Optimizing Reachability Sets in Temporal Graphs by Delaying

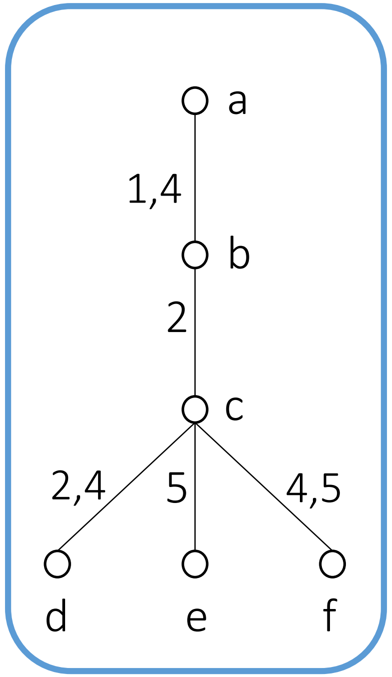
Argyrios Deligkas, Igor Potapov

Algorithmic Aspects of Temporal Graphs II

ICALP 2019

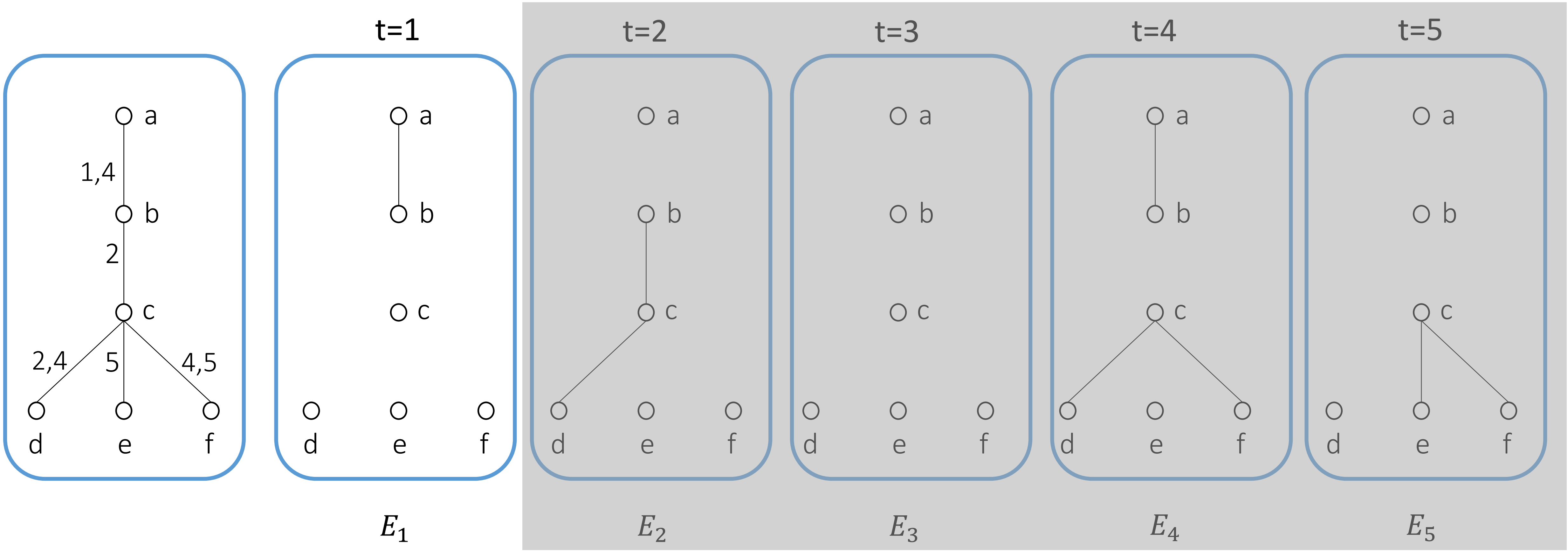
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



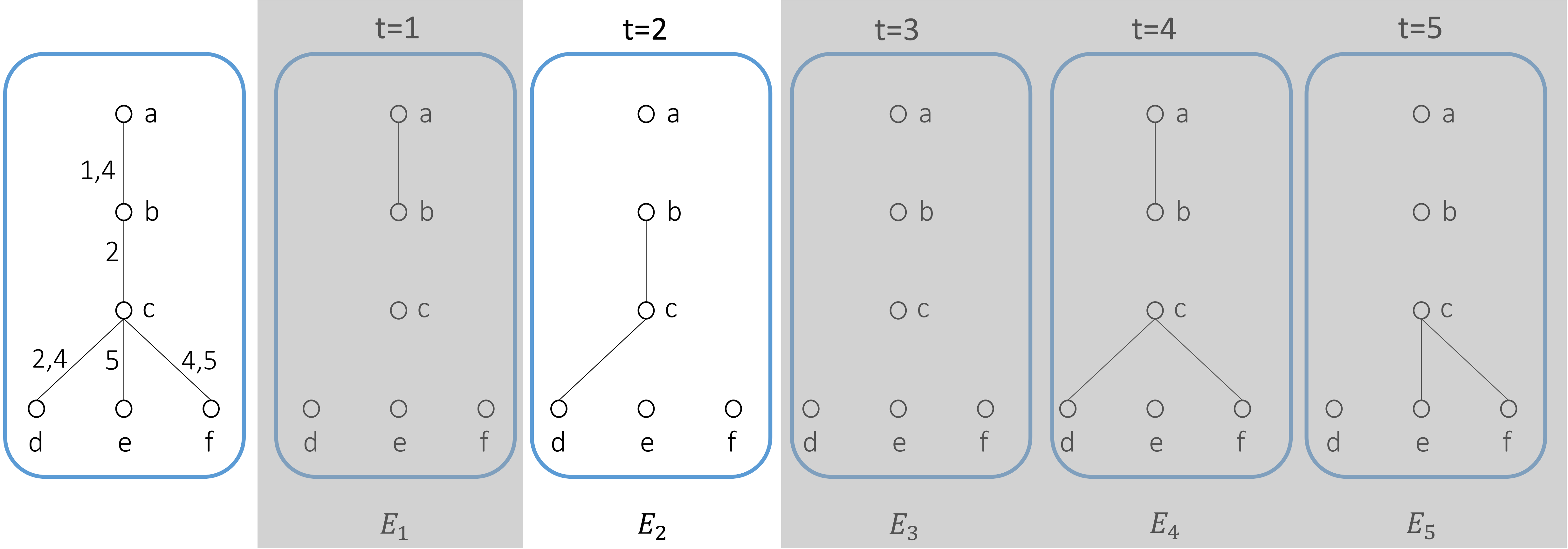
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



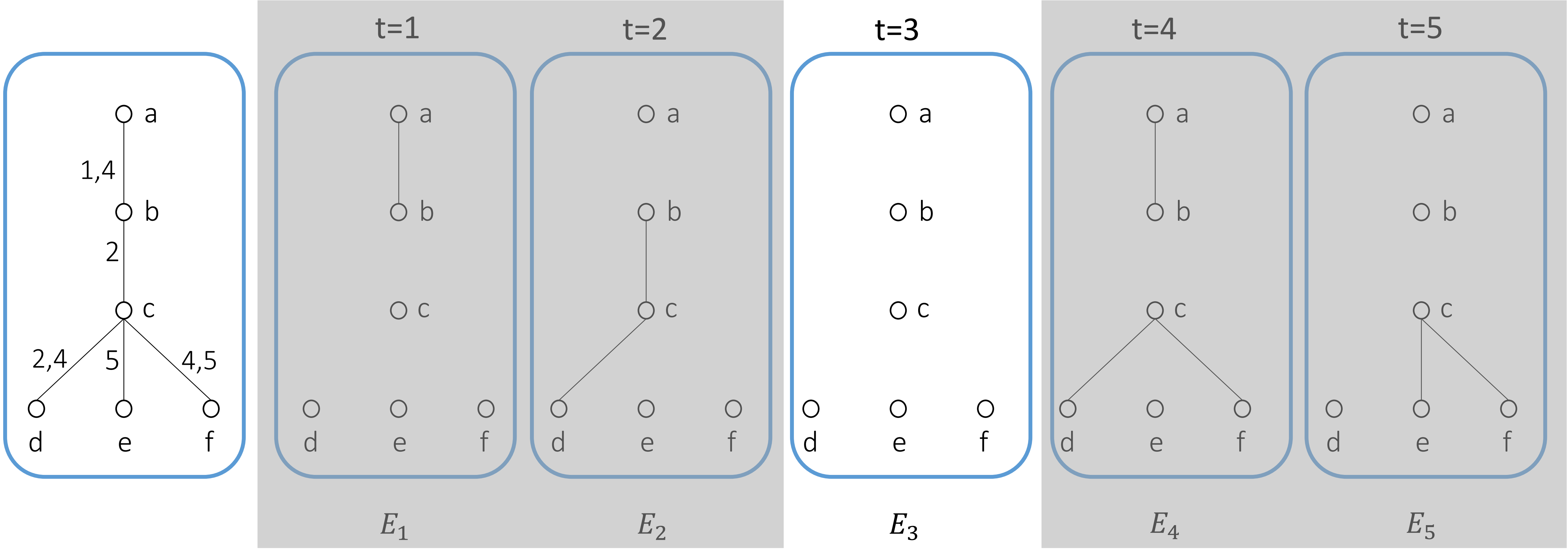
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



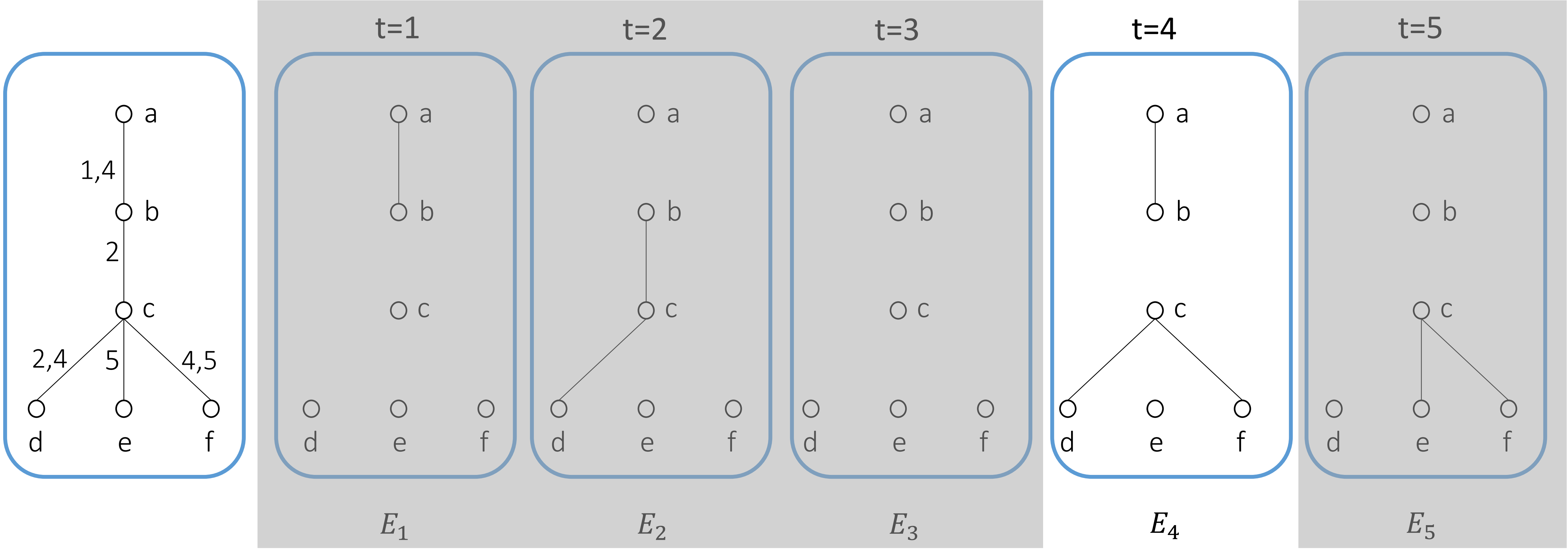
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



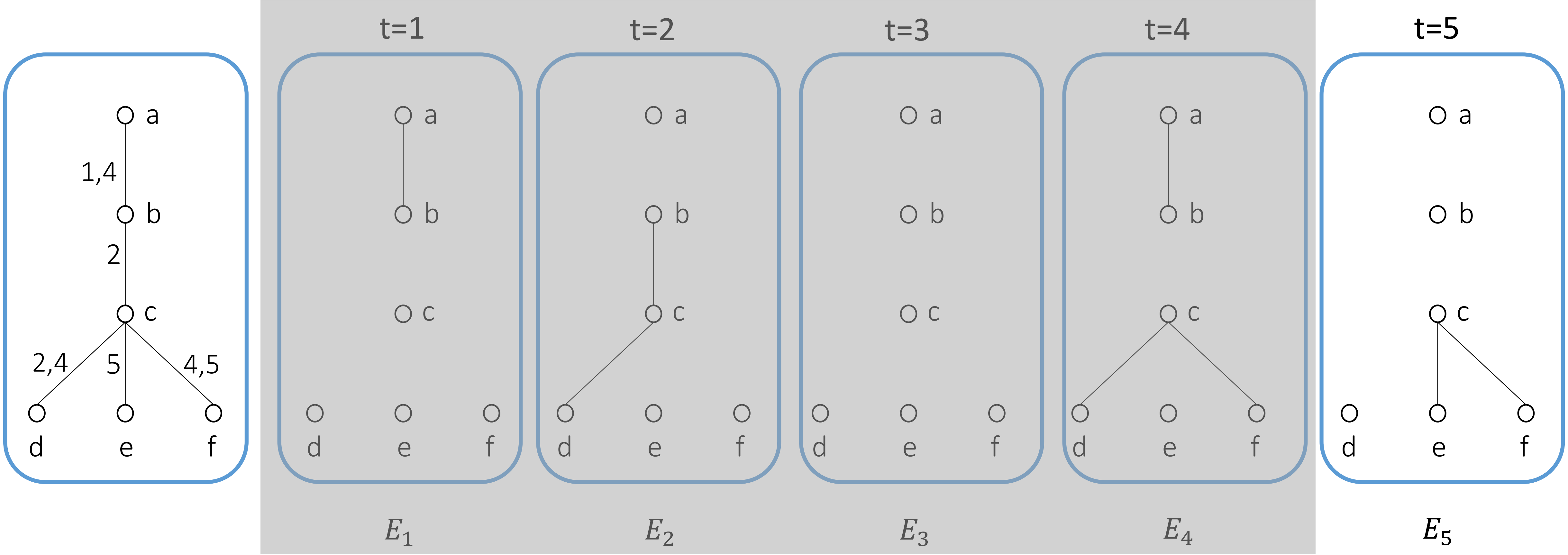
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



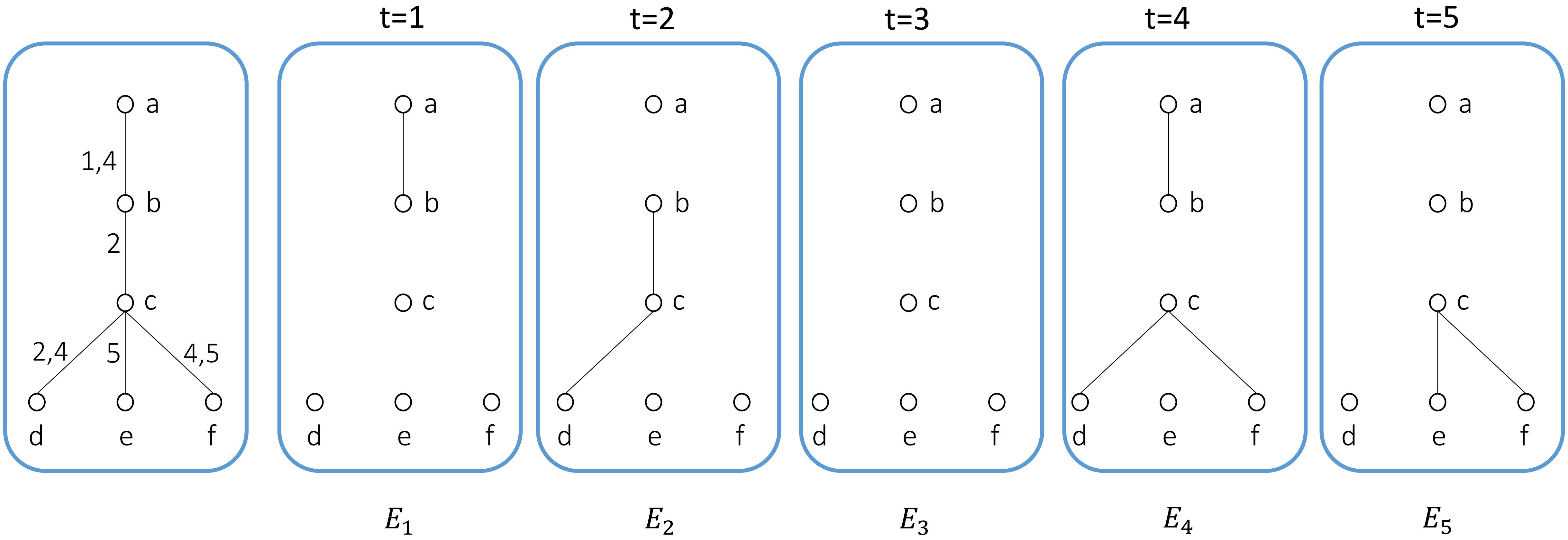
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



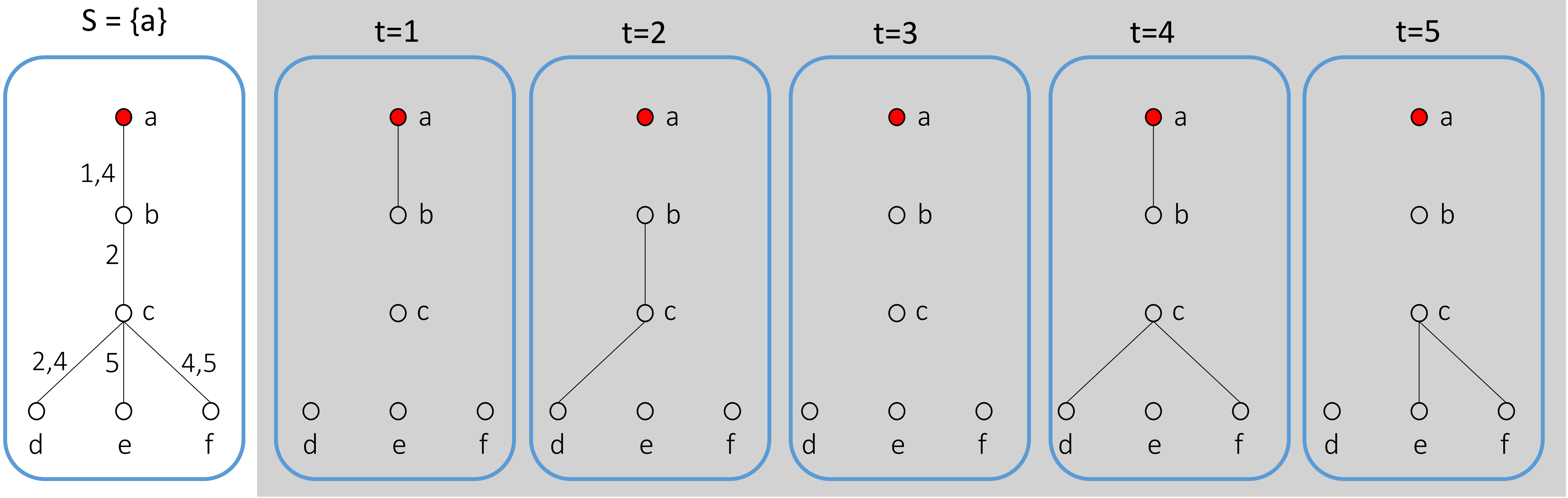
Temporal Graphs

- Graph $G = (V, E)$
- Labelling function T



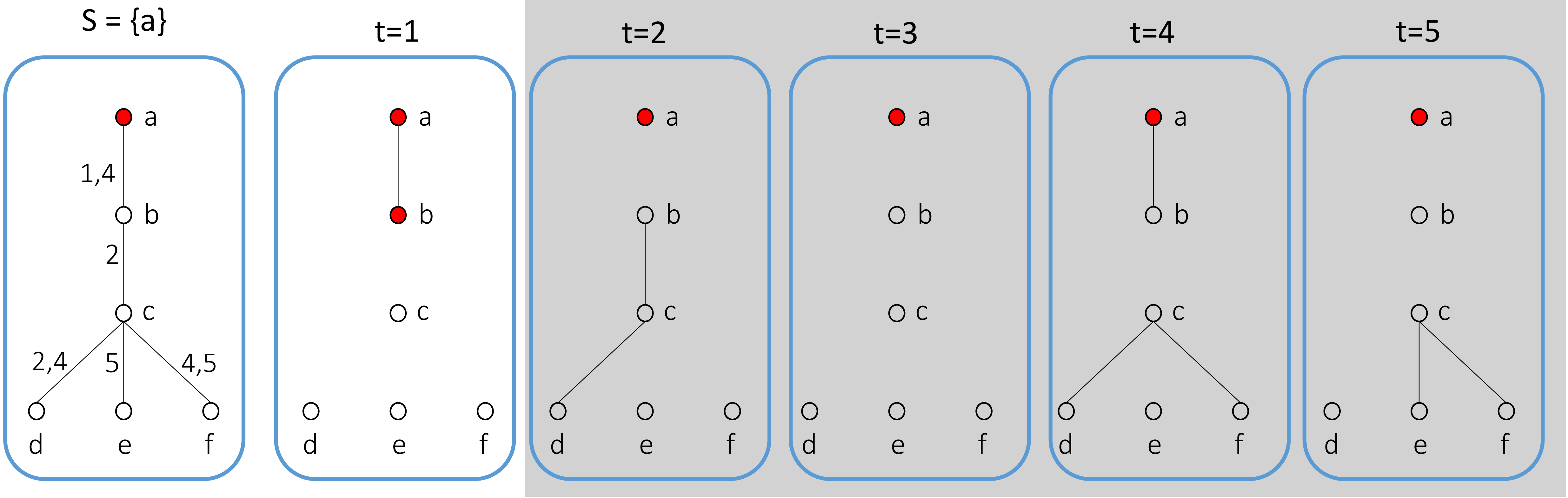
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



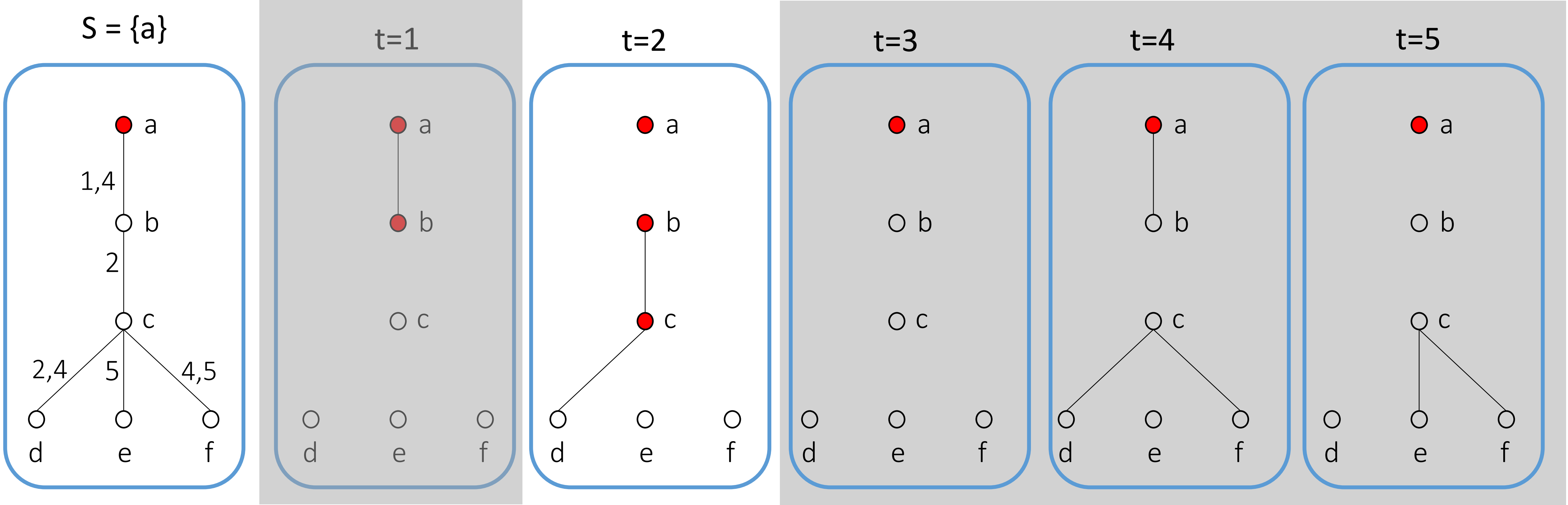
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



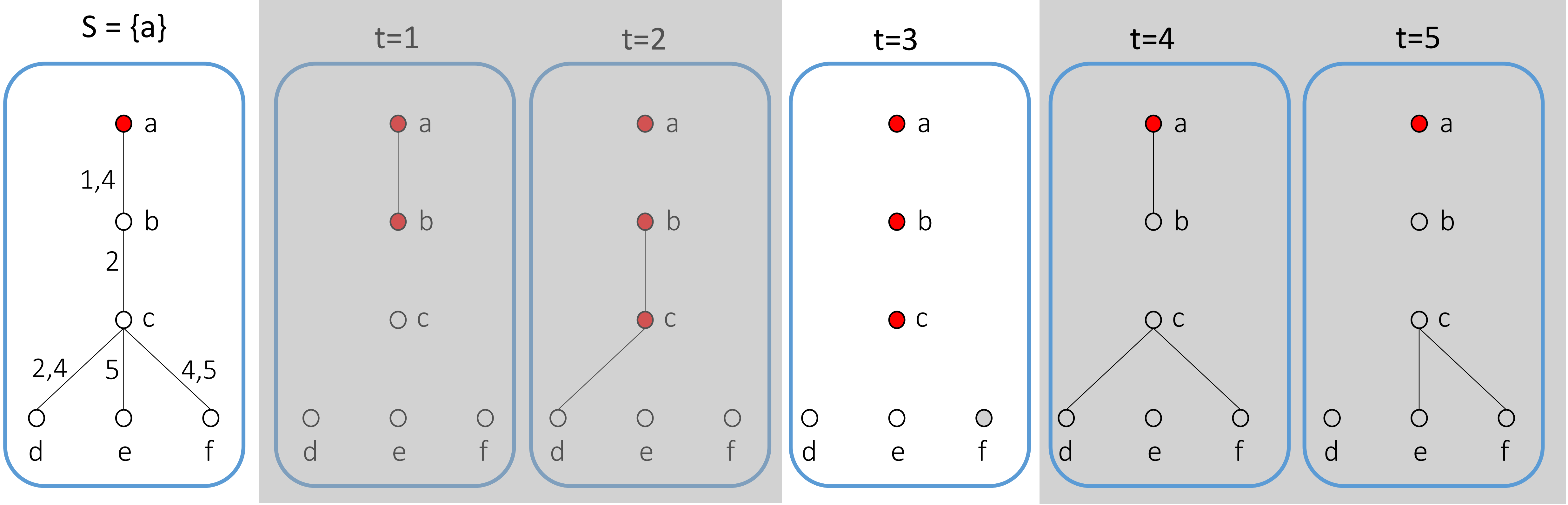
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



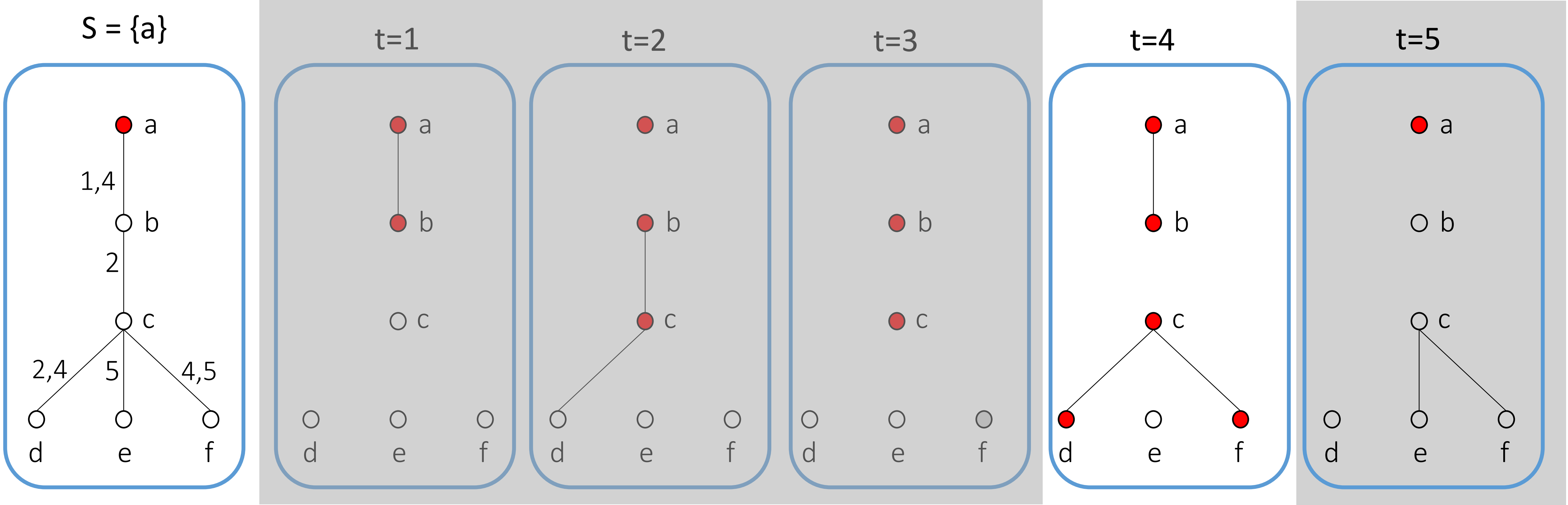
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



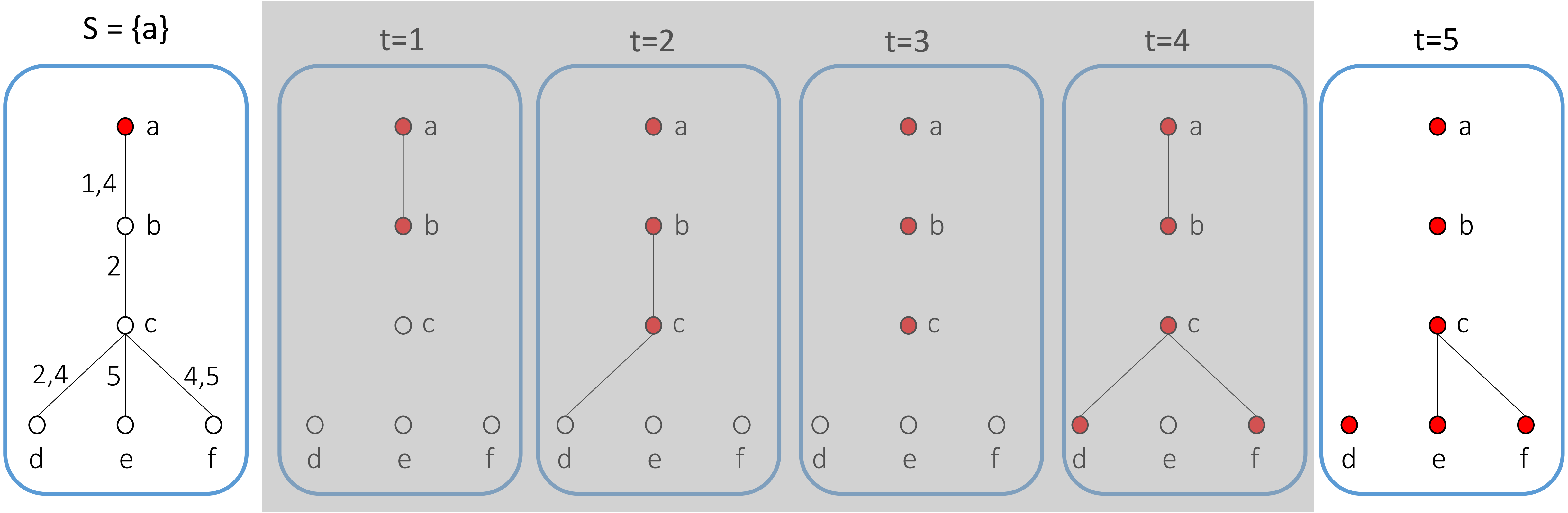
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



Reachability sets in temporal graphs

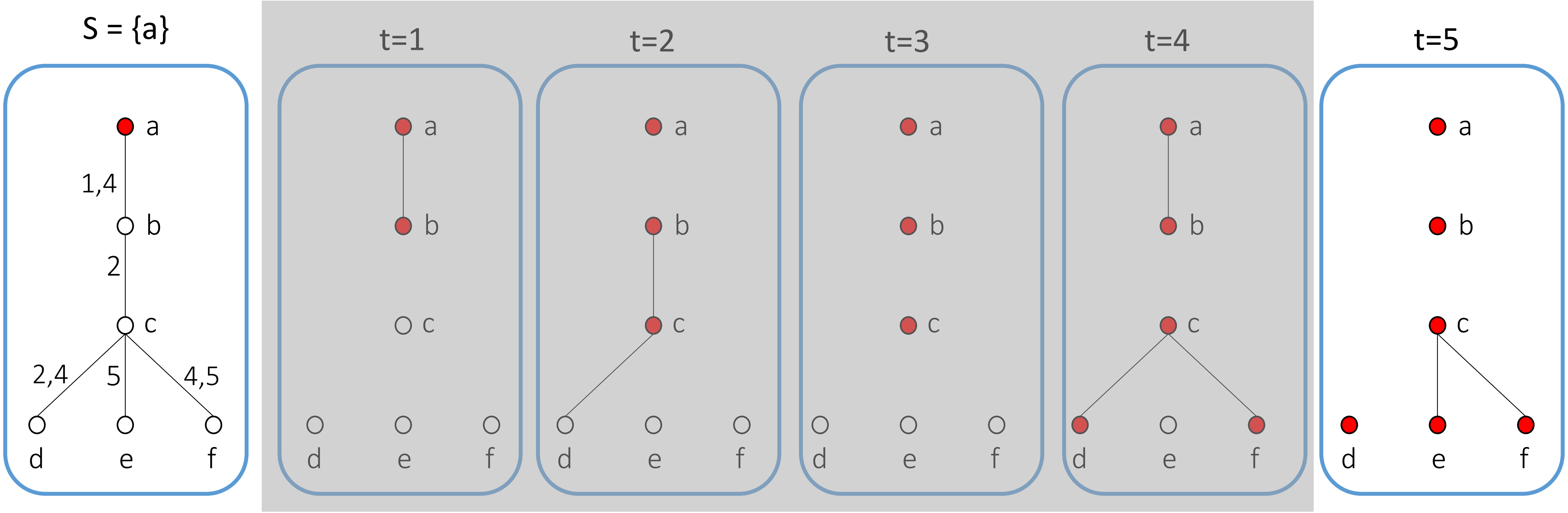
- Graph $G = (V, E)$
- Labelling function T
- Set of sources S



Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

$reach(v, \langle G, T \rangle)$
Reachability set of v: Set of reachable vertices from v



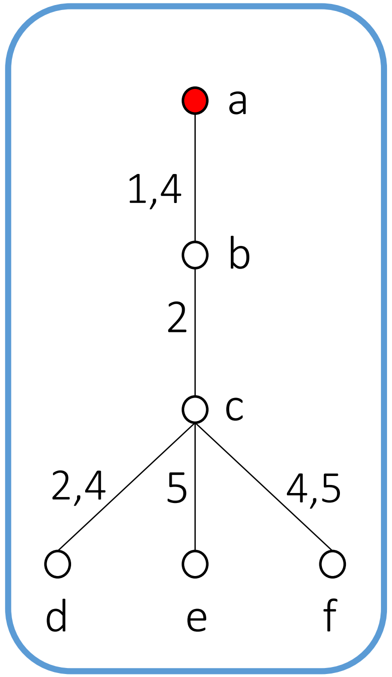
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

$reach(v, \langle G, T \rangle)$
Reachability set of v: Set of reachable vertices from v

Epidemiology
Restrict spread infection

S = {a}



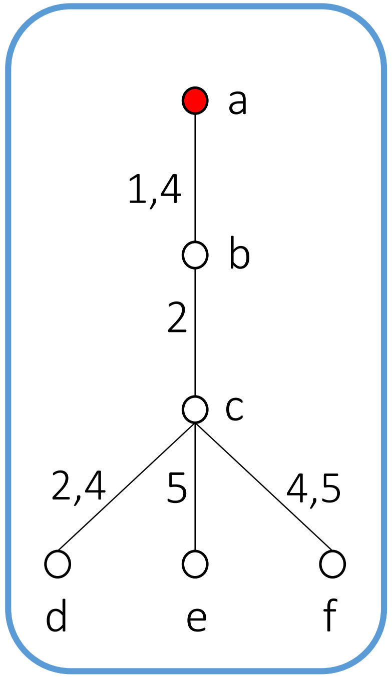
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

How to optimize the reachability set in a temporal graph?

Epidemiology
Restrict spread infection

S = {a}



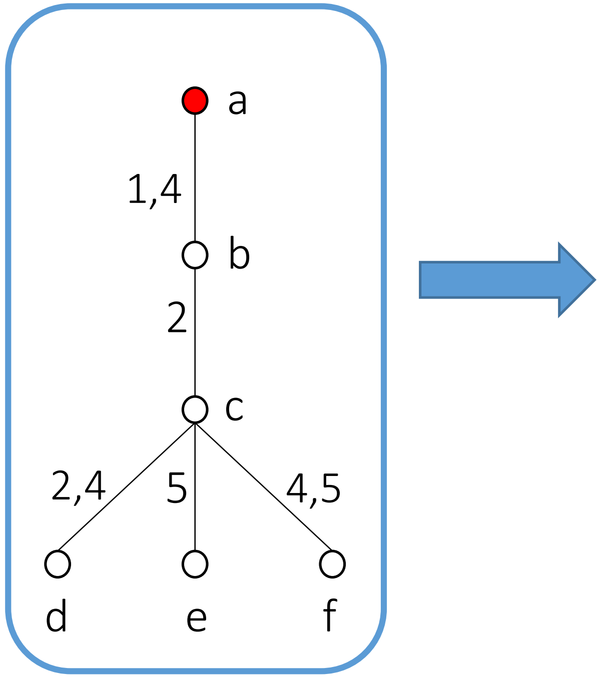
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

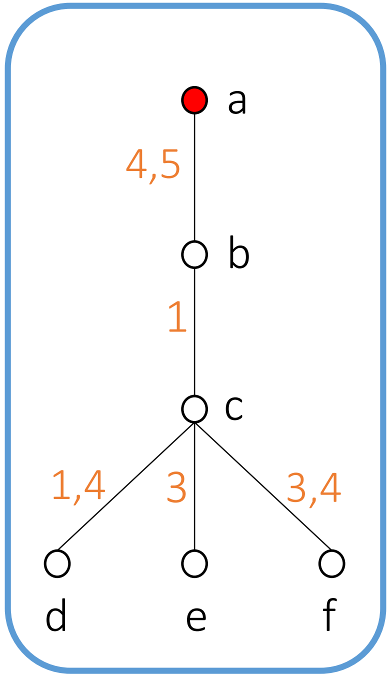
How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

S = {a}



S = {a}



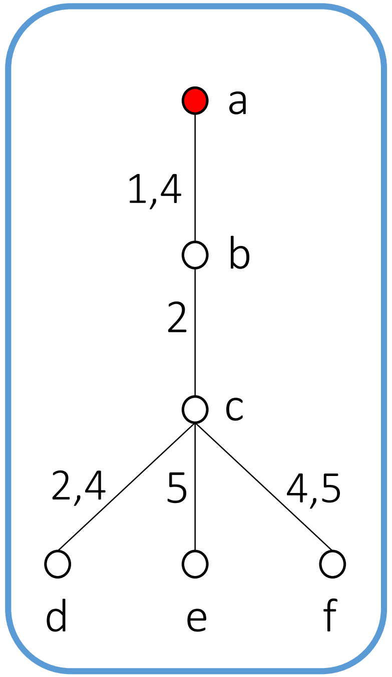
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

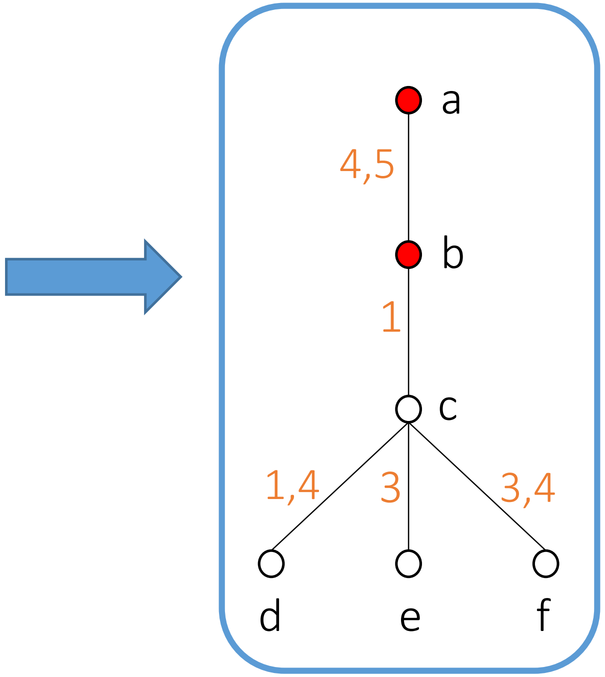
How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

S = {a}



S = {a}



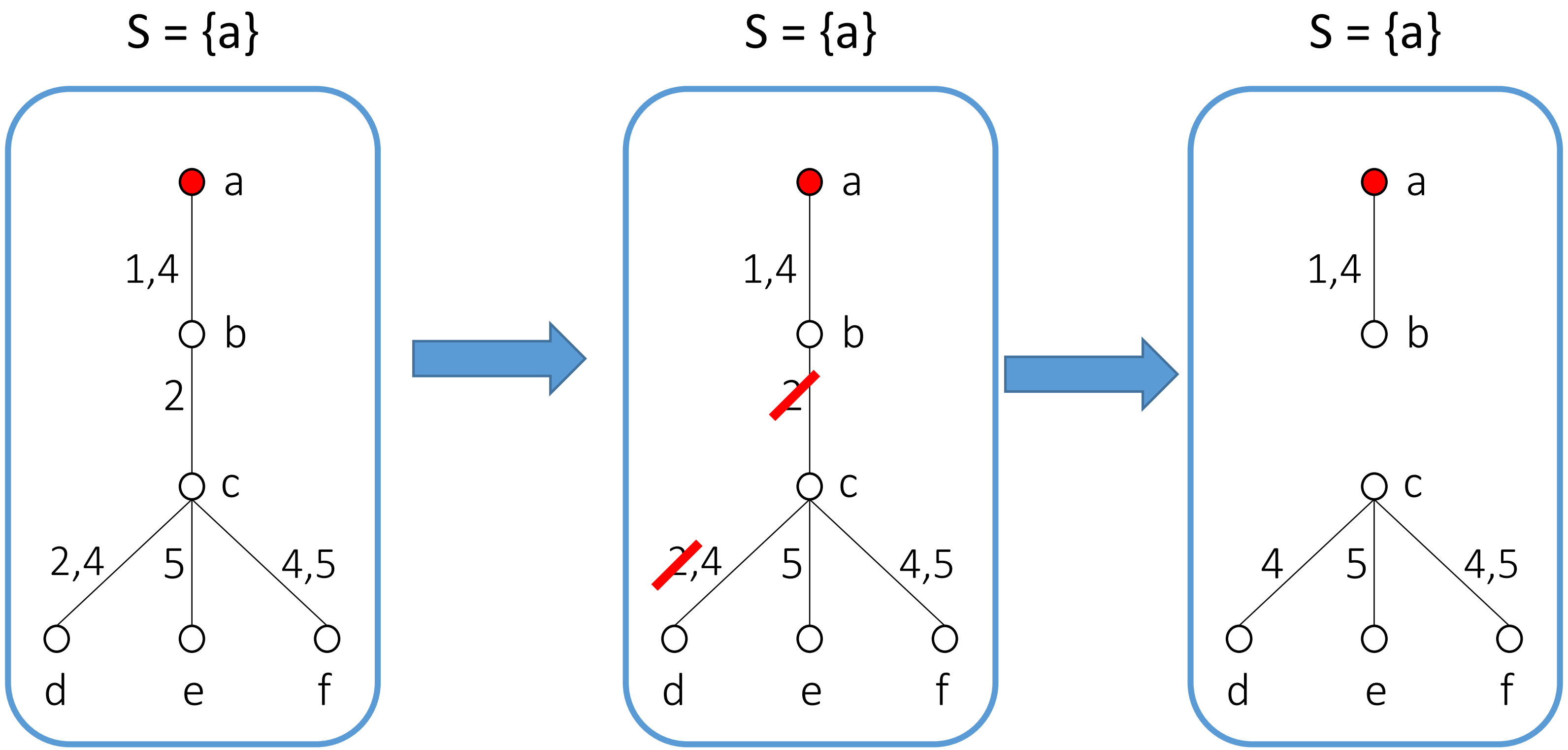
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

Approach 2: Edge deletion
Enright, Meeks, Mertzios, Zamaraev



Delete all edges with label 2

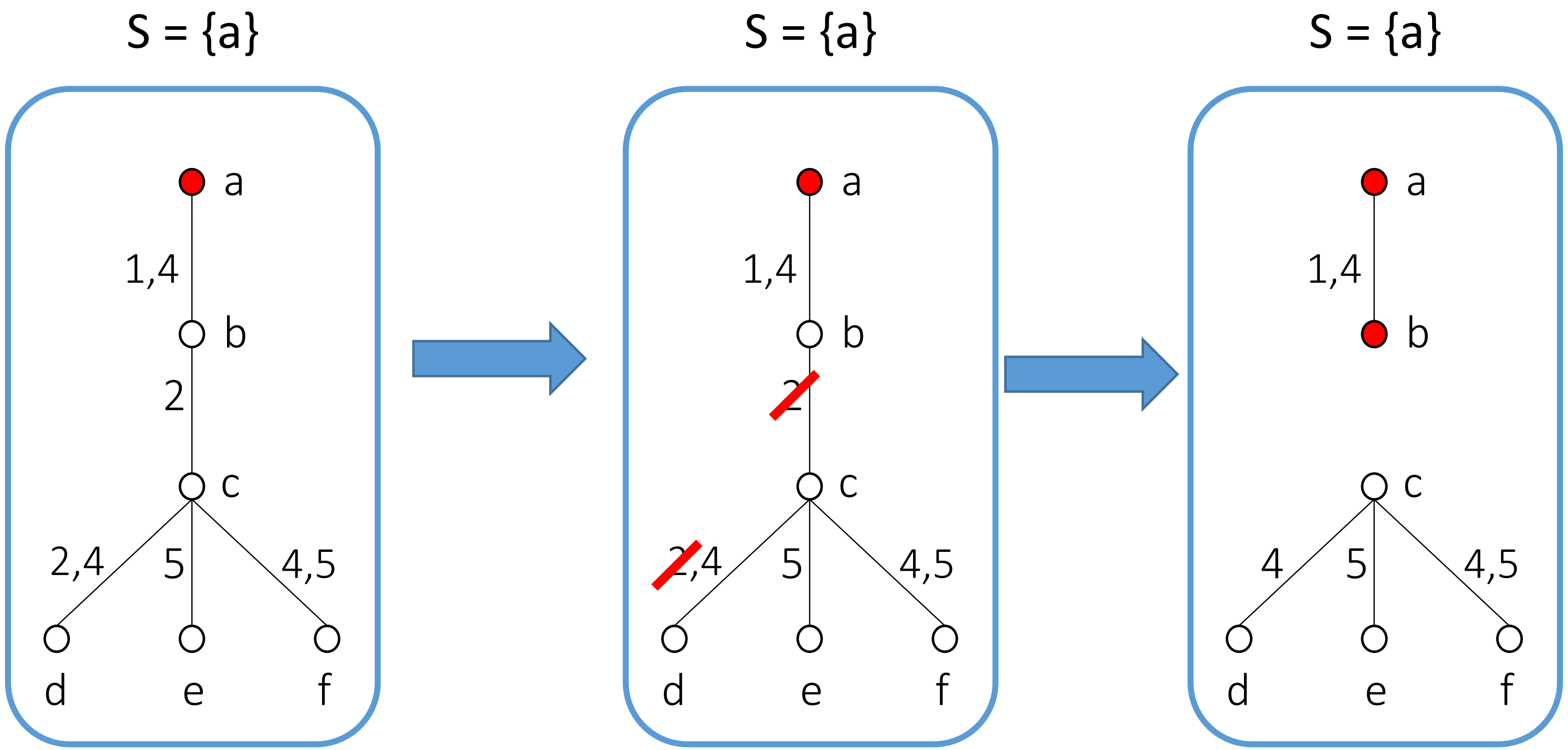
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

Approach 2: Edge deletion
Enright, Meeks, Mertzios, Zamaraev



Delete all edges with label 2

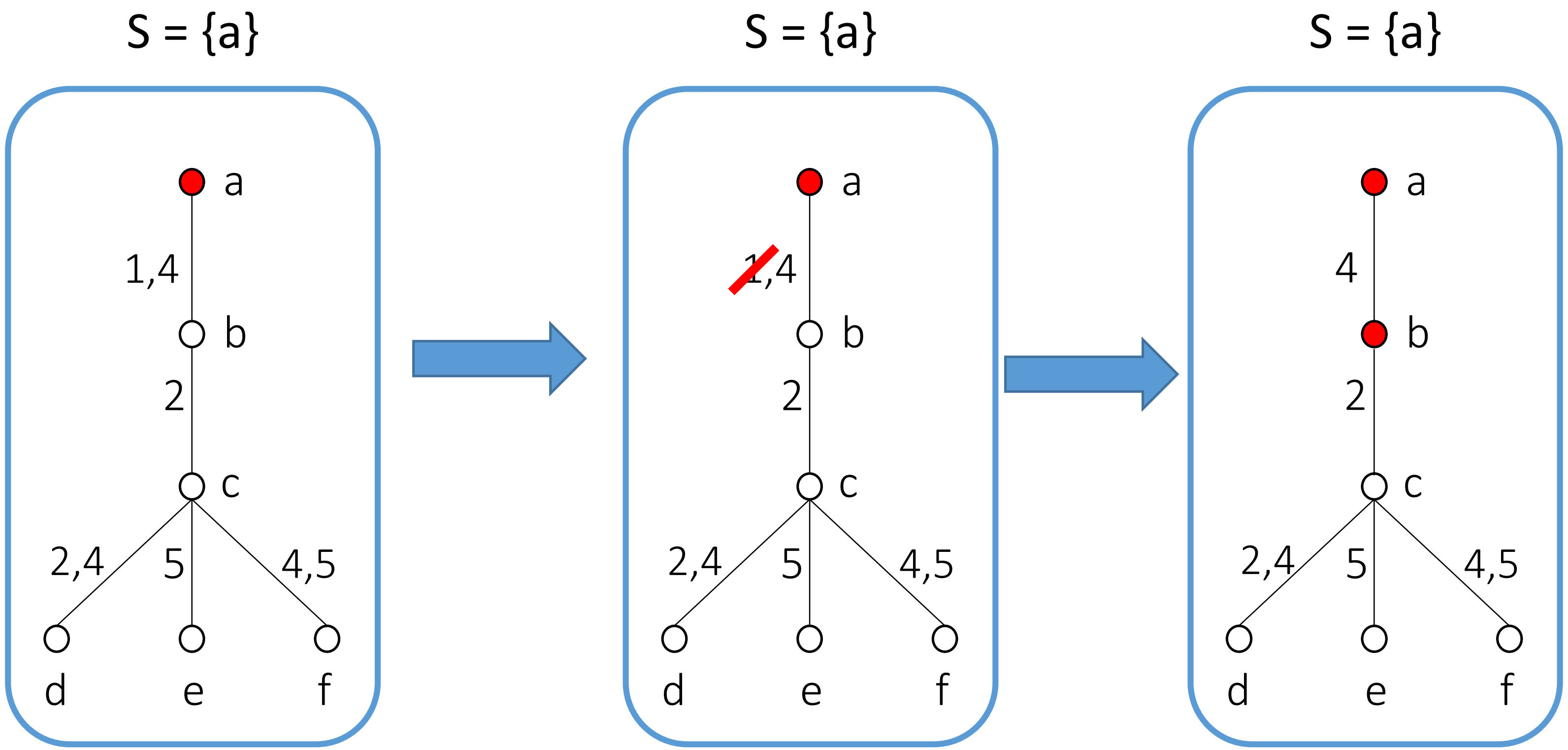
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

Approach 2: Edge deletion
Enright, Meeks, Mertzios, Zamaraev



Delete label 1 from edge ab

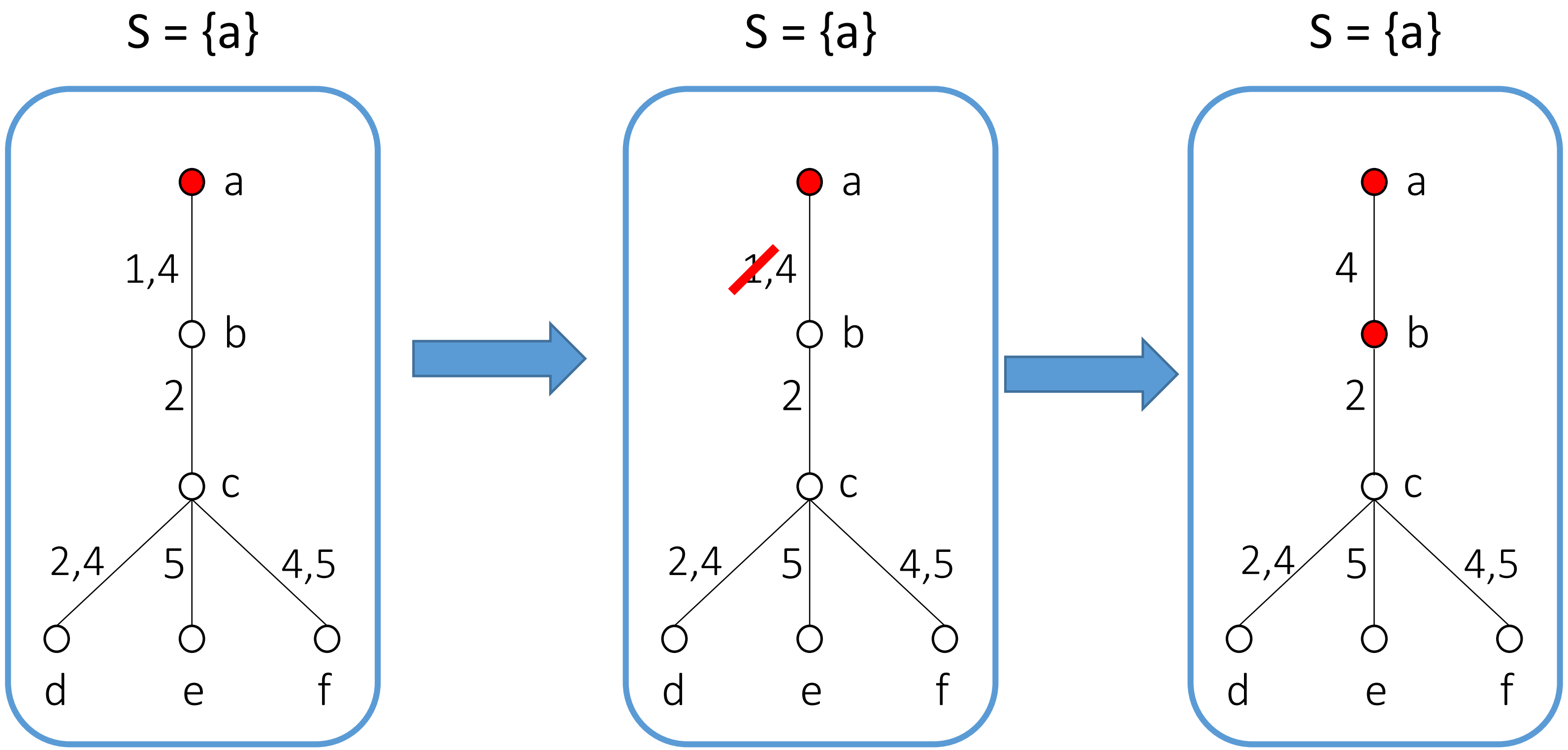
Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

Approach 2: Edge deletion
Enright, Meeks, Mertzios, Zamaraev

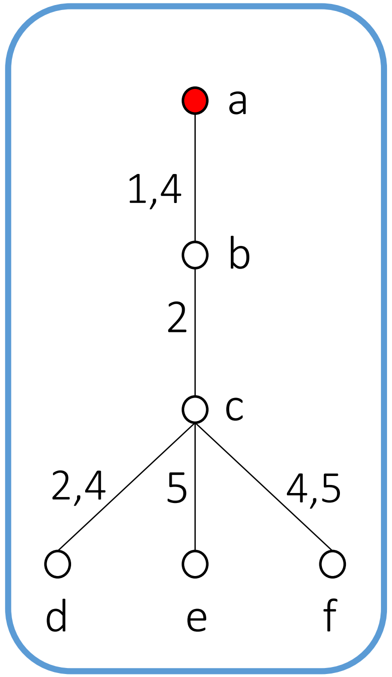


Delete label 1 from edge ab

Reachability sets in temporal graphs

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

S = {a}



How to optimize the reachability set in a temporal graph?

Approach 1: Reshuffling
Enright, Meeks

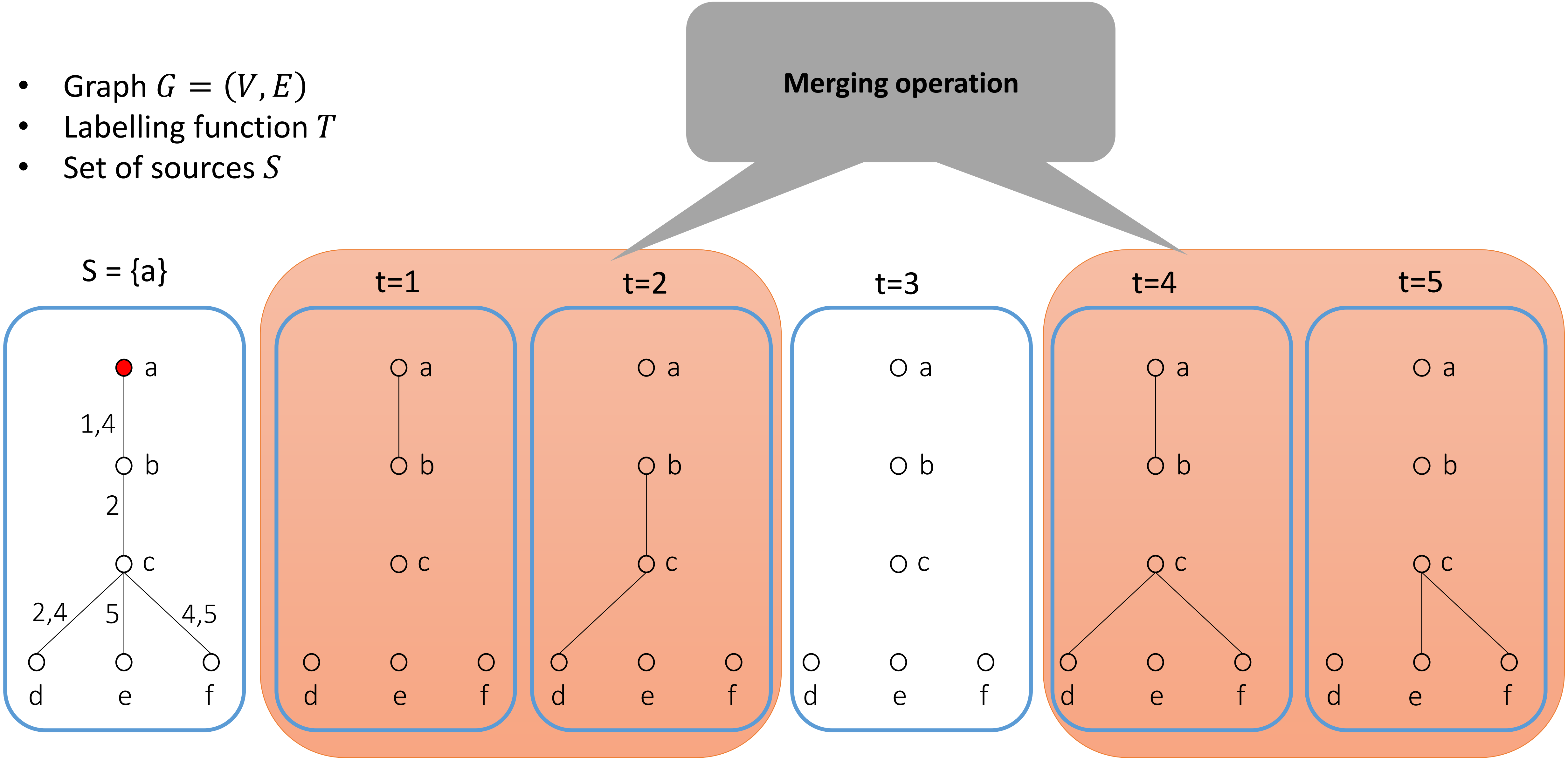
Not always possible in real-life networks
(too many changes)

Approach 2: Edge deletion
Enright, Meeks, Mertzios, Zamaraev

Can create deadends in the network
(blocks the flow)

Our approach: Delaying

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

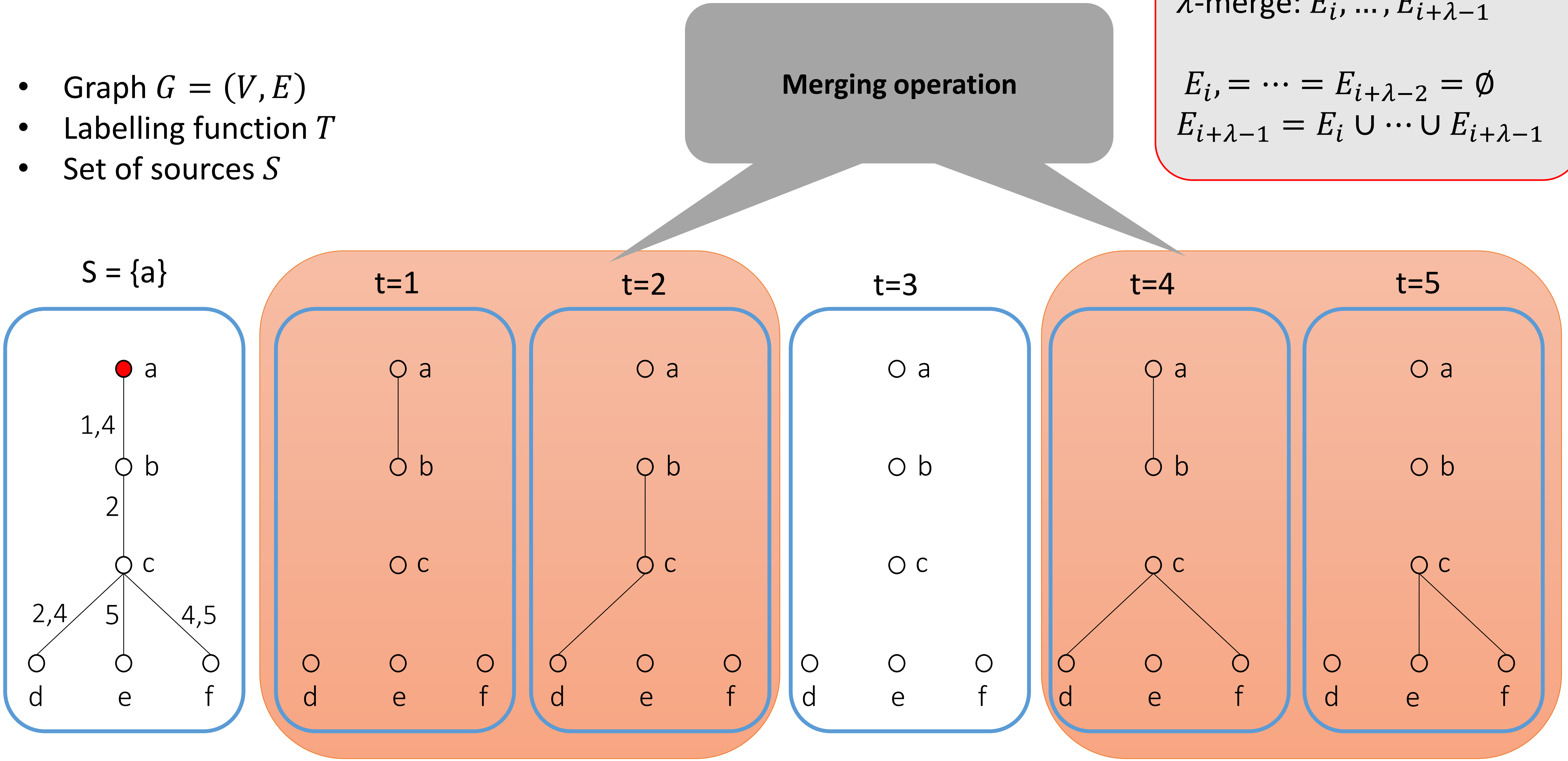


Our approach: Delaying

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

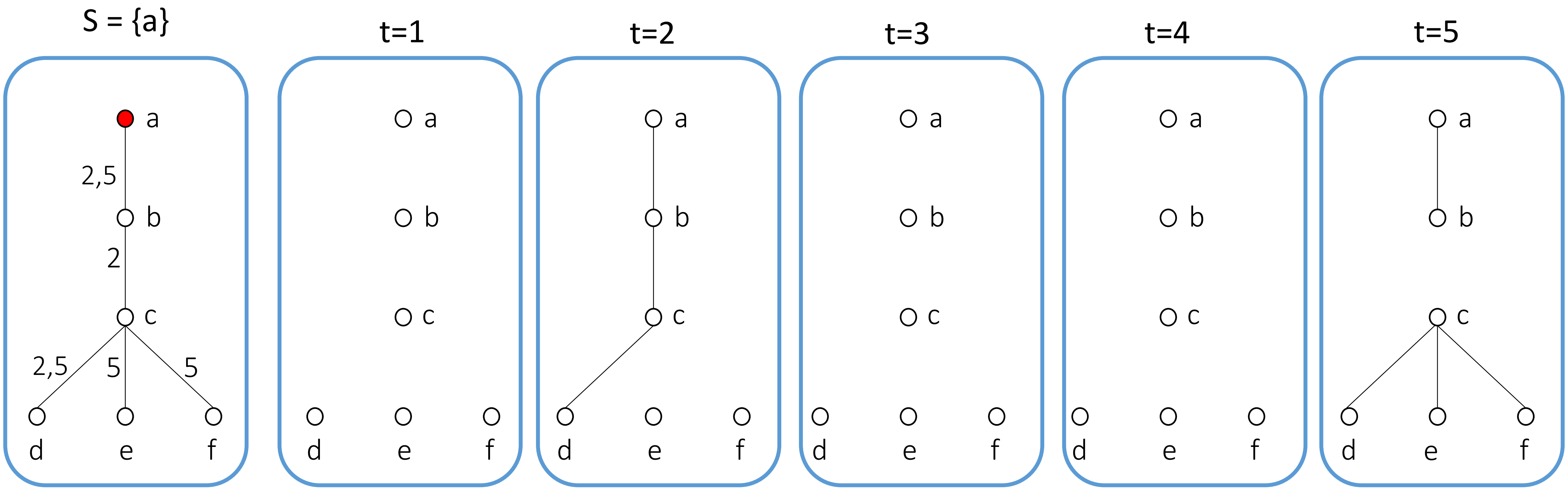


Our approach: Delaying

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

λ-merge: $E_i, \dots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

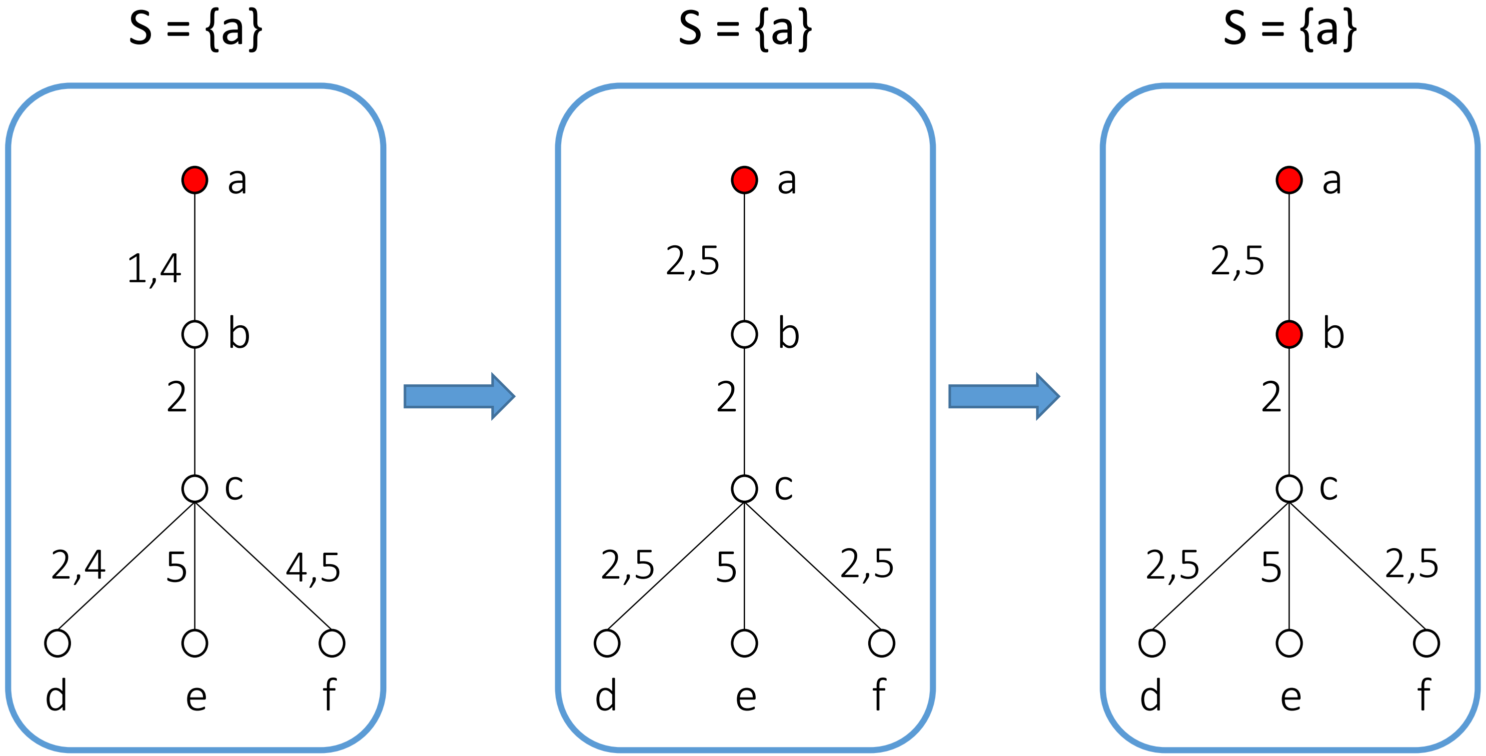


Our approach: Delaying

- Graph $G = (V, E)$
- Labelling function T
- Set of sources S

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$



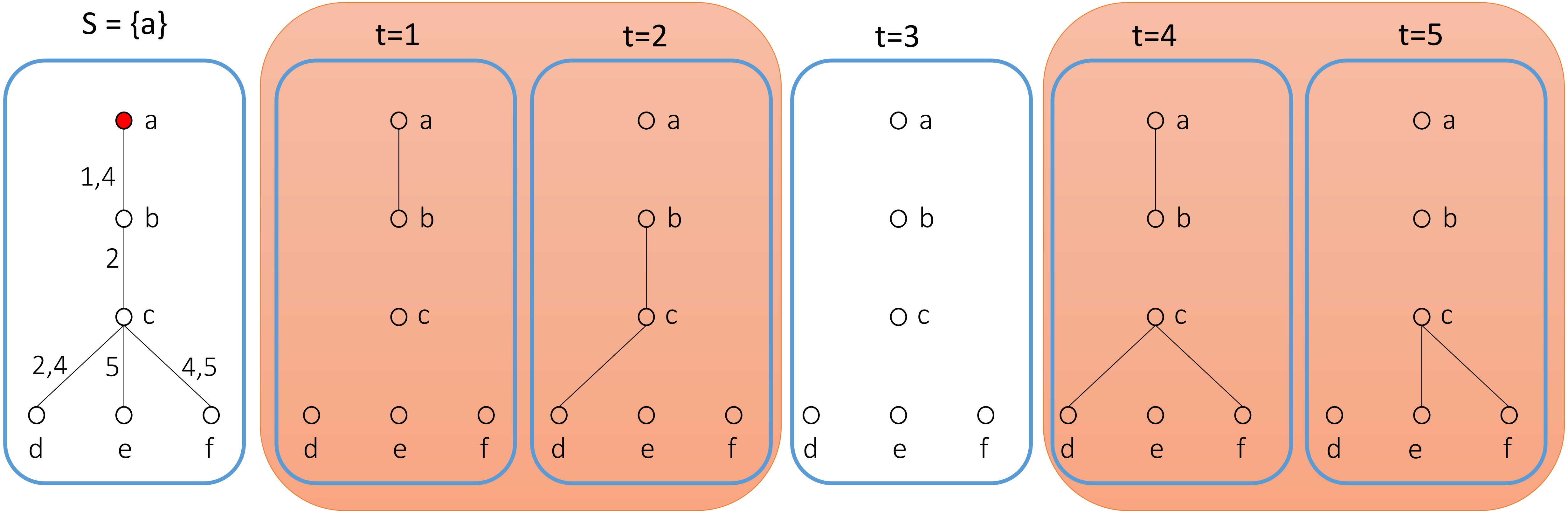
- Minimum modification/disturbance of the original network
- Does not create deadends
- Intuitive

Merging Schemes

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

λ-merge: $E_i, \dots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

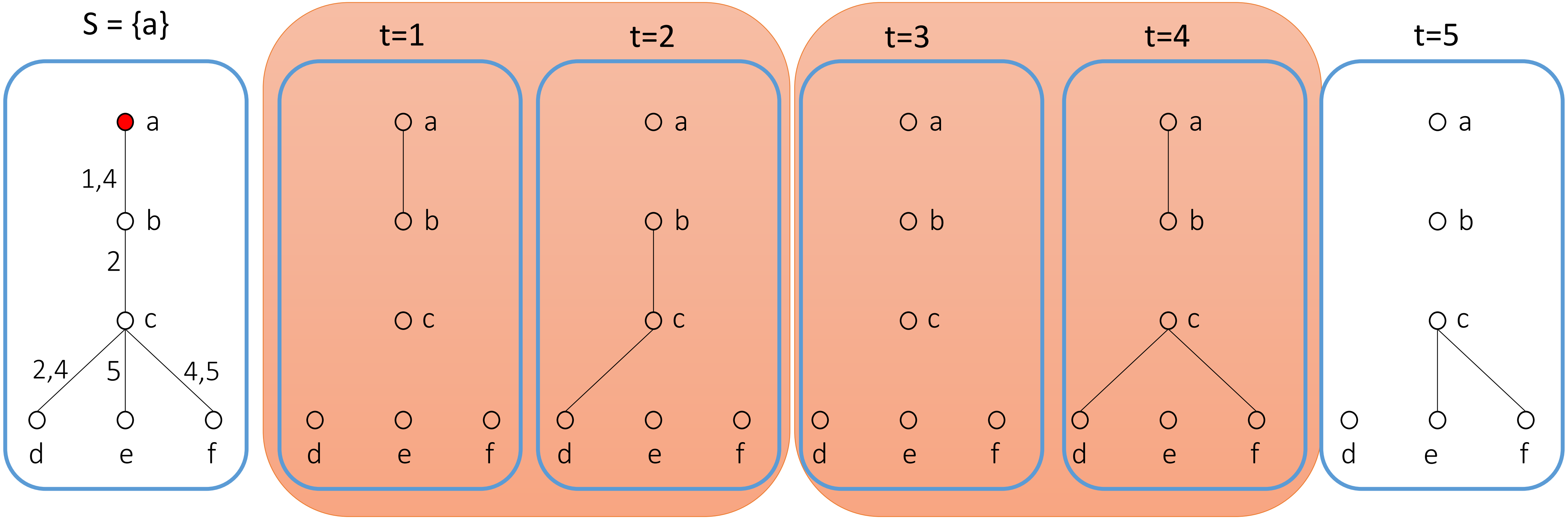


Merging Schemes

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

λ-merge: $E_i, \dots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

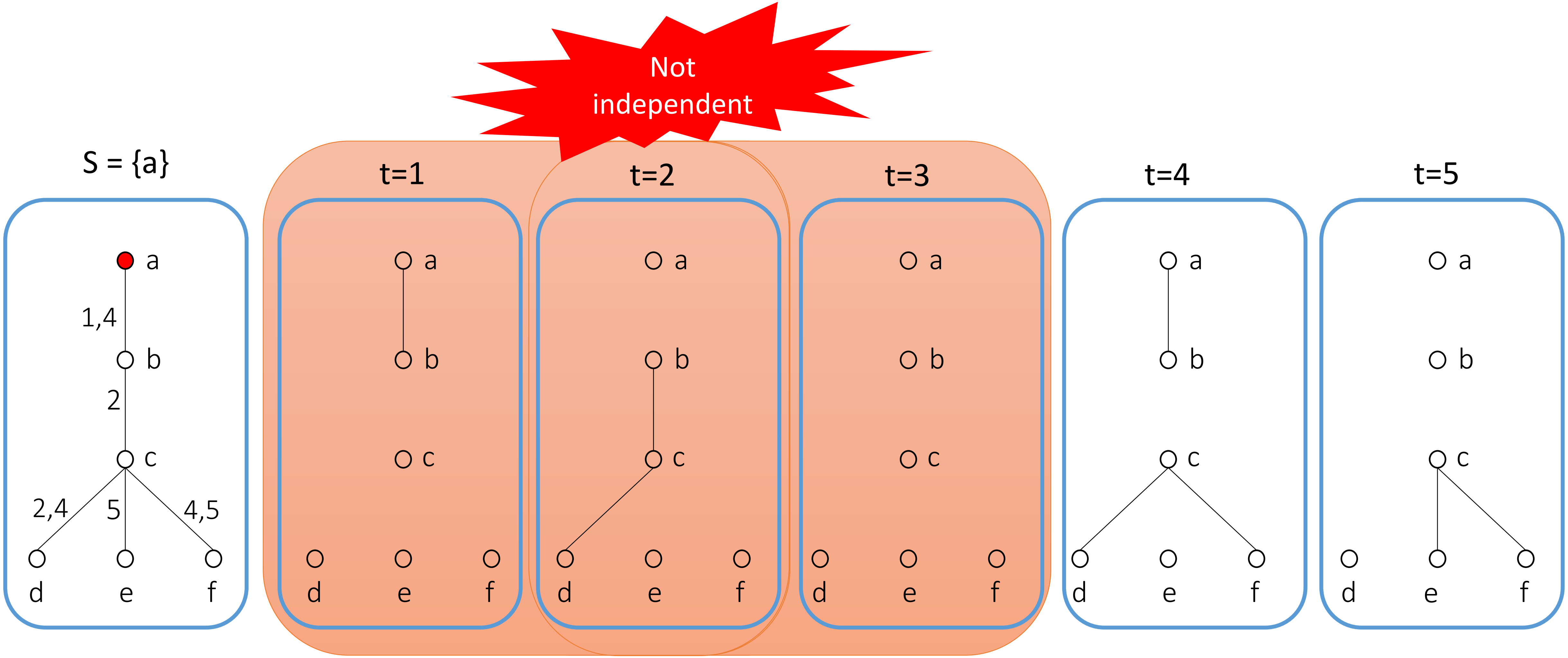


Merging Schemes

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$



Merging Schemes: Objectives

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

Input
- Temporal graph $\langle G, T \rangle$
- Integers λ and μ
- Set of sources S

Merging Schemes: Objectives

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

λ-merge: $E_i, \dots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

Input
- Temporal graph $\langle G, T \rangle$
- Integers λ and μ
- Set of sources S

Minimization Objectives
(λ, μ)-merging scheme uses **at most** μ independent λ-merges

- MinReach: $\min |\cup_{v \in S} reach(v, \langle G, T \rangle)|$
- MinMaxReach: $\min \max_{v \in S} |reach(v, \langle G, T \rangle)|$
- MinAvgReach: $\min \sum_{v \in S} |reach(v, \langle G, T \rangle)|$

Merging Schemes: Objectives

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

Input
- Temporal graph $\langle G, T \rangle$
- Integers λ and μ
- Set of sources S

When $|S| = 1$, all problems coincide

Minimization Objectives
(λ, μ)-merging scheme uses **at most** μ independent λ-merges

- MinReach: $\min |\cup_{v \in S} reach(v, \langle G, T \rangle)|$
- MinMaxReach: $\min \max_{v \in S} |reach(v, \langle G, T \rangle)|$
- MinAvgReach: $\min \sum_{v \in S} |reach(v, \langle G, T \rangle)|$

Merging Schemes: Objectives

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

Input
- Temporal graph $\langle G, T \rangle$
- Integers λ and μ
- Set of sources S

Maximization Objectives
(λ, μ)-merging scheme uses **at least** μ independent λ-merges

- MaxReach: $\max \left| \cup_{v \in S} reach(v, \langle G, T \rangle) \right|$
- MaxMinReach: $\max \min_{v \in S} |reach(v, \langle G, T \rangle)|$
- MaxAvgReach: $\max \sum_{v \in S} |reach(v, \langle G, T \rangle)|$

Merging Schemes: Objectives

λ-merge: $E_i, \ldots, E_{i+\lambda-1}$

$E_i, = \cdots = E_{i+\lambda-2} = \emptyset$
$E_{i+\lambda-1} = E_i \cup \cdots \cup E_{i+\lambda-1}$

A (λ, μ)-merging scheme uses at most/least μ **independent** λ-merges

Input
- Temporal graph $\langle G, T \rangle$
- Integers λ and μ
- Set of sources S

When $|S| = 1$, all problems coincide

Maximization Objectives
(λ, μ)-merging scheme uses **at least** μ independent λ-merges

- MaxReach: $\max |\cup_{v \in S} reach(v, \langle G, T \rangle)|$
- MaxMinReach: $\max \min_{v \in S} |reach(v, \langle G, T \rangle)|$
- MaxAvgReach: $\max \sum_{v \in S} |reach(v, \langle G, T \rangle)|$

Our Results

Problem	Graph	# Sources	# Labels/Edge	# Edges/Step
MinReach	Path	O(n)	1	3
MinReach MinMaxReach MinAvgReach	Tree max degree 3	1	1	1
MaxReach	Path	O(n)	1	4
MaxReach MaxMinReach MaxAvgReach	Bipartite Max degree 3	1	1	4
MaxReach MaxMinReach MaxAvgReach	Tree max degree 3	1	1	10

NP-hard for every λ

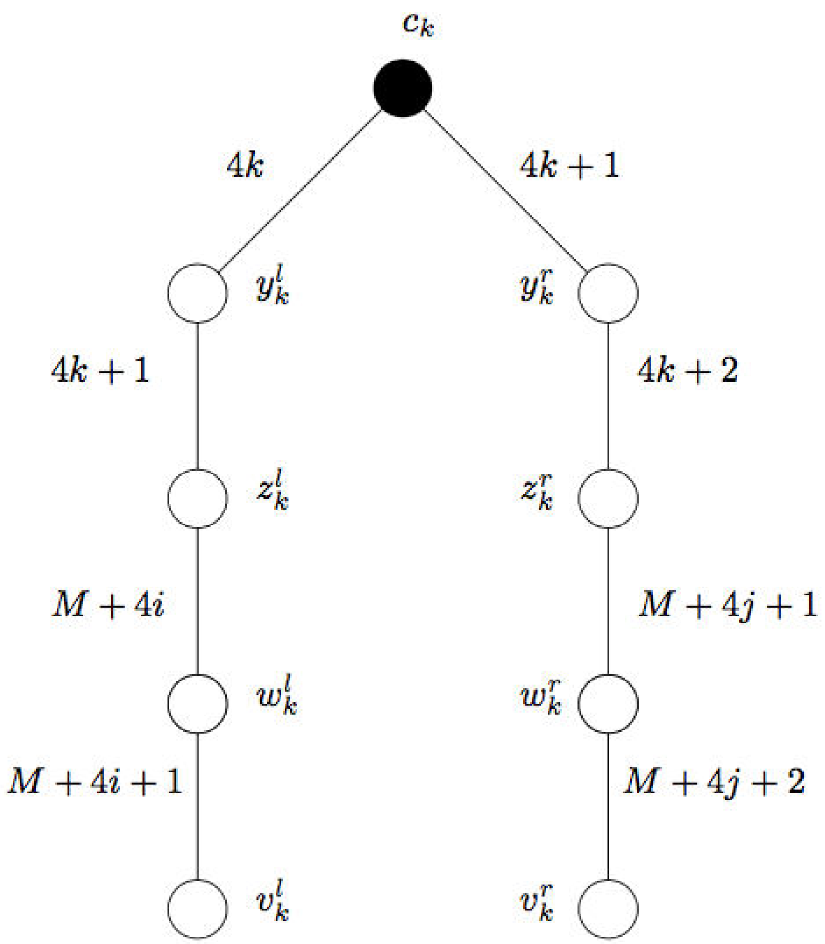
Our Results

Problem	Graph	# Sources	# Labels/Edge	# Edges/Step
MinReach	Path	O(n)	1	3
MinReach MinMaxReach MinAvgReach	Tree max degree 3	1	1	1
MaxReach	Path	O(n)	1	4
MaxReach MaxMinReach MaxAvgReach	Bipartite Max degree 3	1	1	4
MaxReach MaxMinReach MaxAvgReach	Tree max degree 3	1	1	10

NP-hard for every **λ**

- **DAGs**
- **Unit disk graphs**
- **Approximation preserving, no PTAS**

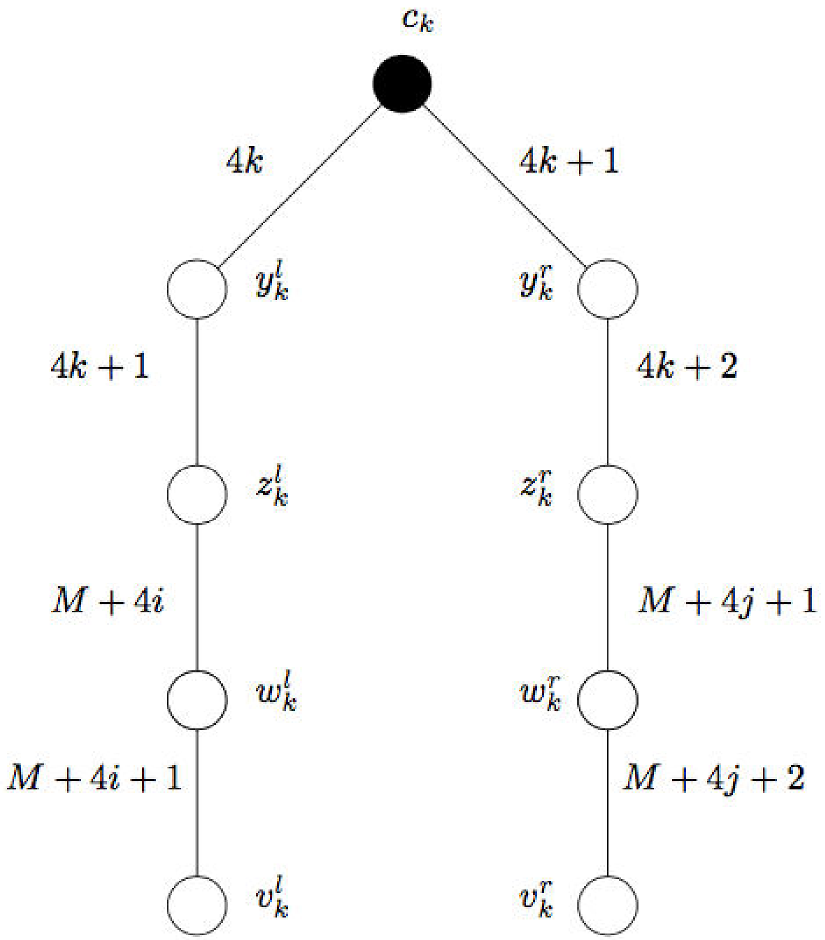
Idea



Reduction from Max2SAT(3)

Open Questions

- Approximation Algorithms
- Tractable Cases/Graph Classes
- FPT algorithms



Thanks!!