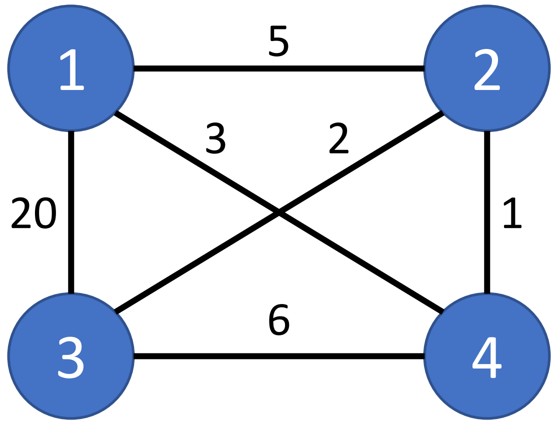
Travelling salesman problem

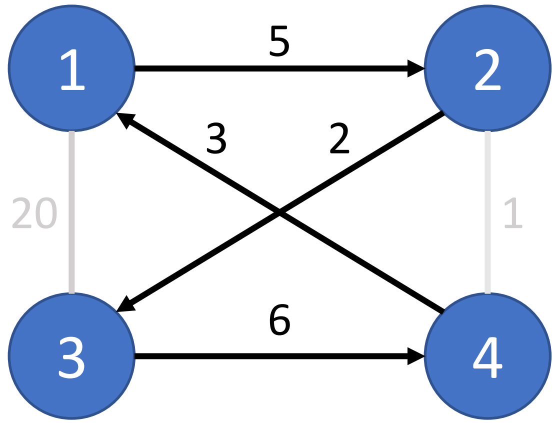
Andrea Mazzolini, Reinforcement learning tutorial, HPC 2020

Statement of the problem

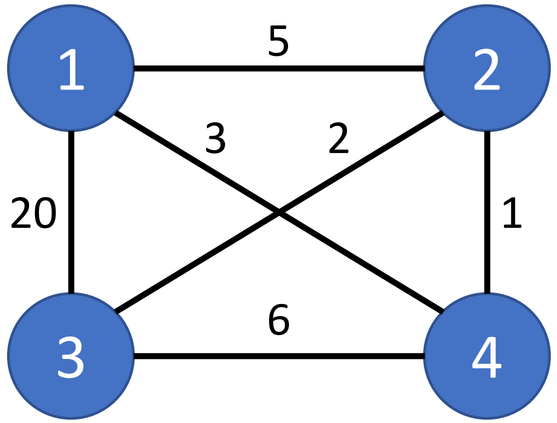
Given N cities and all the distances between them, what is the shortest path that passes by all the cities only once, and then go back to the original city? What is the distance travelled on the shortest path?

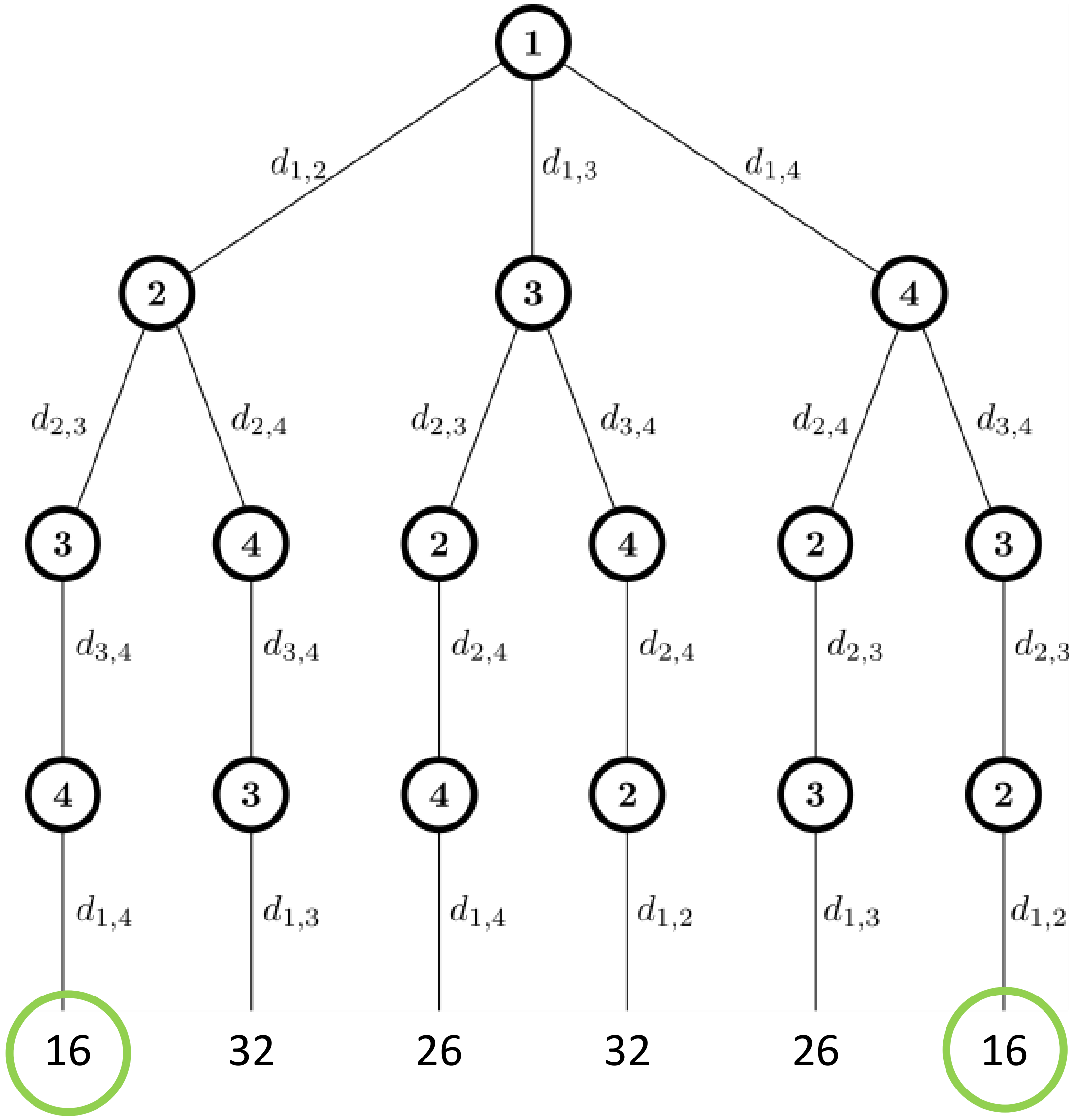


Best path



Brute force solution



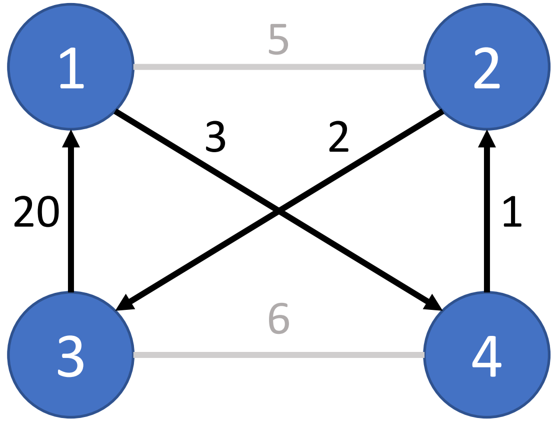


How many paths to enumerate?

Nearest neighbour

- Start from the first city
- Init a set of unvisited cities
- Iterate until unvisited cities is empty:
  - Choose the next city as the nearest one in the unvisited cities
  - Remove the city from the unvisited ones

Not optimal but much faster, $O(n^2)$



As a Markov Decision Process

Optimal solution found in exponential time (better than brute force)

As a Markov Decision Process

Optimal solution found in exponential time (better than brute force)

Current city I'm in

States: $s = (k, A)$

List of cities still to be visited

As a Markov Decision Process

Optimal solution found in exponential time (better than brute force)

Current city I'm in

States: $s = (k, A)$

List of cities still to be visited

Actions: $A_s = A$ Choose one city to be visited

As a Markov Decision Process

Optimal solution found in exponential time (better than brute force)

Current city I'm in

States: $s = (k, A)$

List of cities still to be visited

Actions: $A_s = A$ Choose one city to be visited

Transition probability: $p(s'|a,(k,A)) = \delta(s' - (a, A \backslash \{a\}))$ Deterministic transition to the city that I chose $a \in A_s$

As a Markov Decision Process

Optimal solution found in exponential time (better than brute force)

States: $s = (k, A)$

Current city I'm in

List of cities still to be visited

Actions: $A_s = A$ → Choose one city to be visited

Transition probability: $p(s'|a, (k, A)) = \delta(s' - (a, A\backslash\{a\}))$ → Deterministic transition to the city that I chose $a \in A_s$

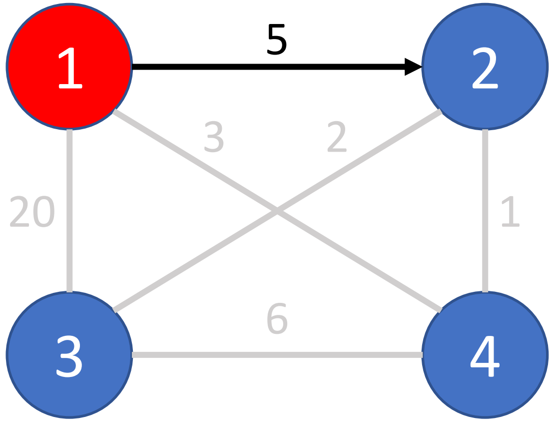
Reward: $r\big((a, A\backslash\{a\}), (k, A)\big) = -d_{k,a}$ → Minus distance from k to a

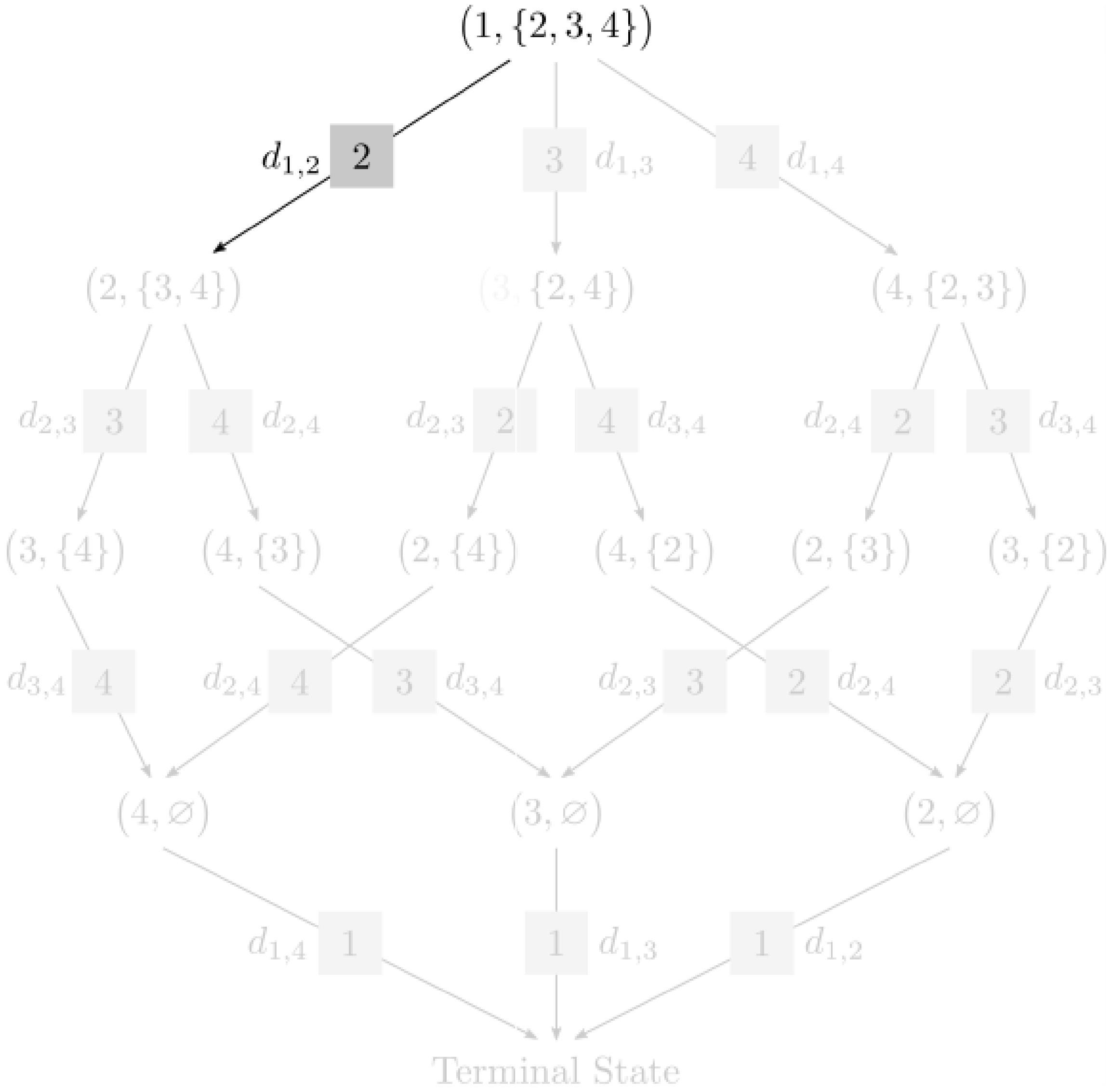
As a Markov Decision Process

Current city I’m in

States: $s = (k, A)$

List of cities still to be visited



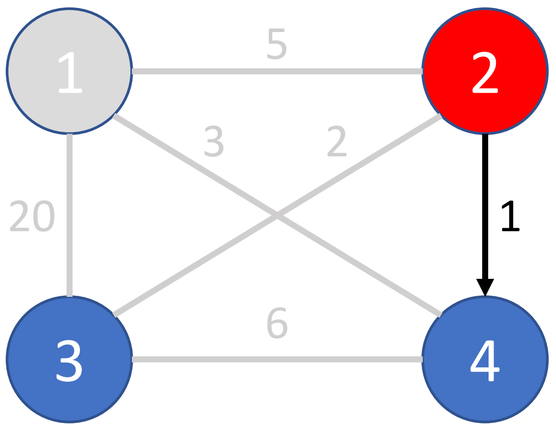


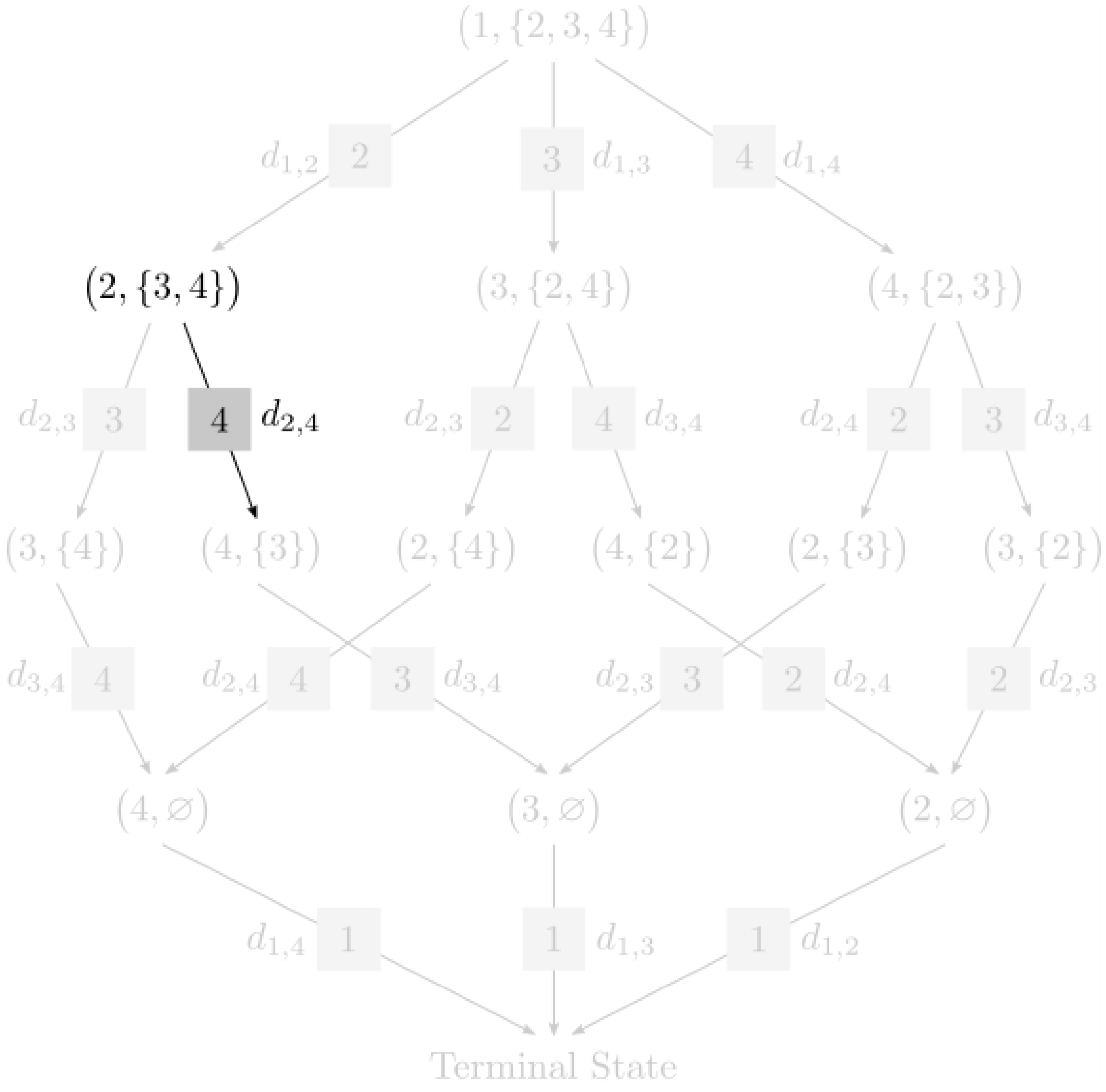
As a Markov Decision Process

States: $s = (k, A)$

Current city I'm in

List of cities still to be visited



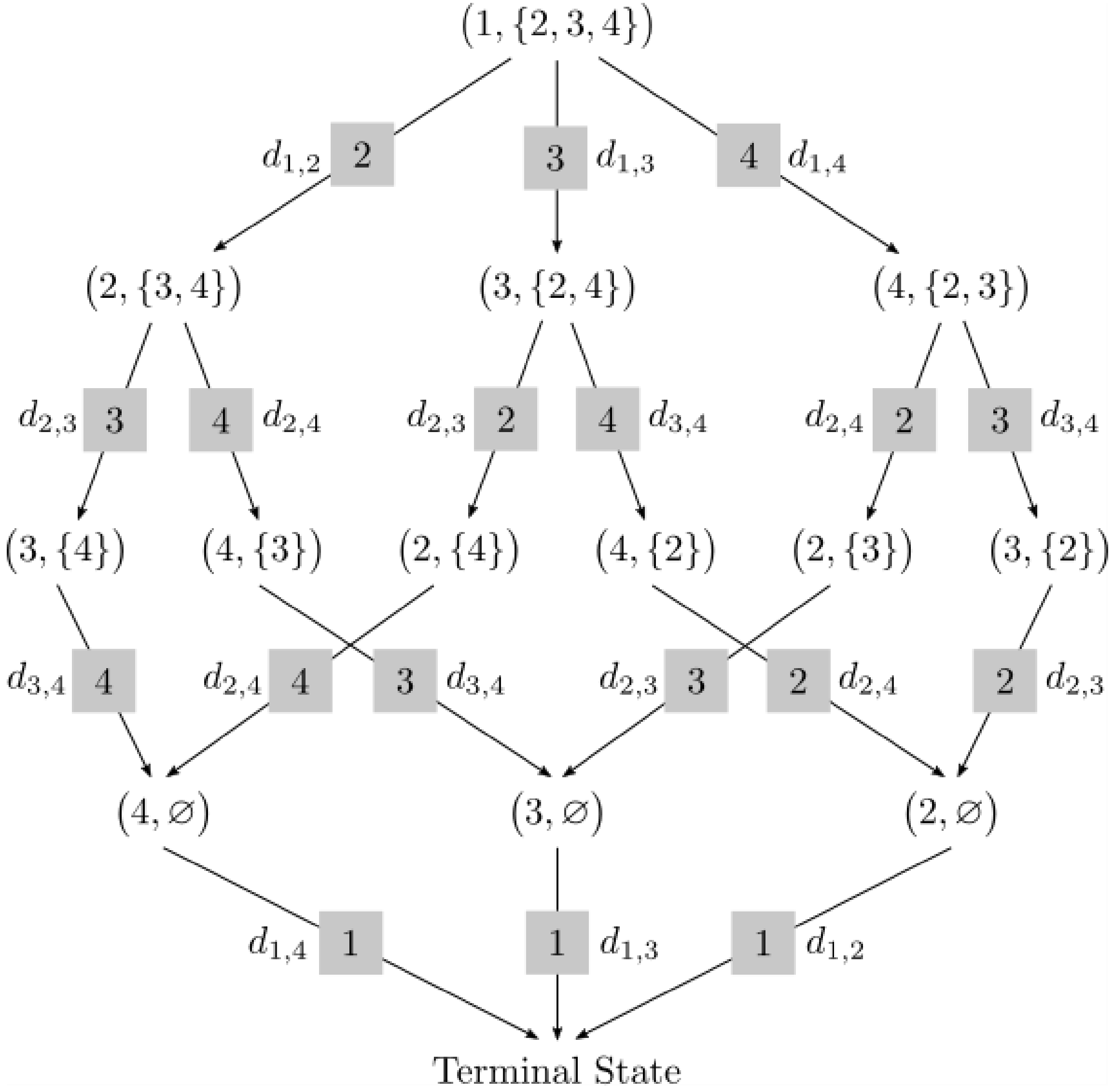


As a Markov Decision Process

States: $s = (k, A)$

Current city I'm in

List of cities still to be visited



Small exercise: diagram of all the transition, how many are them? (cost for memory)

As a Markov Decision Process

States: $s = (k, A)$

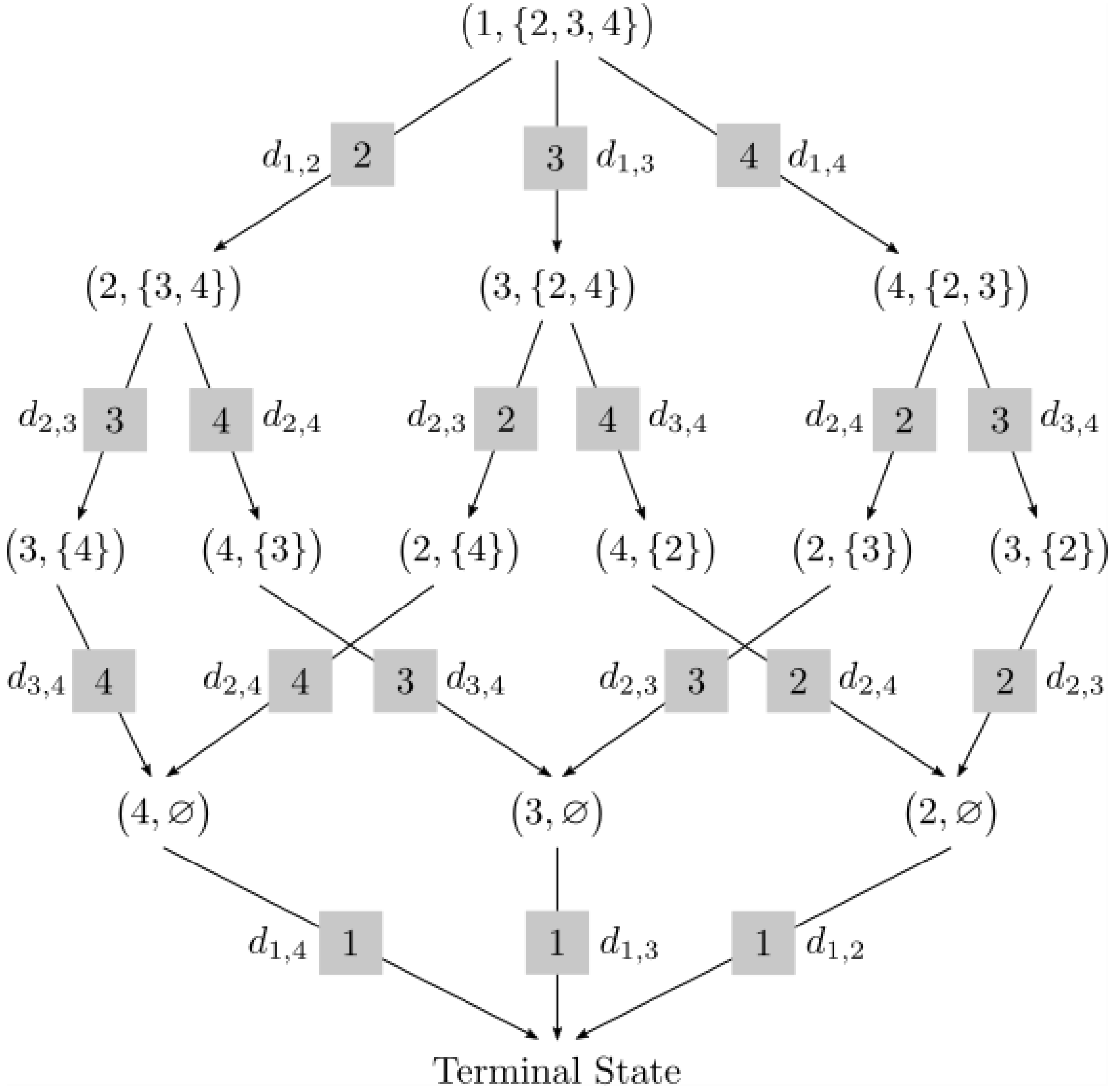
- k: Current city I'm in
- A: List of cities still to be visited

Optimality Bellman equation ($\gamma = 1$):

$$V^*(k, A) = max_{k'}[-d_{k,k'} + V^*(k', A\backslash\{k'\})]$$

Or equivalently,

$$C^*(k, A) = min_{k'}[d_{k,k'} + C^*(k', A\backslash\{k'\})]$$



As a Markov Decision Process

States: $s = (k, A)$

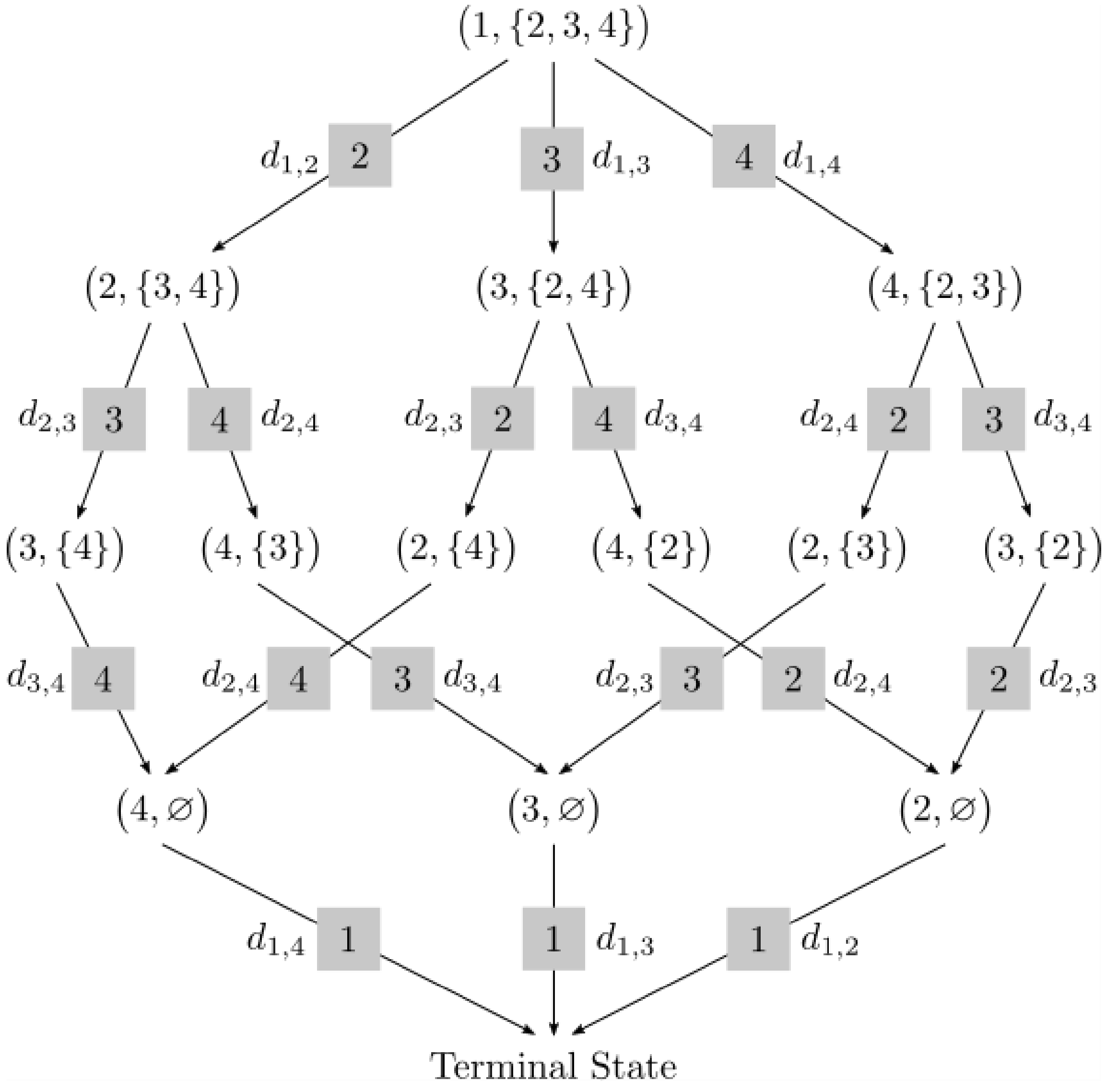
Current city I'm in

List of cities still to be visited

Optimality Bellman equation ($\gamma = 1$):

$$C^*(k, A) = min_{k'}[d_{k,k'} + C^*(k', A\backslash\{k'\})]$$

What is the meaning of $C^*(s)$?
($C^*(term.state) = 0$)

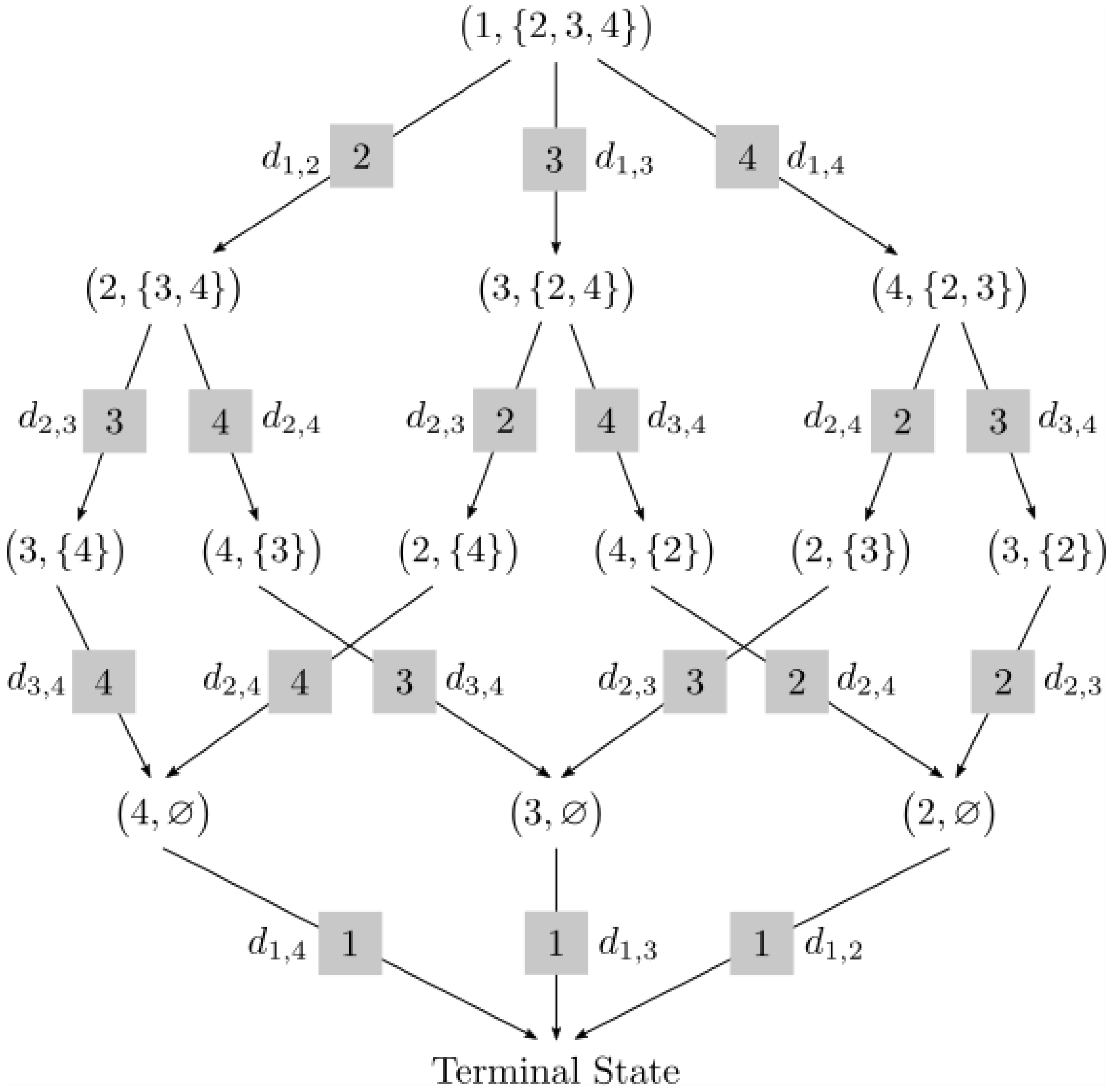


As a Markov Decision Process

States: $s = (k, A)$

Current city I'm in

List of cities still to be visited



Optimality Bellman equation ($\gamma = 1$):

$$C^*(k, A) = min_{k'}[d_{k,k'} + C^*(k', A\backslash\{k'\})]$$

The optimal cost is the minimal distance from k to the first city!

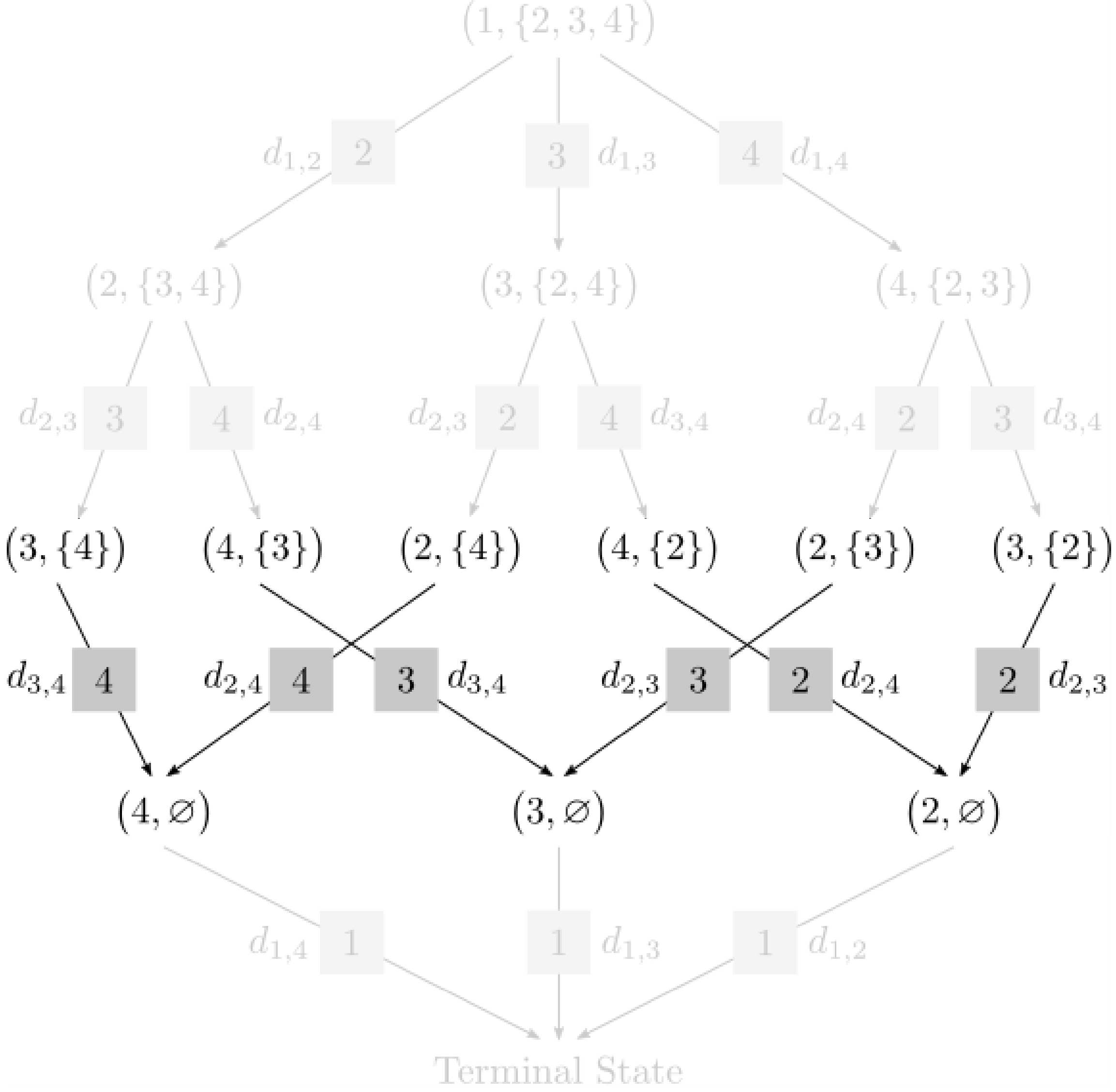
Solving the equation gives us the best policy, i.e. the best path.

How to solve the Bellman equation

Exploit the feedforward structure of the problem!

The cost of the states in this layer depends only on the costs in the next layer (and the distances)

$$C^*(k,A) = min_{k'}[d_{k,k'} + C^*(k', A\backslash\{k'\})]$$

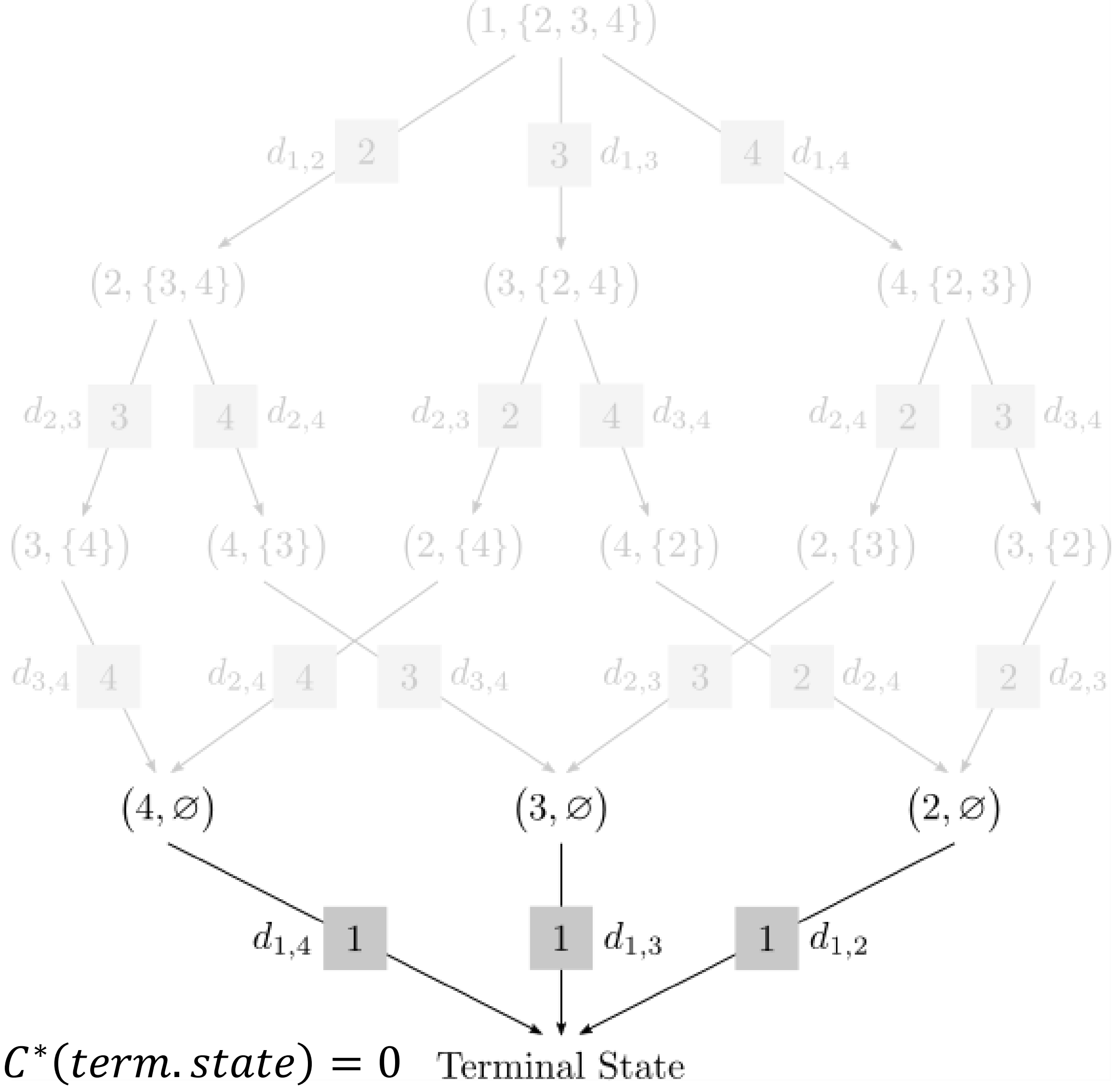


How to solve the Bellman equation

Exploit the feedforward structure of the problem!

Knowing the boundary condition at terminal state, the solution can be propagated backward for all the states

$$C^*(k, A) = min_{k'}[d_{k,k'} + C^*(k', A\backslash\{k'\})]$$



$C^*(term. state) = 0$

How to find the best path

Once the costs are known, the best path, $k_1^*, k_2^*, \ldots, k_N^*$, can be computed iteratively (now going forward) by knowing:

$$k_{t+1}^* = argmin_{k' \in A_t}[d_{k_t,k} + C^*(k, A \backslash \{k\})]$$