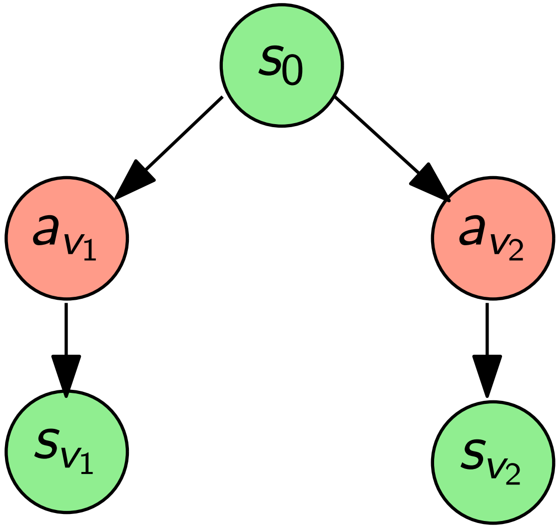


MDPs: Deterministic example

Graph:

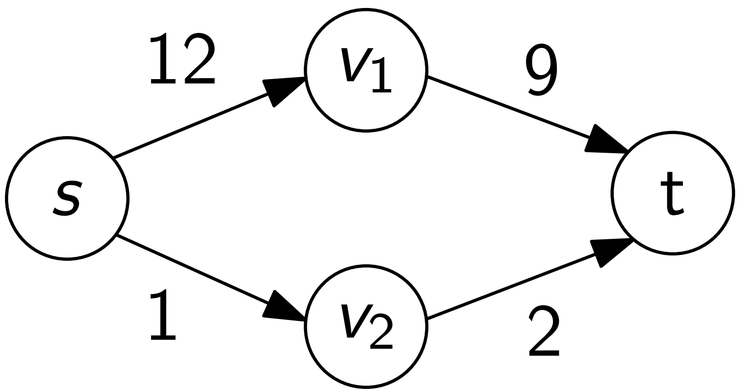


resulting MDP:

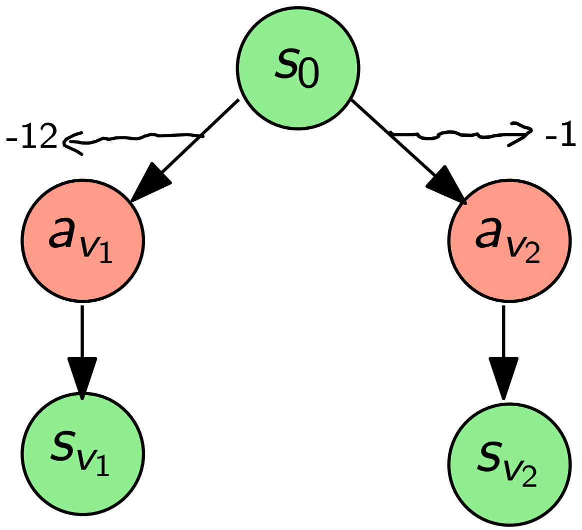


MDPs: Deterministic example

Graph:

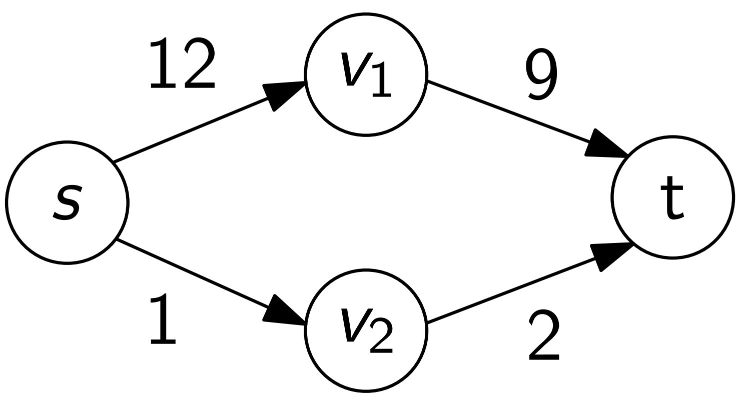


resulting MDP:

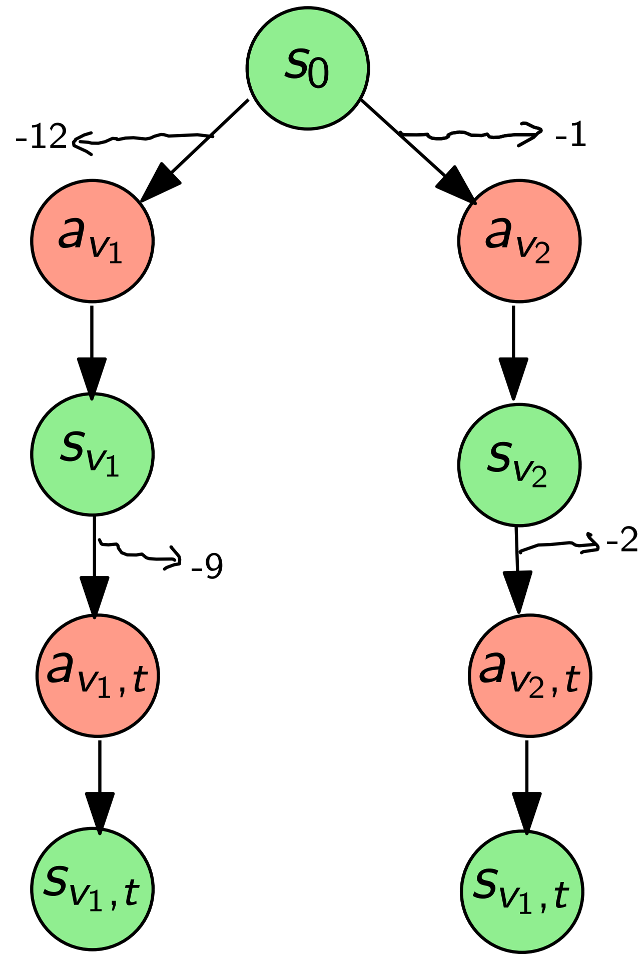


MDPs: Deterministic example

Graph:

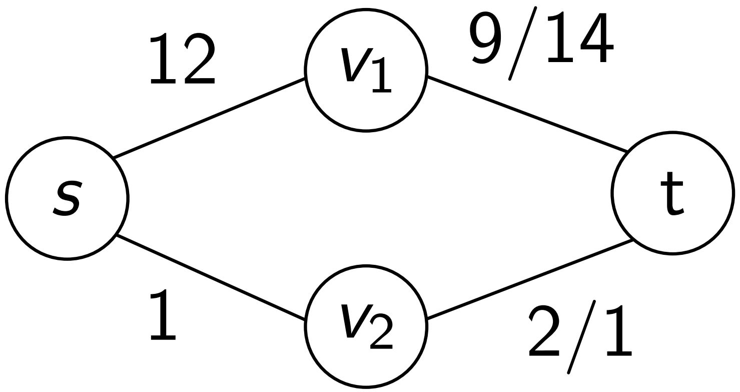


resulting MDP:



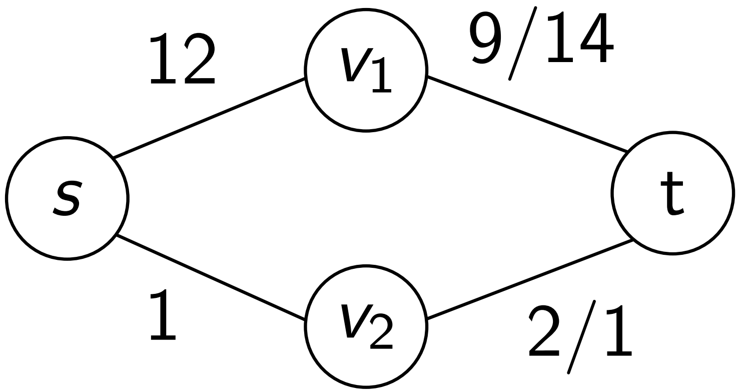
MDPs: Example with uncertainty

Graph:

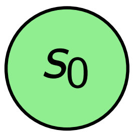


MDPs: Example with uncertainty

Graph:

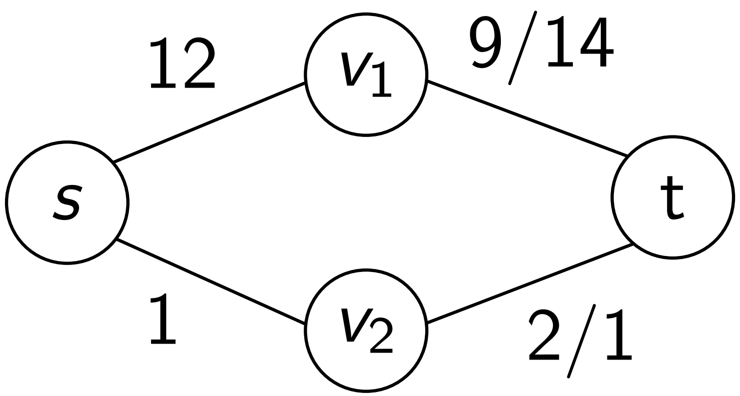


resulting MDP:

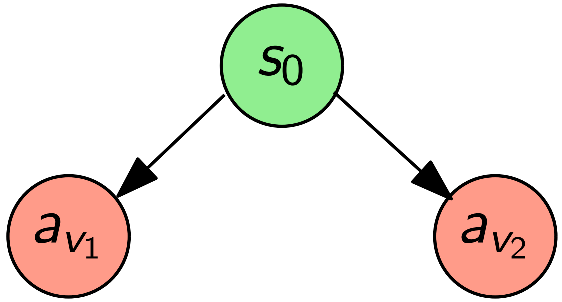


MDPs: Example with uncertainty

Graph:

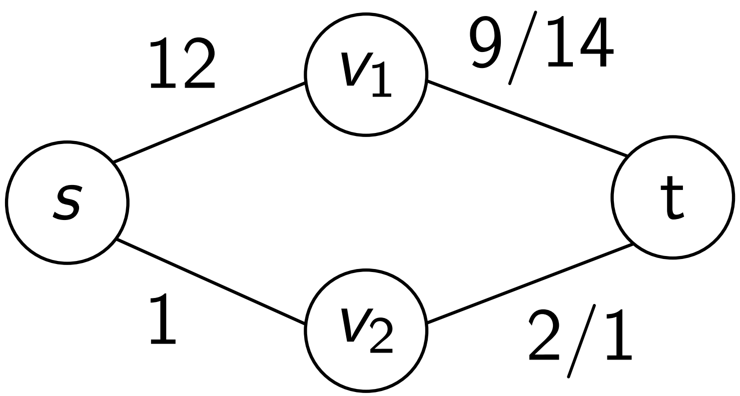


resulting MDP:

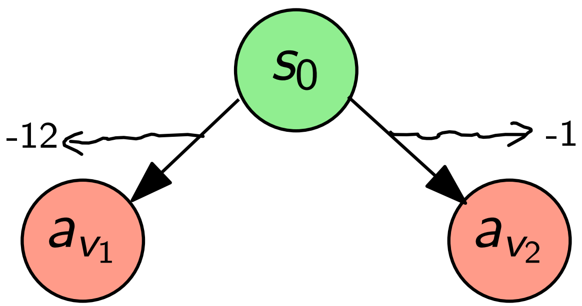


MDPs: Example with uncertainty

Graph:

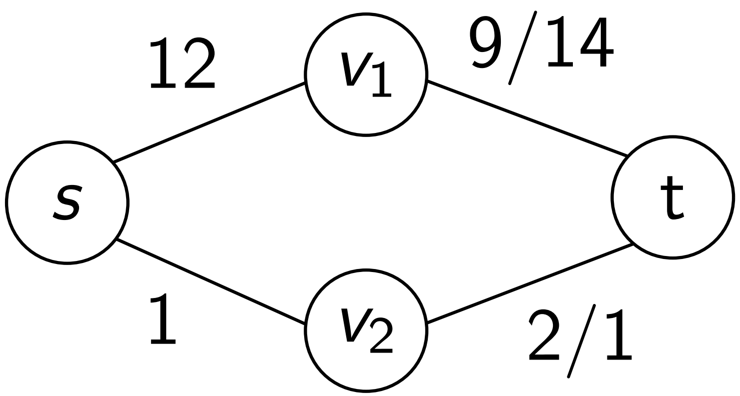


resulting MDP:

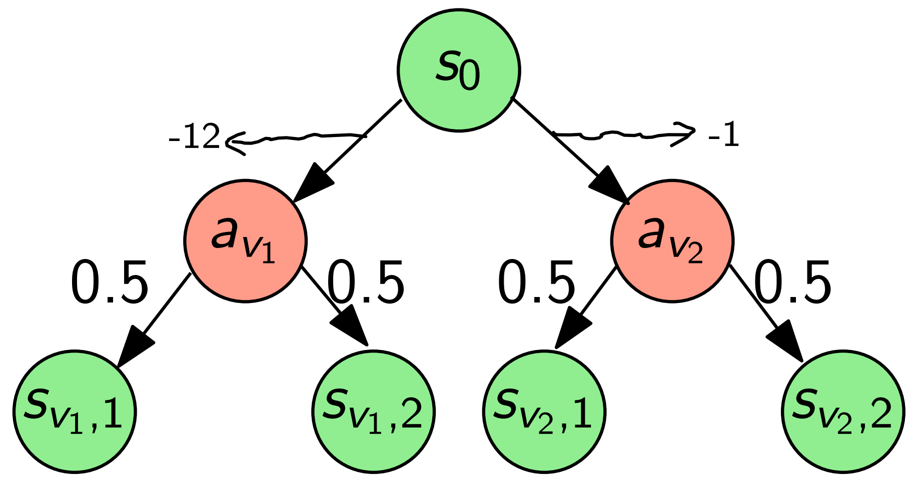


MDPs: Example with uncertainty

Graph:

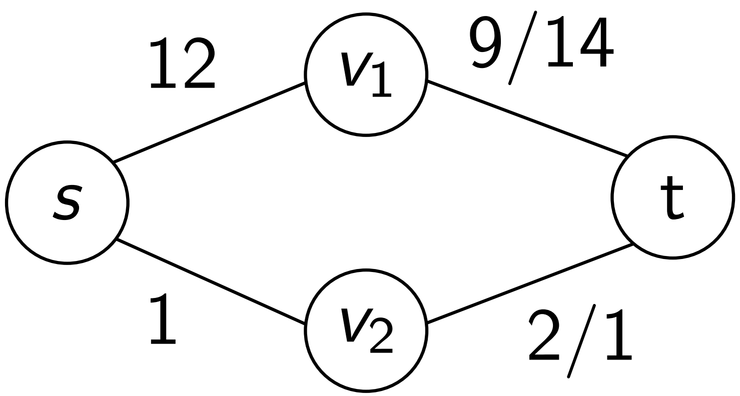


resulting MDP:

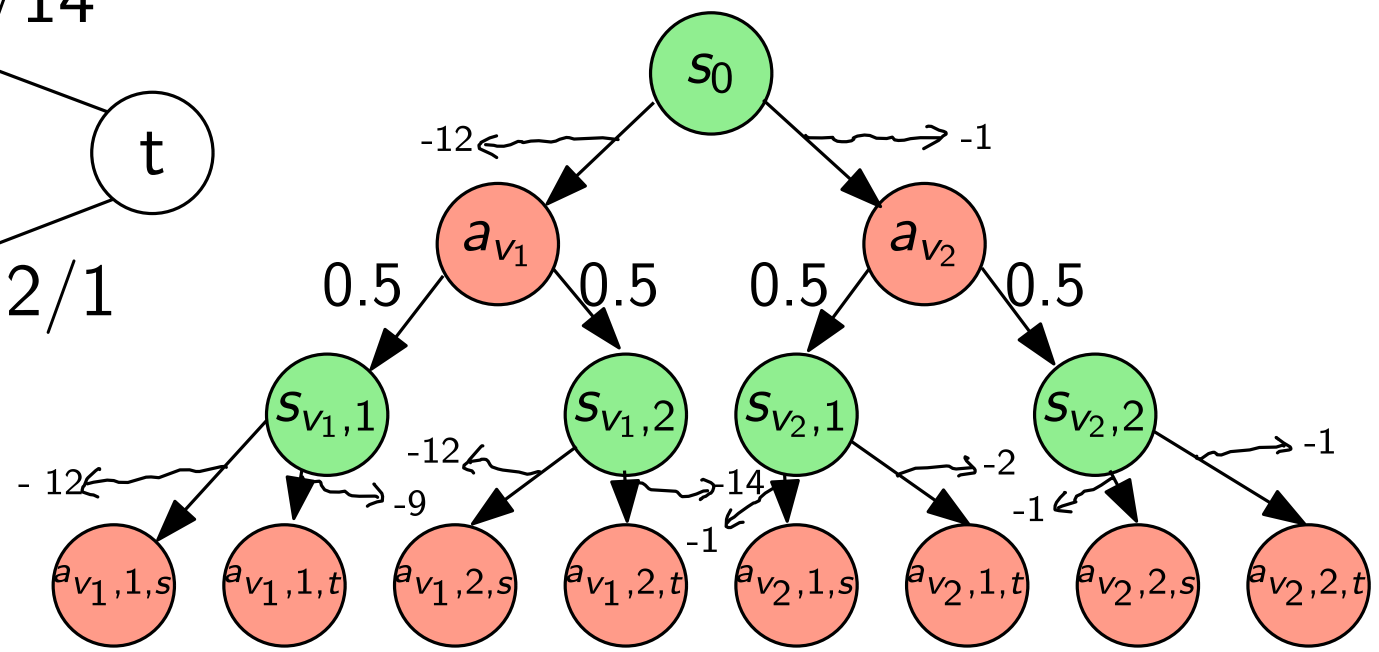


MDPs: Example with uncertainty

Graph:

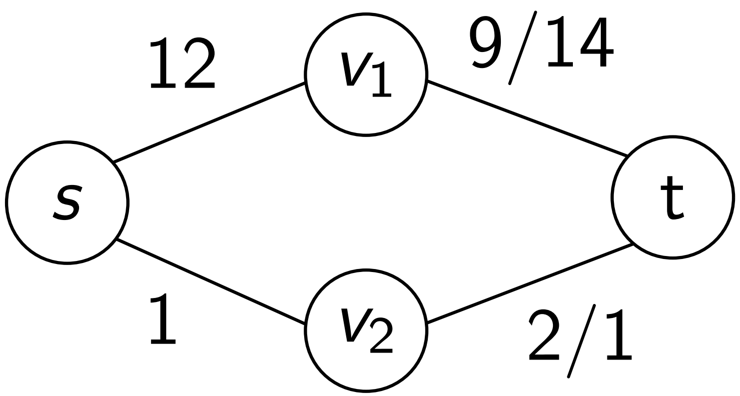


resulting MDP:

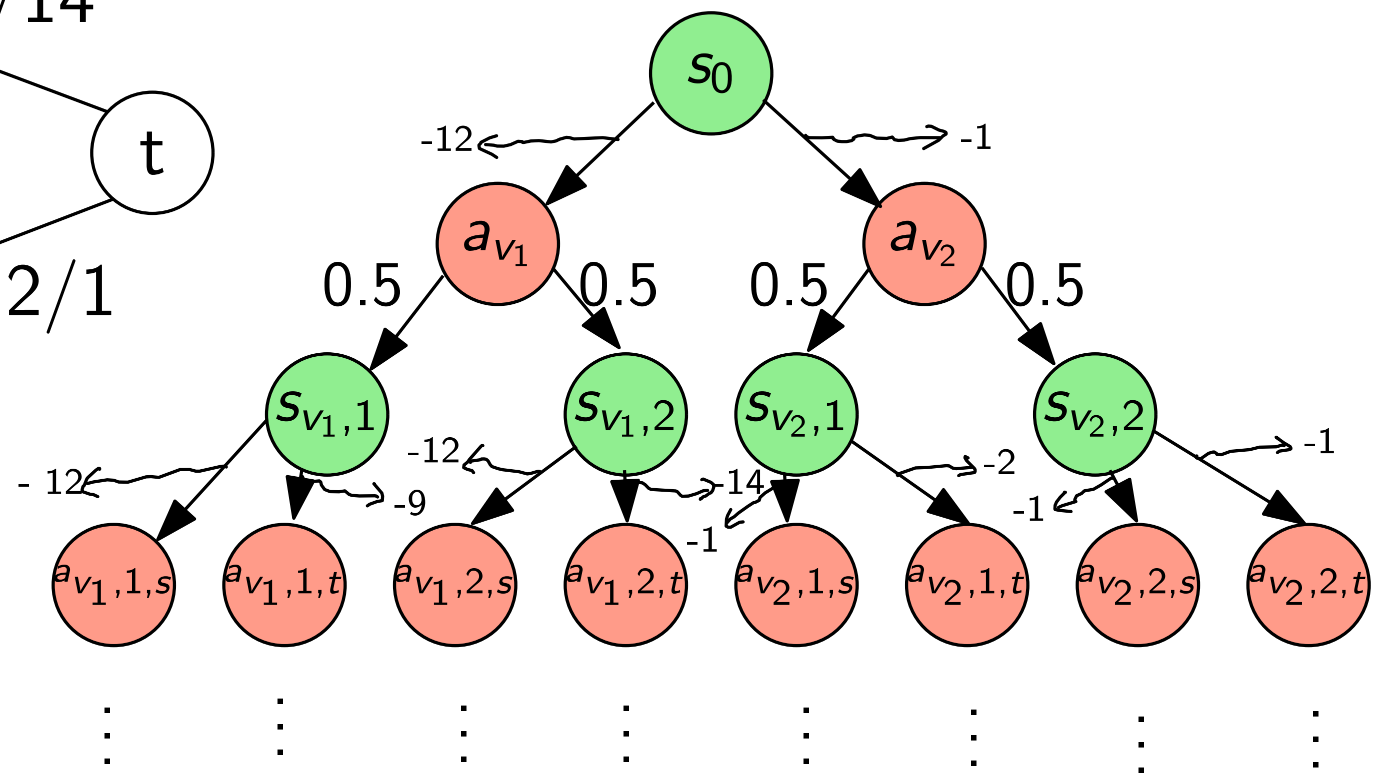


MDPs: Example with uncertainty

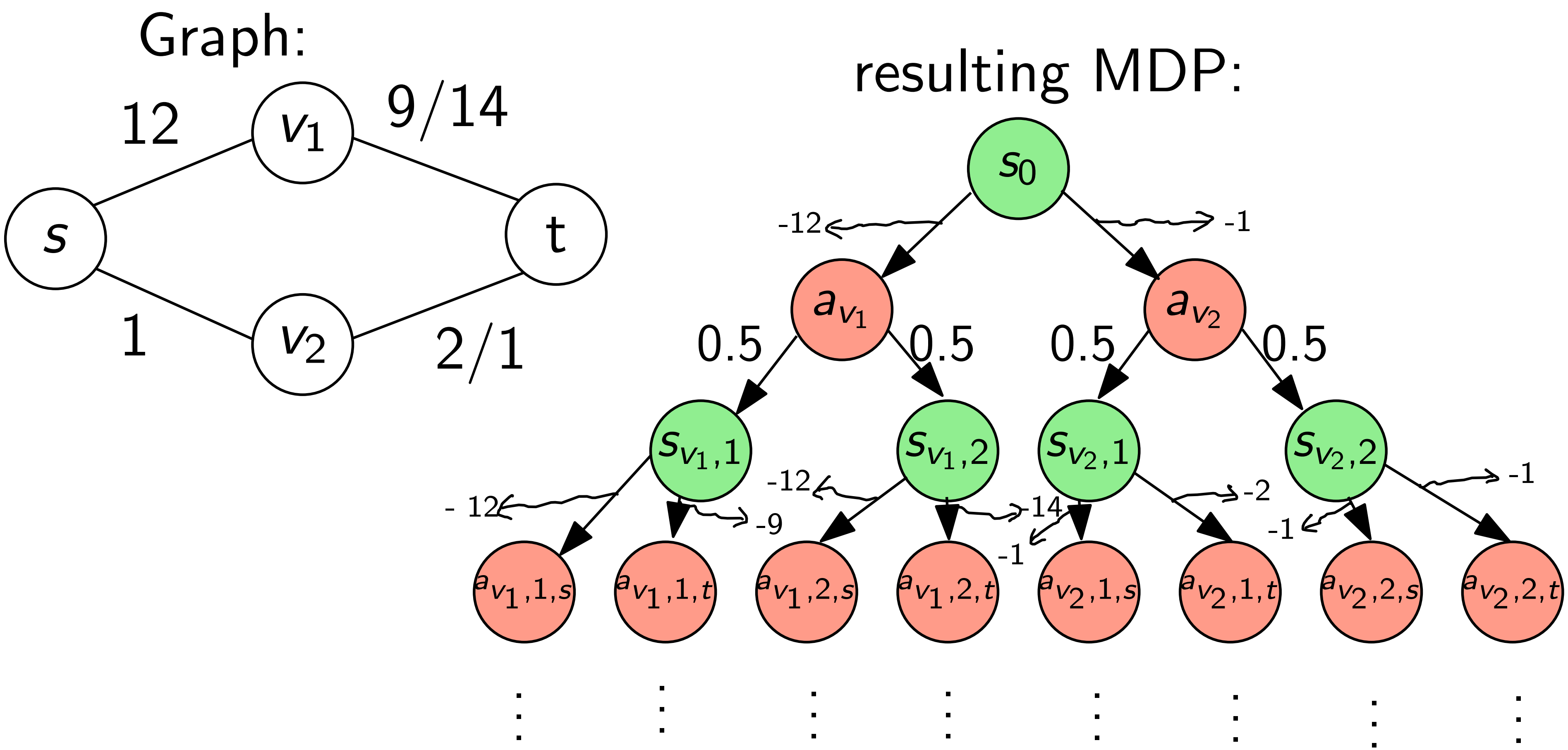
Graph:



resulting MDP:



MDPs: Example with uncertainty



Problem: obvious state space is exponential
in the size of the graph

Problem: Canadian Traveller is #P-hard

#P: class of functions that can be computed by a nondeterministic Turing machine of polynomial time complexity

$\Rightarrow$ optimal policy that minimizes the expected cost of travel can not be found in polynomial time

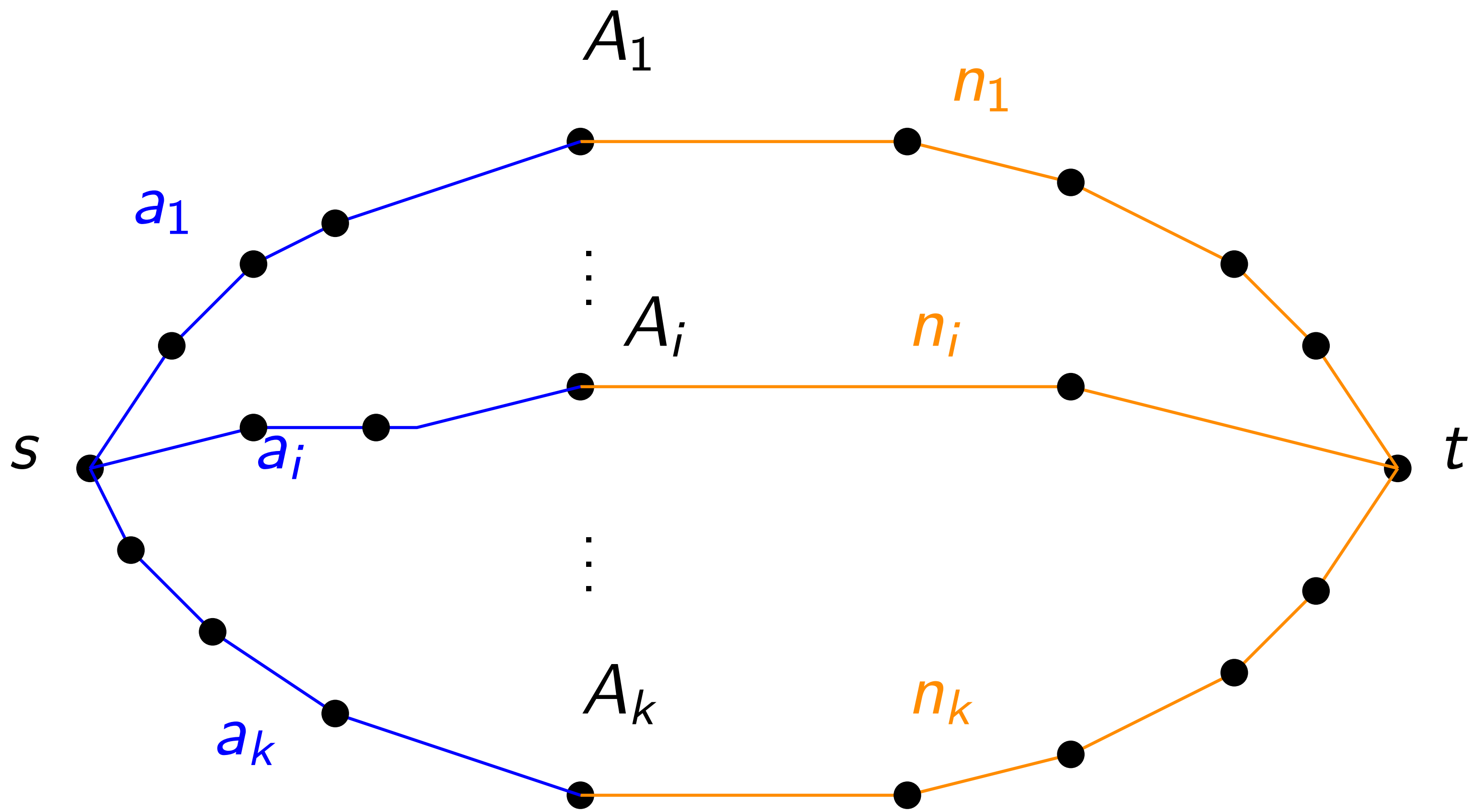
Problem: Canadian Traveller is #P-hard

#P: class of functions that can be computed by a nondeterministic Turing machine of polynomial time complexity

$\Rightarrow$ optimal policy that minimizes the expected cost of travel can not be found in polynomial time

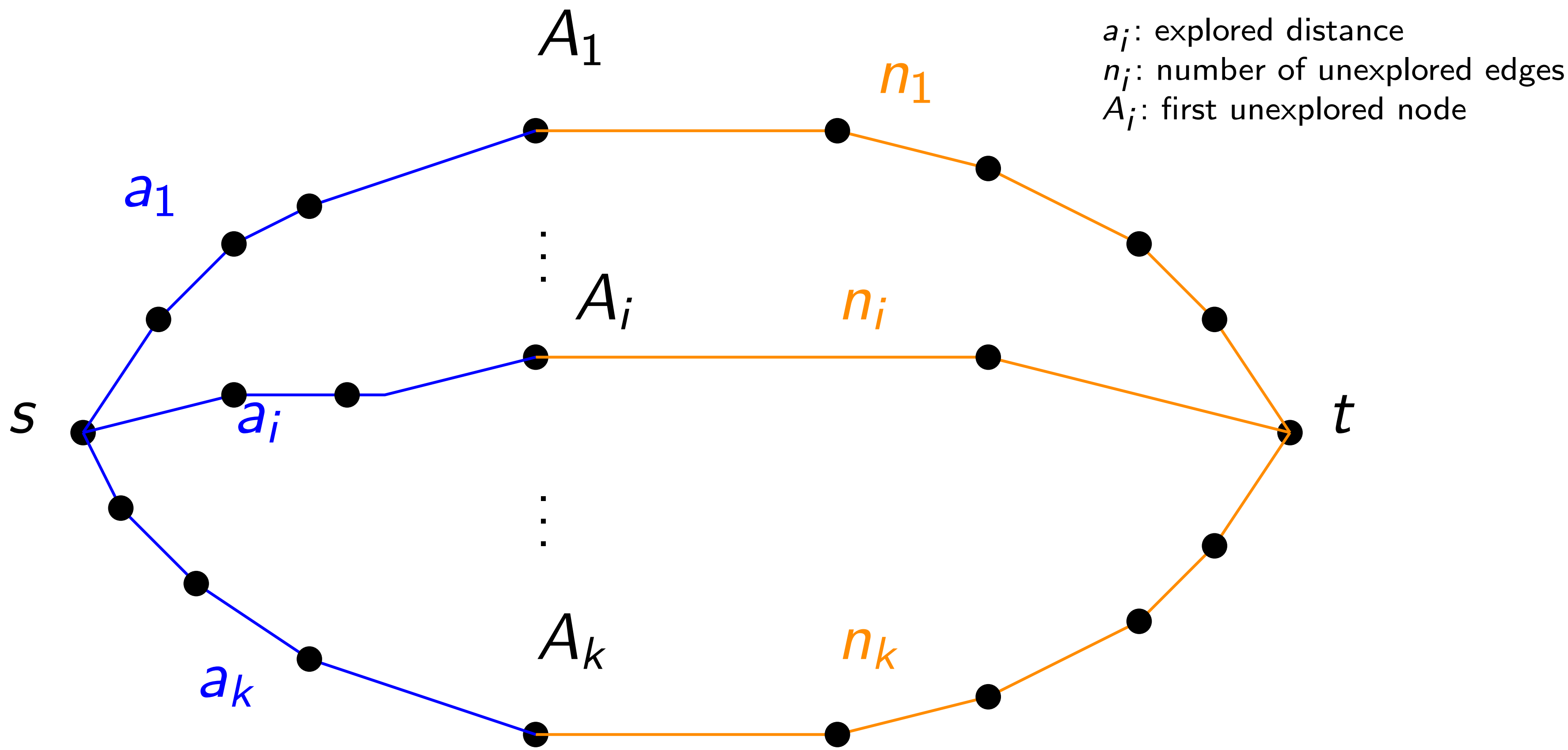
$\Rightarrow$ Focus on *exact* solutions on *special classes of graphs* that can be found in *polynomial time*

Optimal policy for disjoint-path graphs



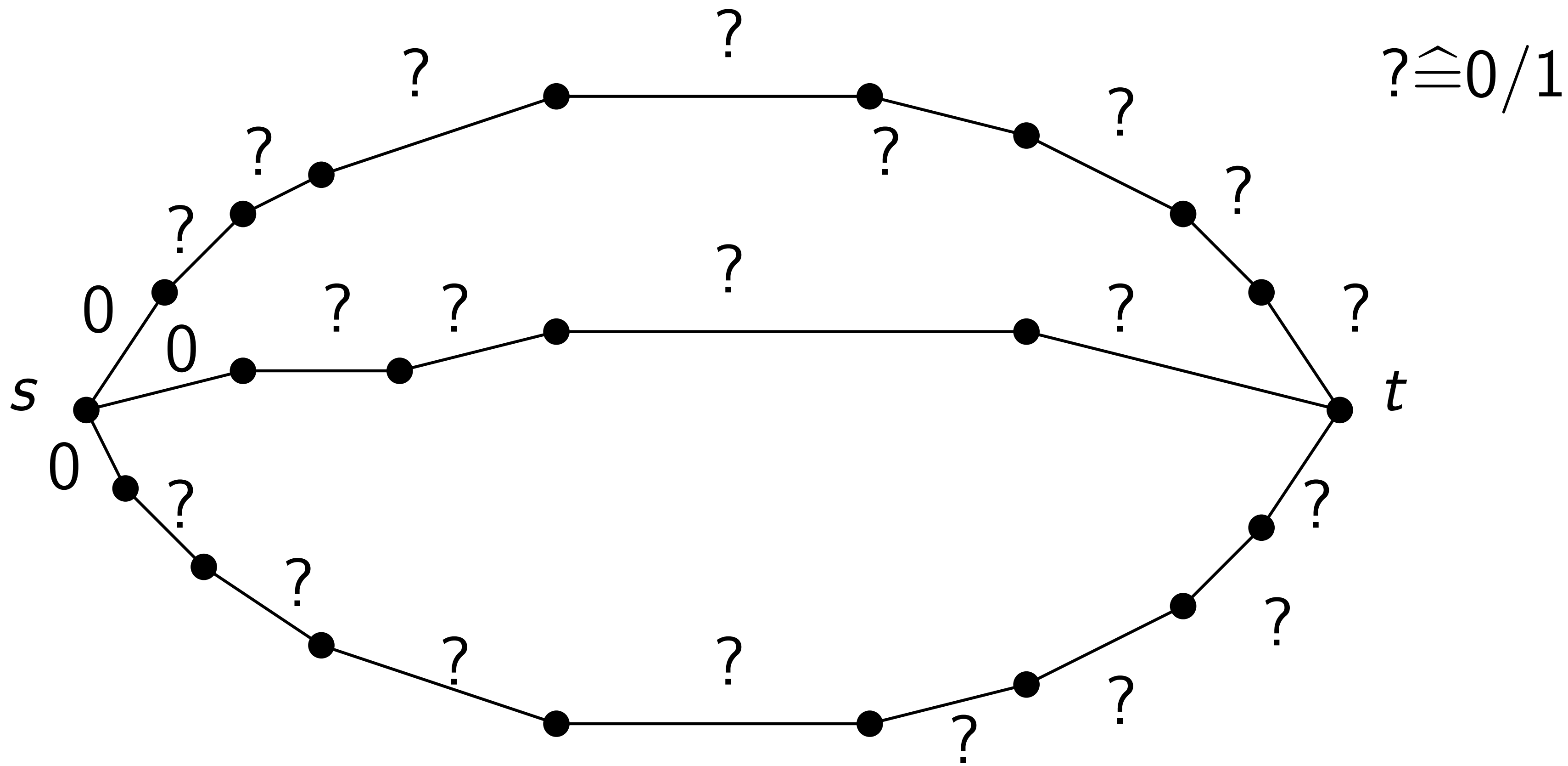
a_i: explored distance on the i-th path
n_i: number of unexplored edges on the i-th path
A_i: first unexplored node on the i-th path

Optimal policy for disjoint-path graphs



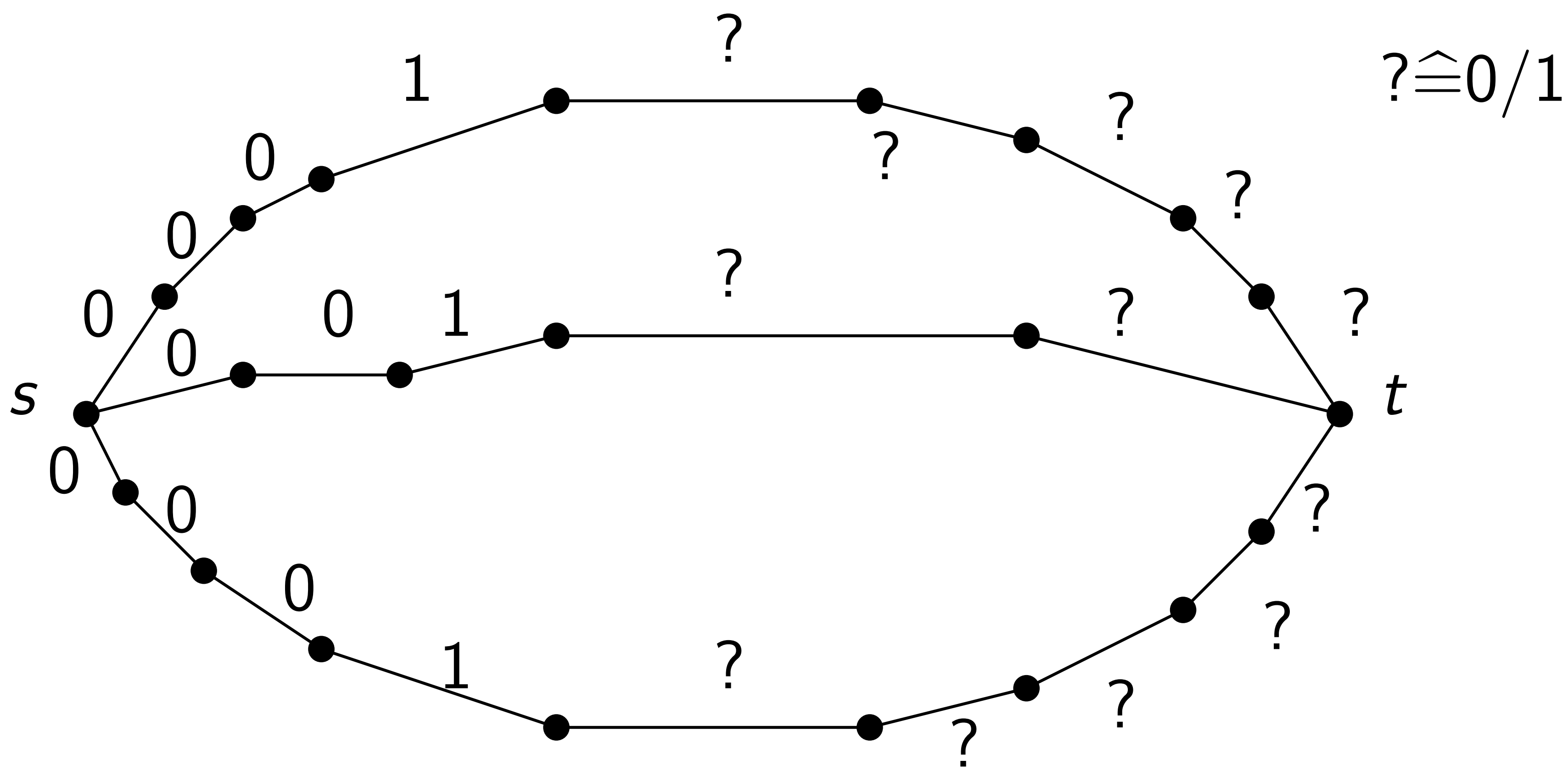
Constraints for edge costs:
- independent and identically distributed
- can only take on two distinct (non-negative) values

Optimal policy for disjoint-path graphs



What would you do (intuitively)?

Optimal policy for disjoint-path graphs

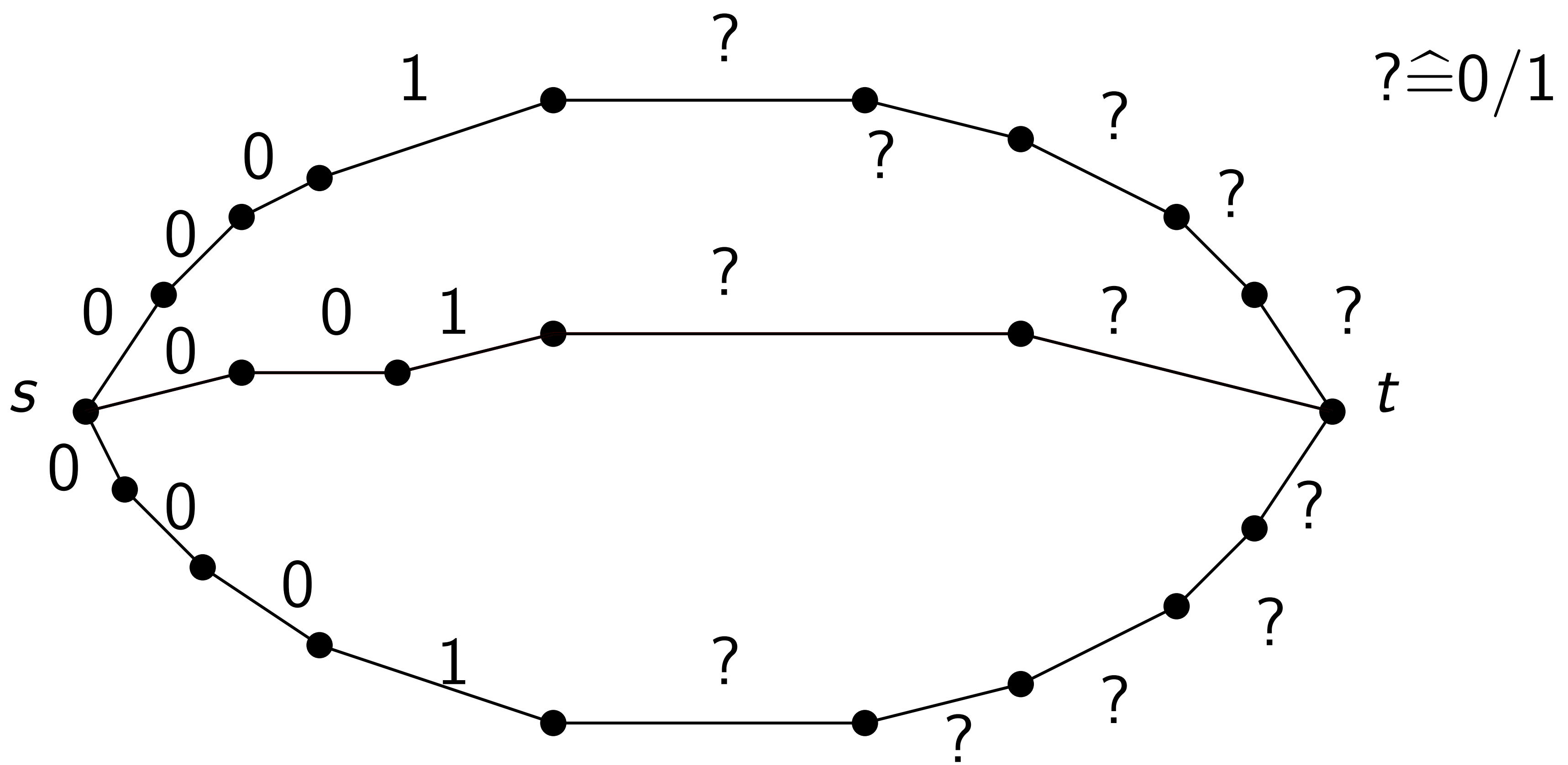


What would you do (intuitively)?

$\Rightarrow$ explore all paths up to the first cost-1 edge

And now?

Optimal policy for disjoint-path graphs



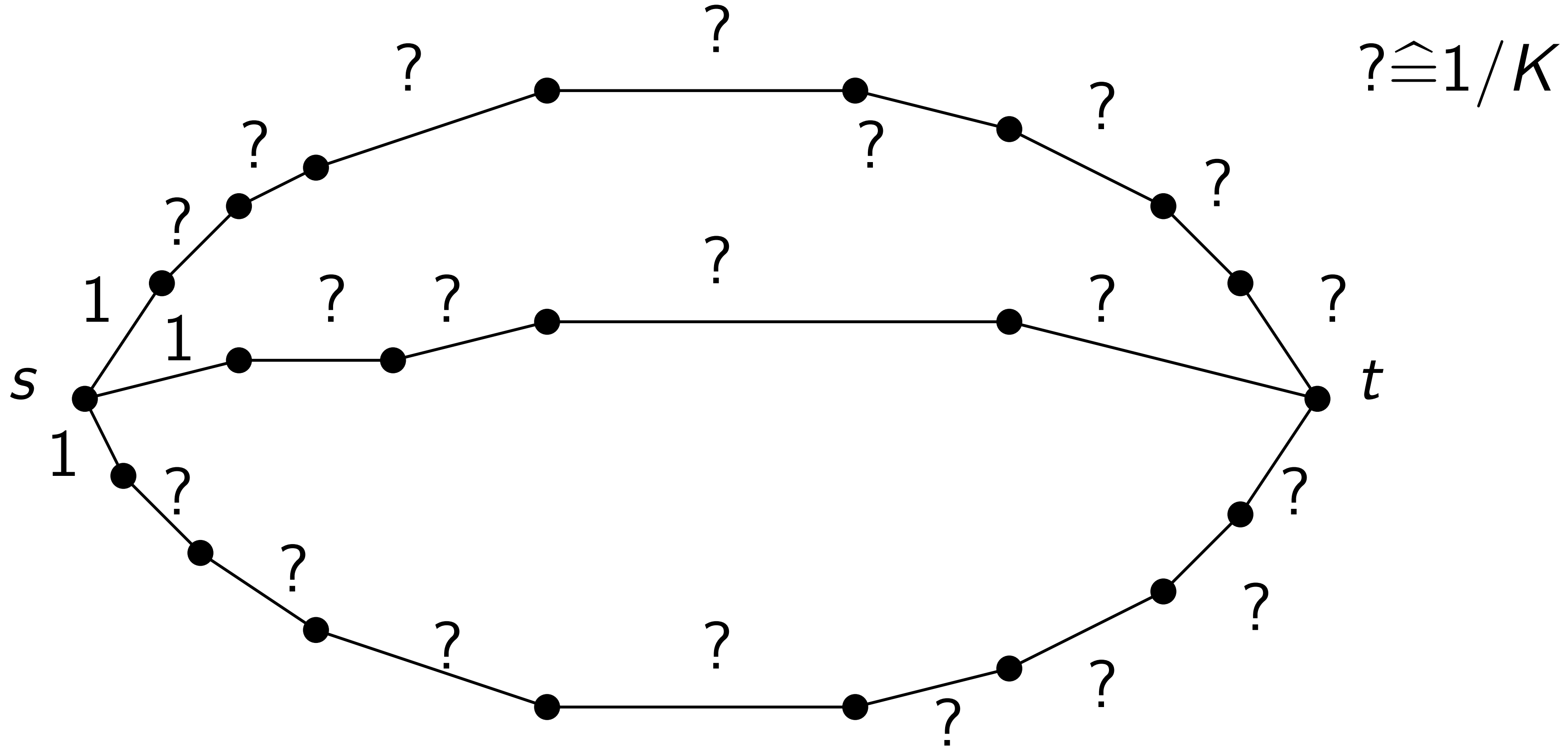
What would you do?

$\Rightarrow$ explore all paths up to the first cost-1 edge

And now?

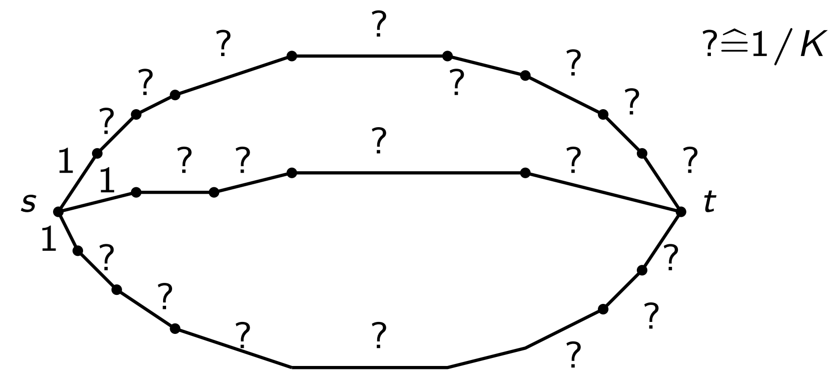
$\Rightarrow$ Select the path with fewest unexplored edges

Optimal policy for disjoint-path graphs



And in this case?

Optimal policy for disjoint-path graphs



And in this case?

$\Rightarrow$ properties of the optimal policy:

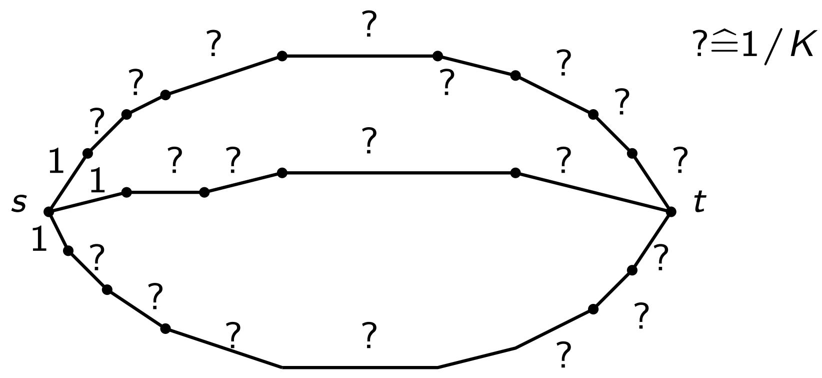
- once we have crossed a K-edge, we will never cross it again
- once we have chosen to follow a path we follow it until the first K-edge

Optimal policy for disjoint-path graphs

And in this case?

$\Rightarrow$ properties of the optimal policy:

- once we have crossed a K-edge,
we will never cross it again
- once we have chosen to follow a path
we follow it until the first K-edge

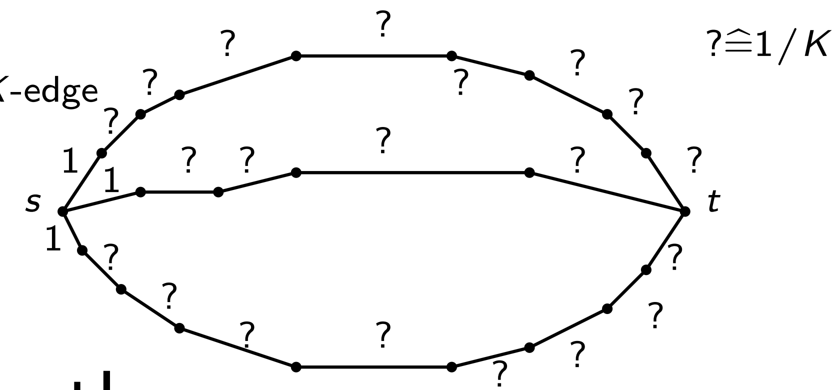


$\Rightarrow$ still needed:

policy for how to explore the paths
(what order, how many)
$\Rightarrow$ MDP

Optimal policy for disjoint-path graphs

Properties of the optimal policy:
- once we have crossed a K-edge, we will never cross it again
- once we have chosen to follow a path we follow it until the first K-edge



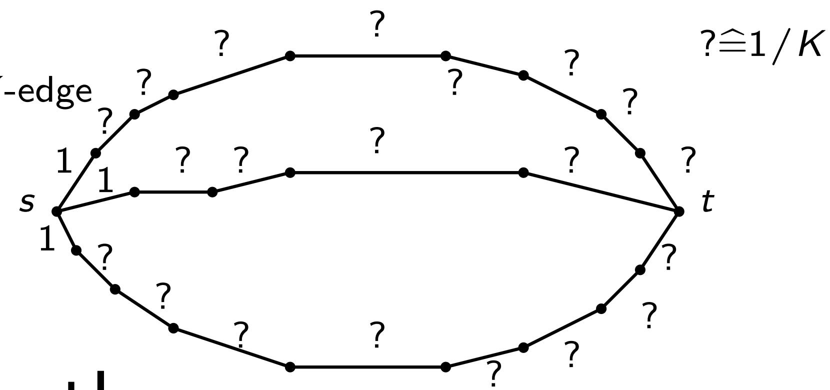
Policy for how to explore two paths:

Current knowledge: $(a_1, x_1, n_1; a_2, x_2, n_2; i)$

a_i: number of observed cost-1 edges;
$x_i = 1$ or K: Last observation;
n_i: number of unobserved edges remaining;
$i = 1, 2$: index of the current path

Optimal policy for disjoint-path graphs

<u>Properties of the optimal policy:</u>
- once we have crossed a K-edge, we will never cross it again
- once we have chosen to follow a path we follow it until the first K-edge



Policy for how to explore two paths:

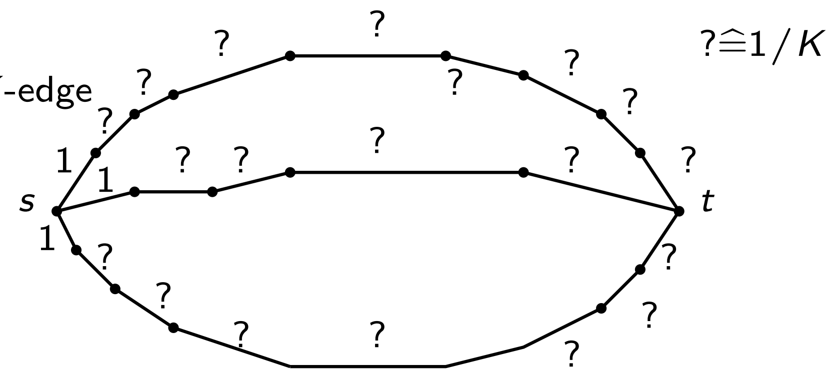
Current knowledge: $(a_1, x_1, n_1; a_2, x_2, n_2; i)$

a_i: number of observed cost-1 edges;
$x_i = 1$ or K: Last observation;
n_i: number of unobserved edges remaining;
$i = 1, 2$: index of the current path

$\Rightarrow$ only $\mathcal{O}(|E|^4)$ many states and only two actions (to continue along the current path or turn back to the other path)

Optimal policy for disjoint-path graphs

Properties of the optimal policy:
- once we have crossed a K-edge, we will never cross it again
- once we have chosen to follow a path we follow it until the first K-edge



Comparison of two paths:
$(a_1, x_1, n_1; a_2, x_2, n_2; i)$

Arbitrary many paths:

- use subroutine for two paths (keep only information about the best explored path so far)
- order of the paths:
  - paths which start with a cost-1 edge in order of increasing length
  - paths which start with a cost-K-edge: keep only the one with the fewest edges

$\Rightarrow$ only $\mathcal{O}(|E|^5)$ many states and only three actions
(to continue along the current path, turn back to the best previously explored path or continue to the next unexplored path)

Optimal policy for DAGs

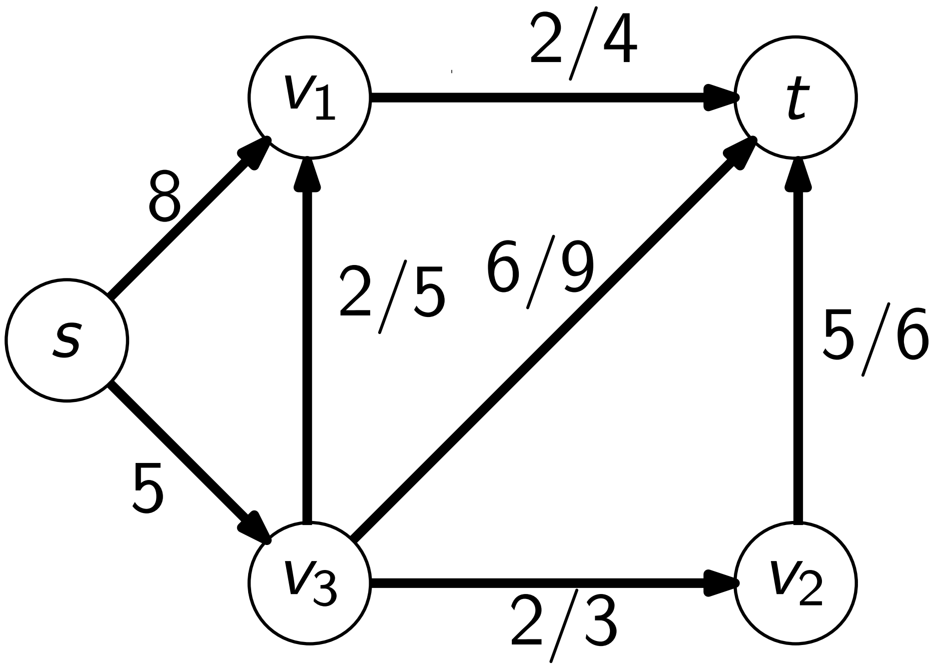
DAG $=$ directed acyclic graph

Problem can be solved in $\mathcal{O}(|E|)$ with dynamic programming

Notations:
- $w(v)$: expected cost of following the optimal policy from node v to t
- $X_{vv'}$: random cost of edge (v, v')

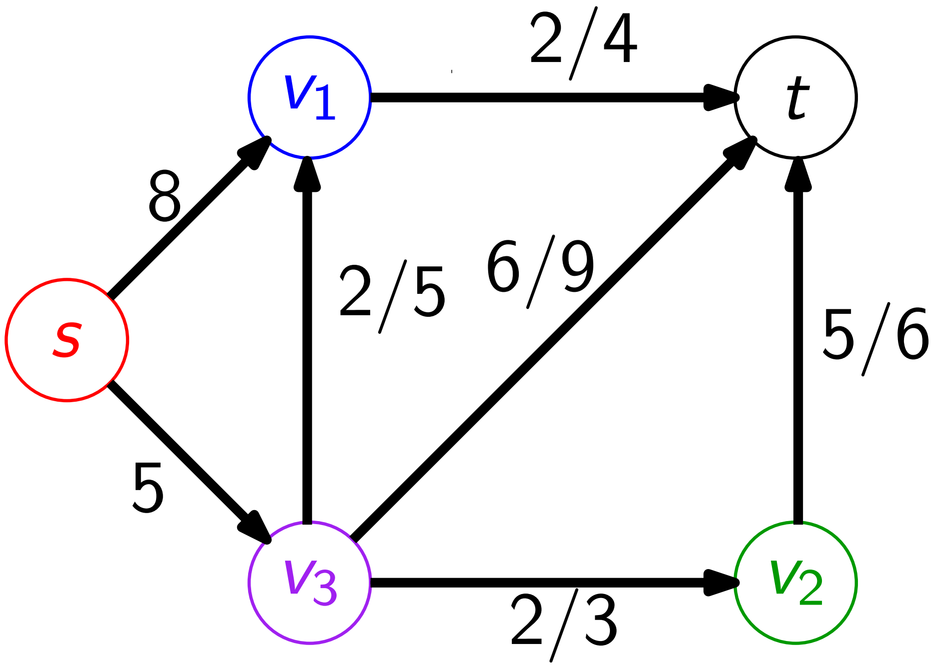
Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



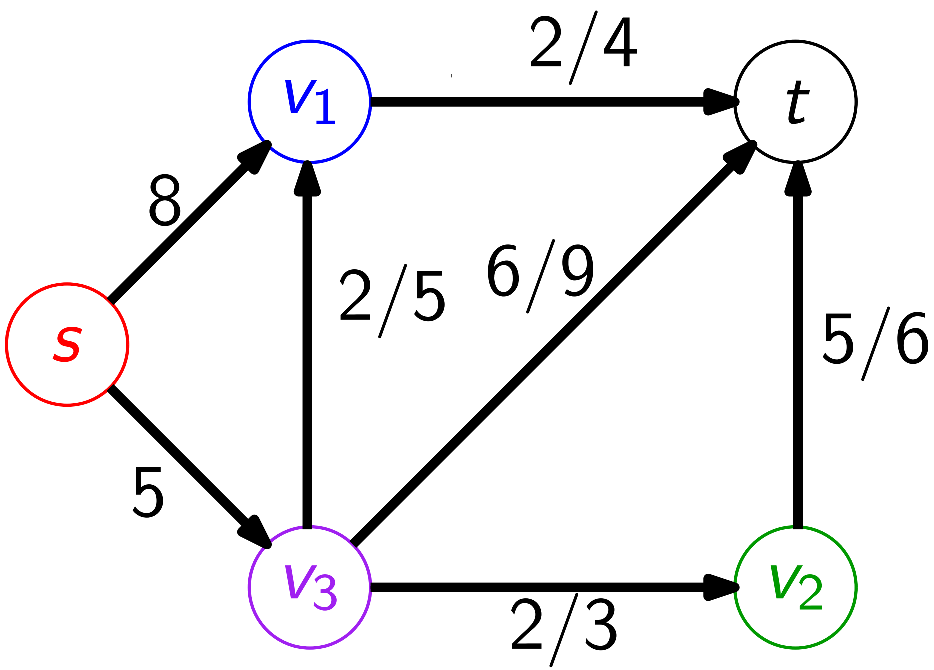
$w(t) = 0$

$w(v_1) = 0.5 \cdot 2 + 0.5 \cdot 4 = 3$

$w(v_2) = 0.5 \cdot 5 + 0.5 \cdot 6 = 5.5$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



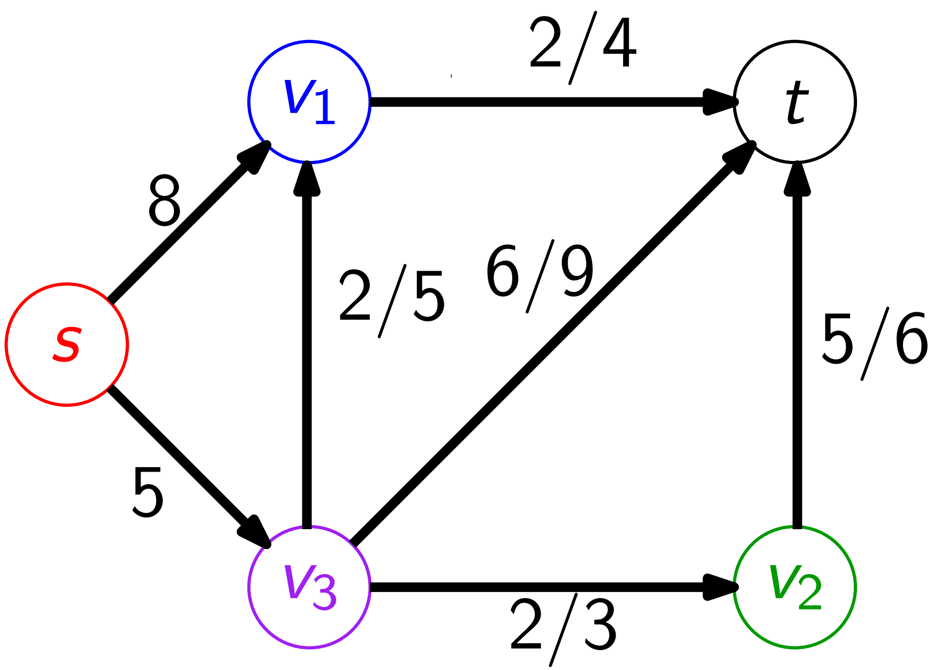
$w(t) = 0$

$w(v_1) = 3$

$w(v_2) = 5.5$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



$w(t) = 0$

$w(v_1) = 3$

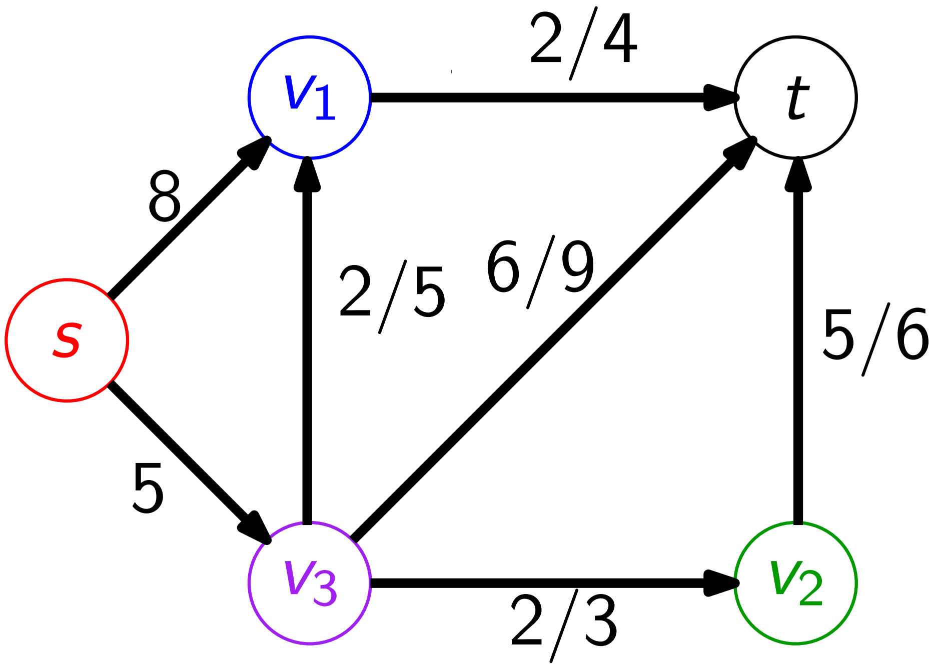
$w(v_2) = 5.5$

Case 1: $c_{v_3,v_1} = 2$; $c_{v_3,t} = 6$, $c_{v_3,v_2} = 2$

$\Rightarrow w_1(v_3) = E[\min_{v'}\{2 + w(v_1), 6 + w(t), 2 + w(v_2)\} = 5$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



$w(t) = 0$

$w(v_1) = 3$

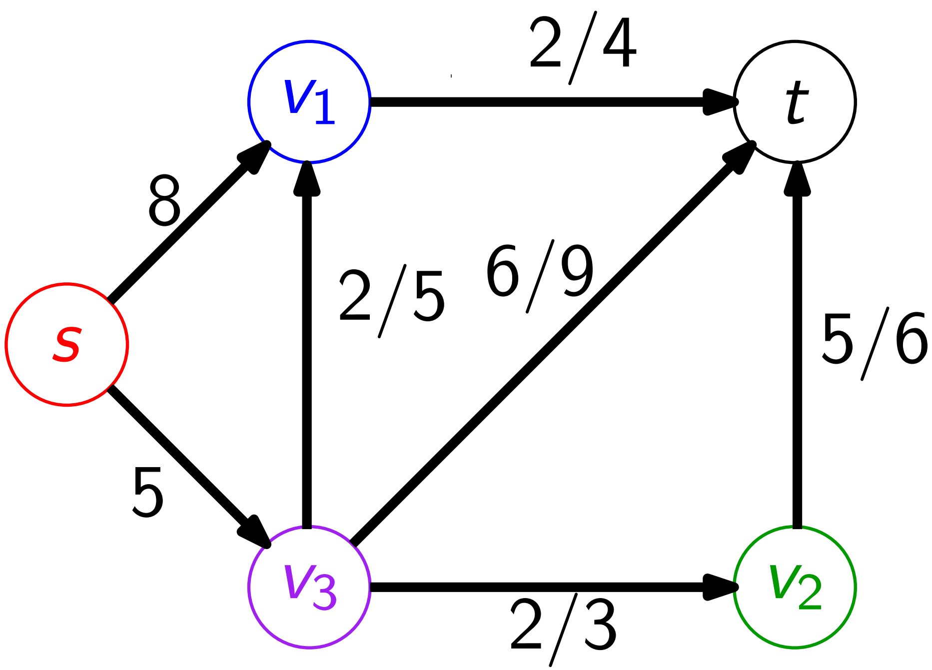
$w(v_2) = 5.5$

Case 2: $c_{v_3,v_1} = 2$; $c_{v_3,t} = 6$, $c_{v_3,v_2} = 3$

$\Rightarrow w_2(v_3) = E[\min_{v'}\{2 + w(v_1), 6 + w(t), 3 + w(v_2)\} = 5$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



$w(t) = 0$

$w(v_1) = 3$

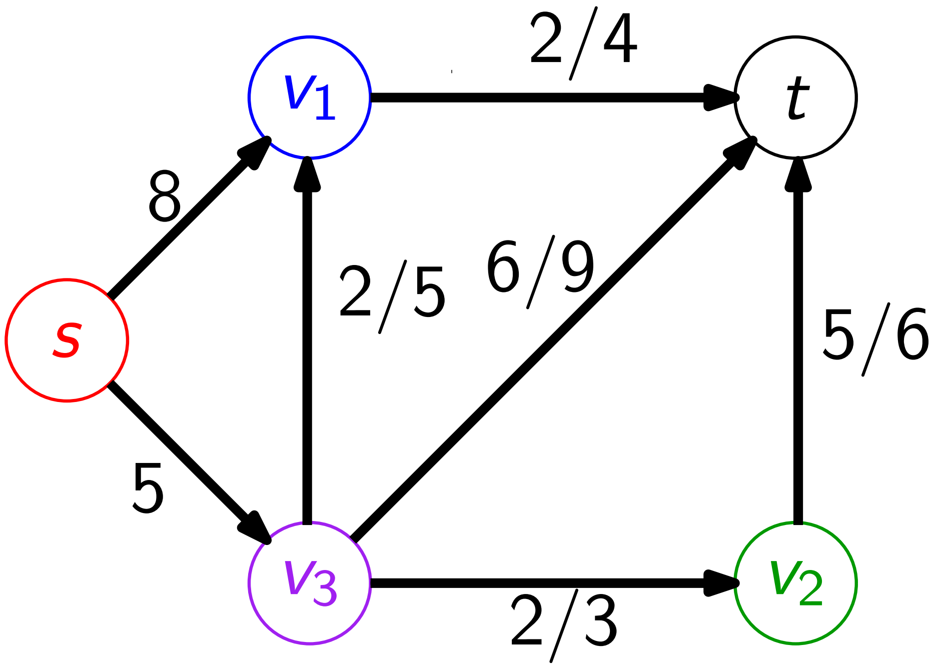
$w(v_2) = 5.5$

Case 5: $c_{v_3,v_1} = 5$; $c_{v_3,t} = 6$, $c_{v_3,v_2} = 2$

$\Rightarrow w_5(v_3) = E[\min_{v'}\{5 + w(v_1), 6 + w(t), 2 + w(v_2)\} = 6$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



$w(t) = 0$

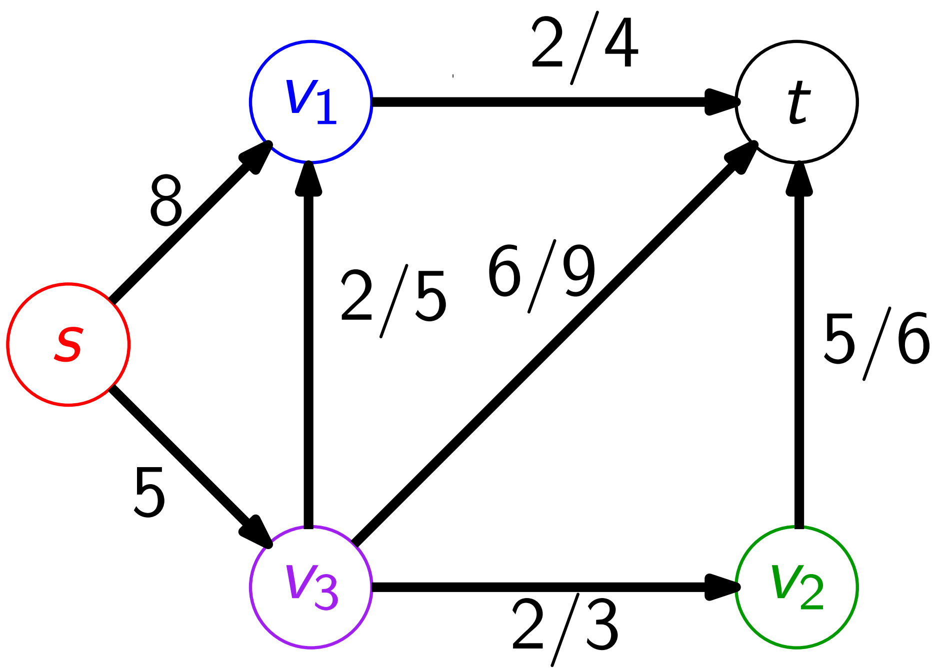
$w(v_1) = 3$

$w(v_2) = 5.5$

$$\Rightarrow w(v_3) = \sum_{i=1}^{8} \frac{1}{2}^3 \cdot w_i(v_3)$$
$$= \frac{1}{8} \cdot (4 \cdot 5 + 2 \cdot 6 + 7,5 + 8) \approx 5,94$$

Optimal policy for DAGs

$$w(v) = E[\min_{v'}\{X_{vv'} + w(v')\}]$$



$w(t) = 0$

$w(v_1) = 3$

$w(v_2) = 5.5$

$w(v_3) \approx 5.94$

Comparison:

$$c(e_{sv_1}) + w(v_1) = 8 + 3 = 11$$
$$c(e_{sv_3}) + w(v_3) = 5 + 5.94 = 10.94$$
$$\Rightarrow \text{edge to } v_3 \text{ should be selected}$$

Recap

Canadian Traveller Problem (according to this paper):

- distributions for the cost values of the edges are given
- upon arriving at a node: actual cost values of incident edges

Problem: cannot be solved in polynomial time

- minimum expected distance heuristic has exponential gap from the optimal policy
- Modelling with MDPs possible, but the obvious state space is exponential in the size of the graph

For special classes of graphs an exact solution can be found efficiently:

- Disjoint-path graphs (with random two-valued edge costs): small MDPs

(- DAGs: dynamic programming)

Sources

This Presentation is based on:

Evdokia Nikolova und David R. Karger: Route planning under uncertainty: the Canadian traveller problem. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2, AAAI'08, Seite 969–974. AAAI Press, 2008, ISBN 9781577353683.

Further sources:

- Andrew G. Barto und Richard S. Sutton: Reinforcement Learning. MIT Press, Cambride, MA, 2. Auflage, 2018.

- David Karger und Evdokia Nikolova: Exact algorithms for the Canadian traveller problem on paths and trees. 2008.

- Christos H. Papadimitriou und Mihalis Yannakakis: Shortest paths without a map. Theoretical Computer Science, 84(1):127–150, 1991, https://doi.org/10.1016/0304-3975(91)90263-2, ISSN 0304-3975. https://www.sciencedirect.com/science/article/pii/ 0304397591902632.

- L.G. Valiant: The complexity of computing the permanent. Theoretical Computer Science, 8(2):189–201, 1979, https://doi.org/10.1016/0304-3975(79)90044-6, ISSN 0304-3975. https://www.sciencedirect.com/science/article/pii/ 0304397579900446.