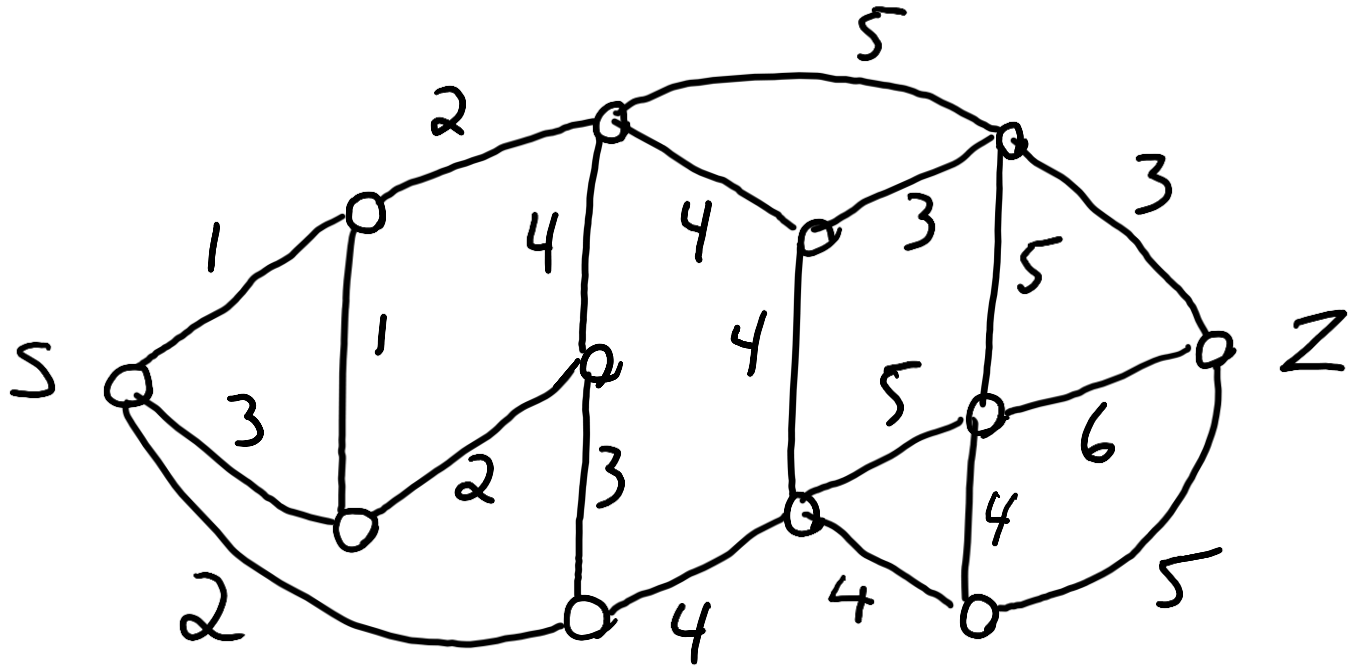
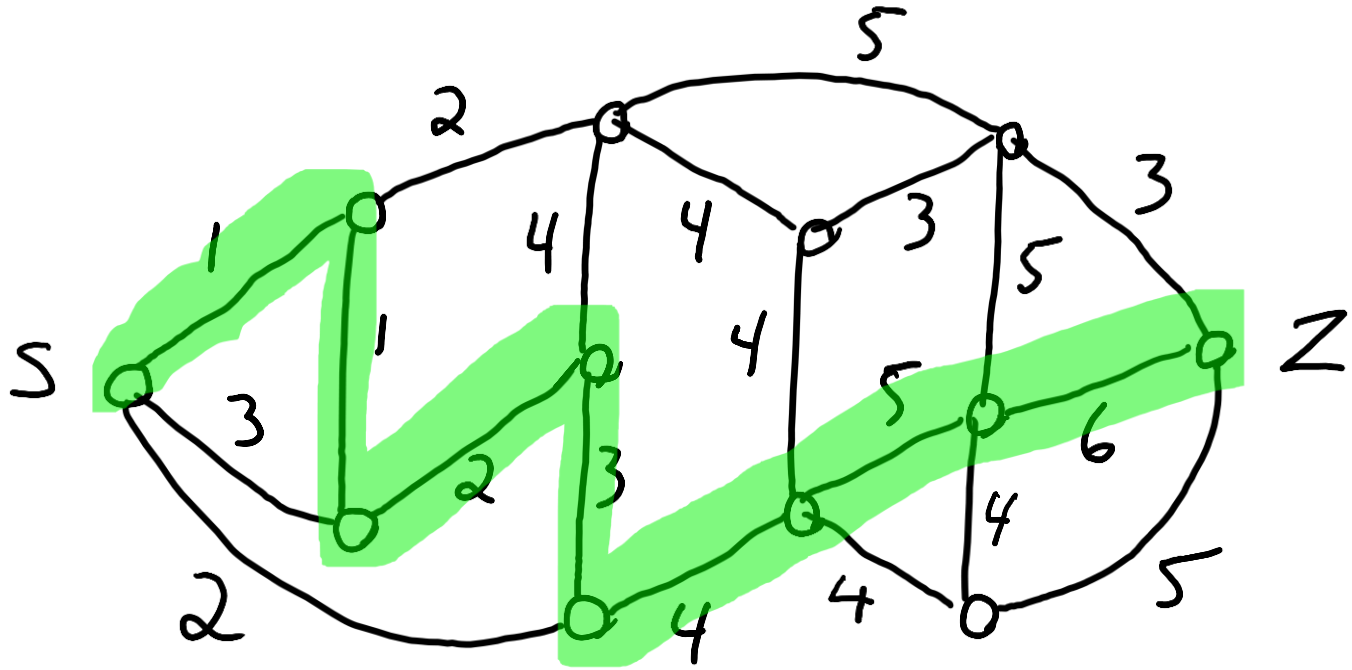
Novel Complexity Results for Temporal Separators with Deadlines

Riccardo Dondi
University of Bergamo
Manuel Lafond
Université de Sherbrooke

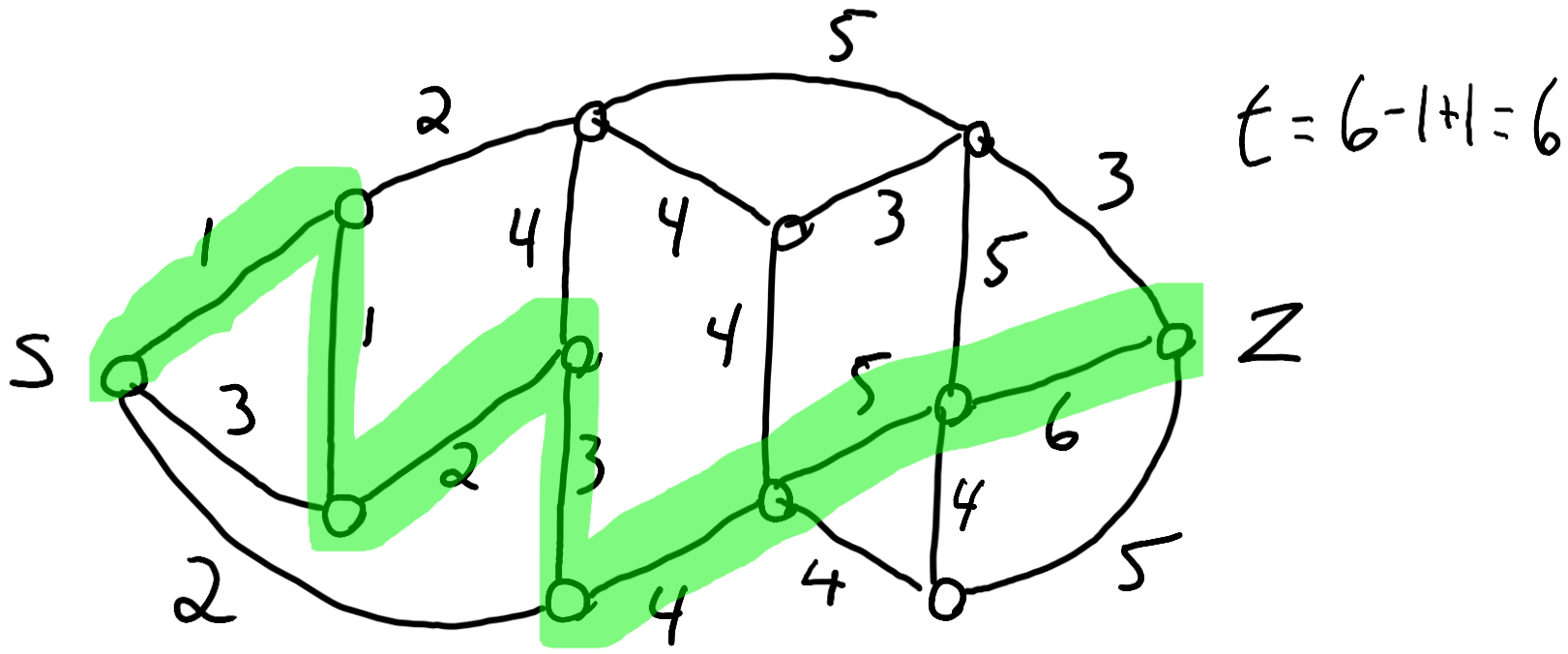
- Temporal graph
  - Undirected graph
  - Each edge is labeled by a time (an integer)



- Temporal graph
  - Undirected graph
  - Each edge is labeled by a time (an integer)
- Temporal path
  - Path with edges appearing in non-decreasing order



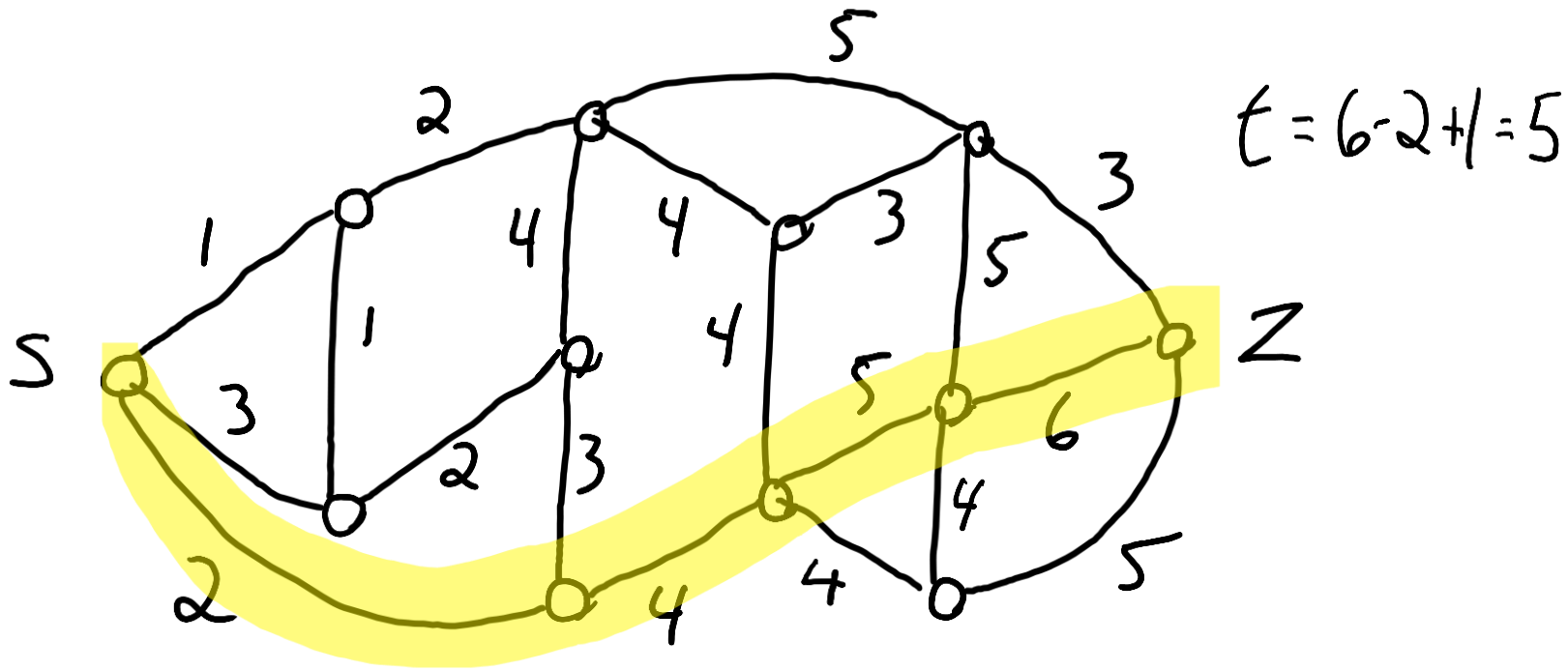
- Temporal graph
  - Undirected graph
  - Each edge is labeled by a time (an integer)
- Temporal path
  - Path with edges appearing in non-decreasing order
- Traveling time of a path
  - $Maxtime - Mintime + 1$ (min and max times on edges)



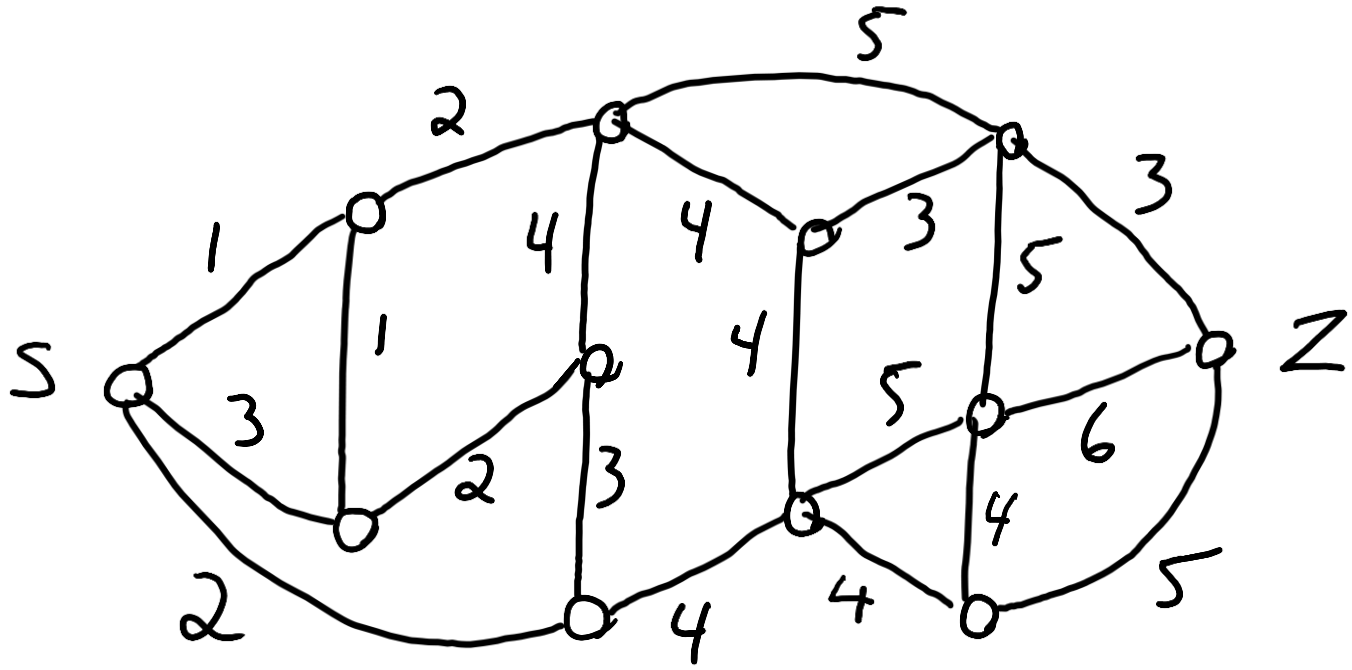
- Temporal graph
  - Undirected graph
  - Each edge is labeled by a time (an integer)
- Temporal path
  - Path with edges appearing in non-decreasing order
- Traveling time of a path
  - $Maxtime - Mintime + 1$ (min and max times on edges)

- A temporal path is ***strict*** if its edges appear in strictly increasing order.

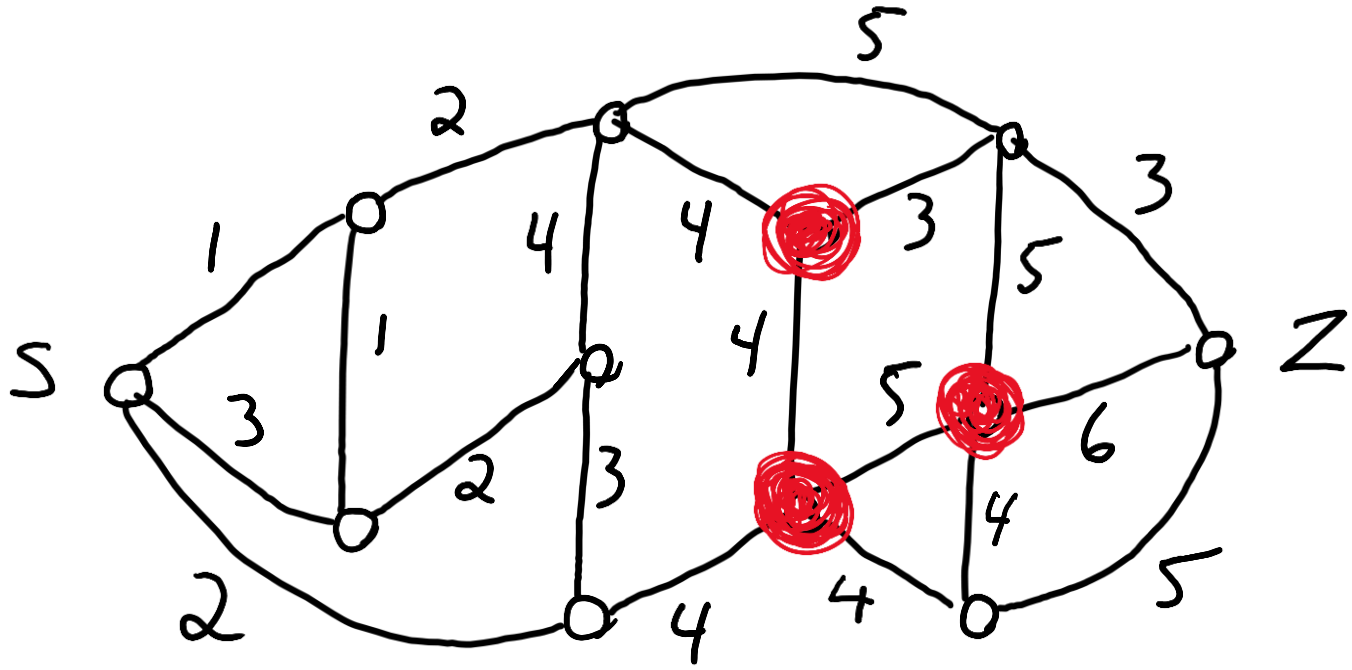
- Temporal graph
  - Undirected graph
  - Each edge is labeled by a time (an integer)
- Temporal path
  - Path with edges appearing in non-decreasing order
- Traveling time of a path
  - $Maxtime - Mintime + 1$ (min and max times on edges)



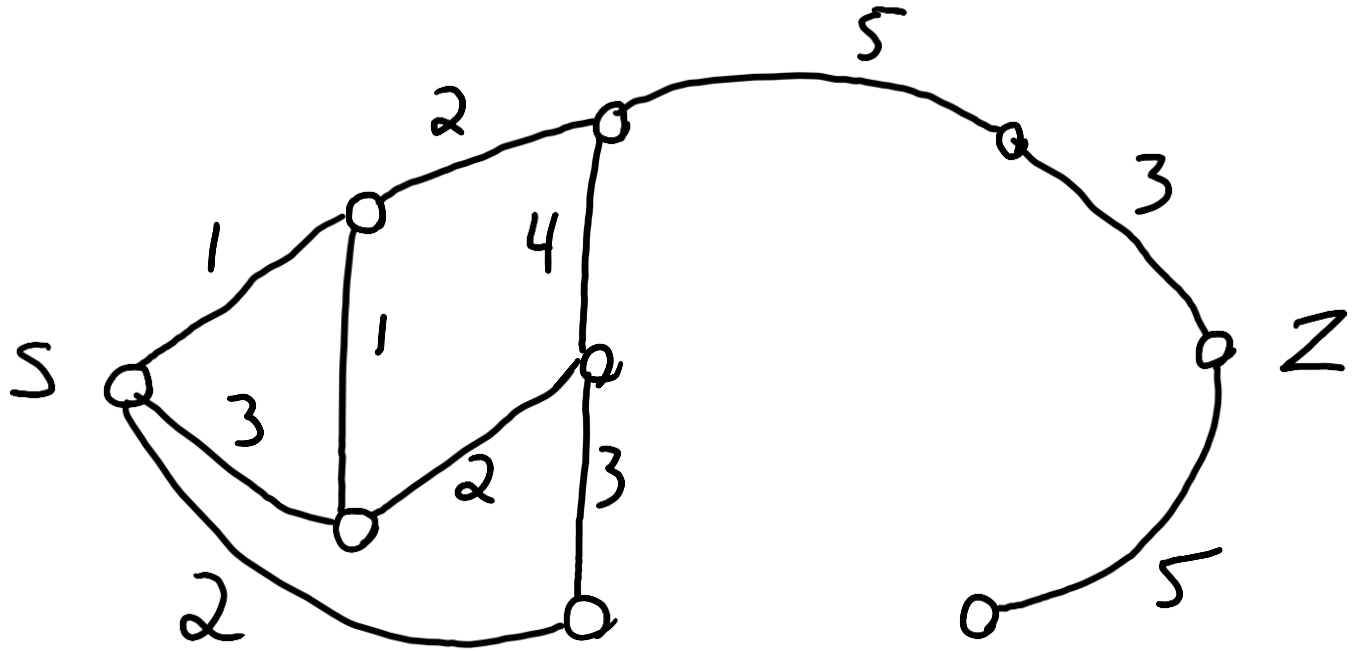
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z



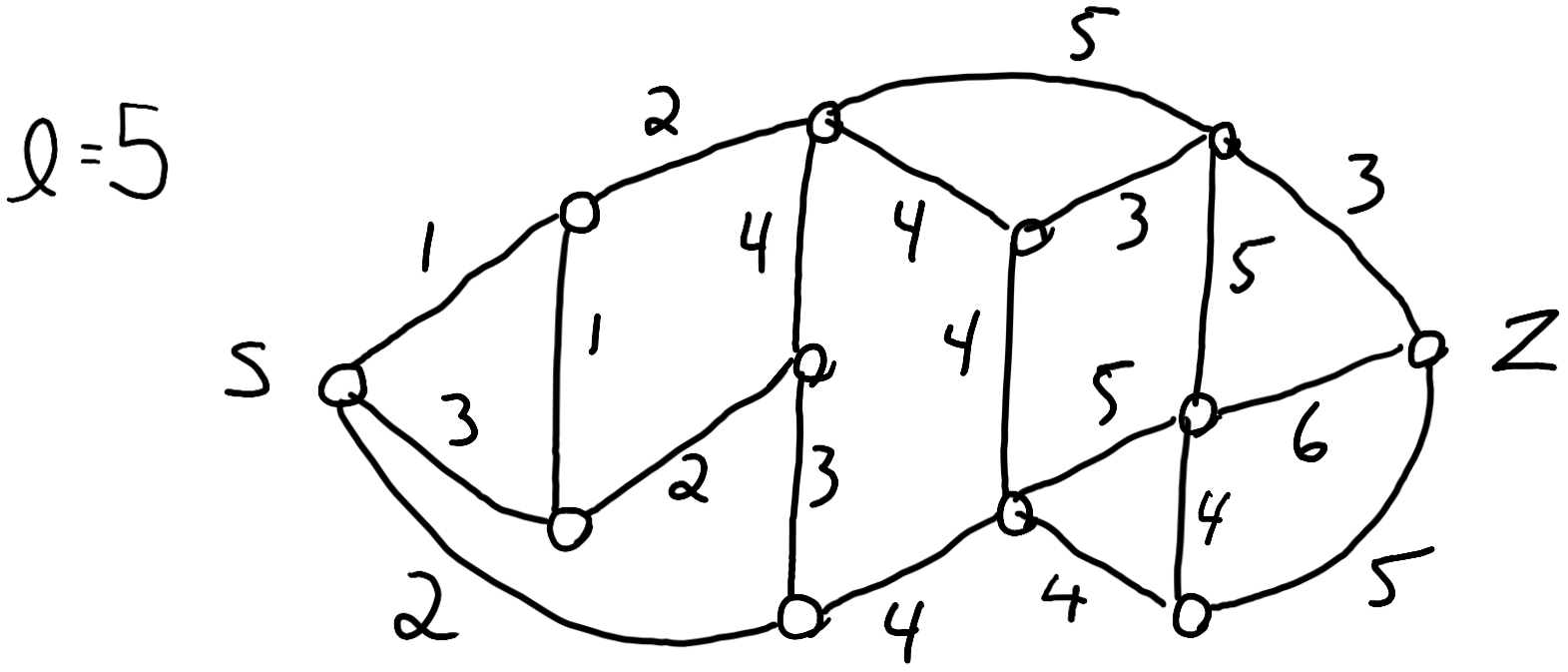
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z



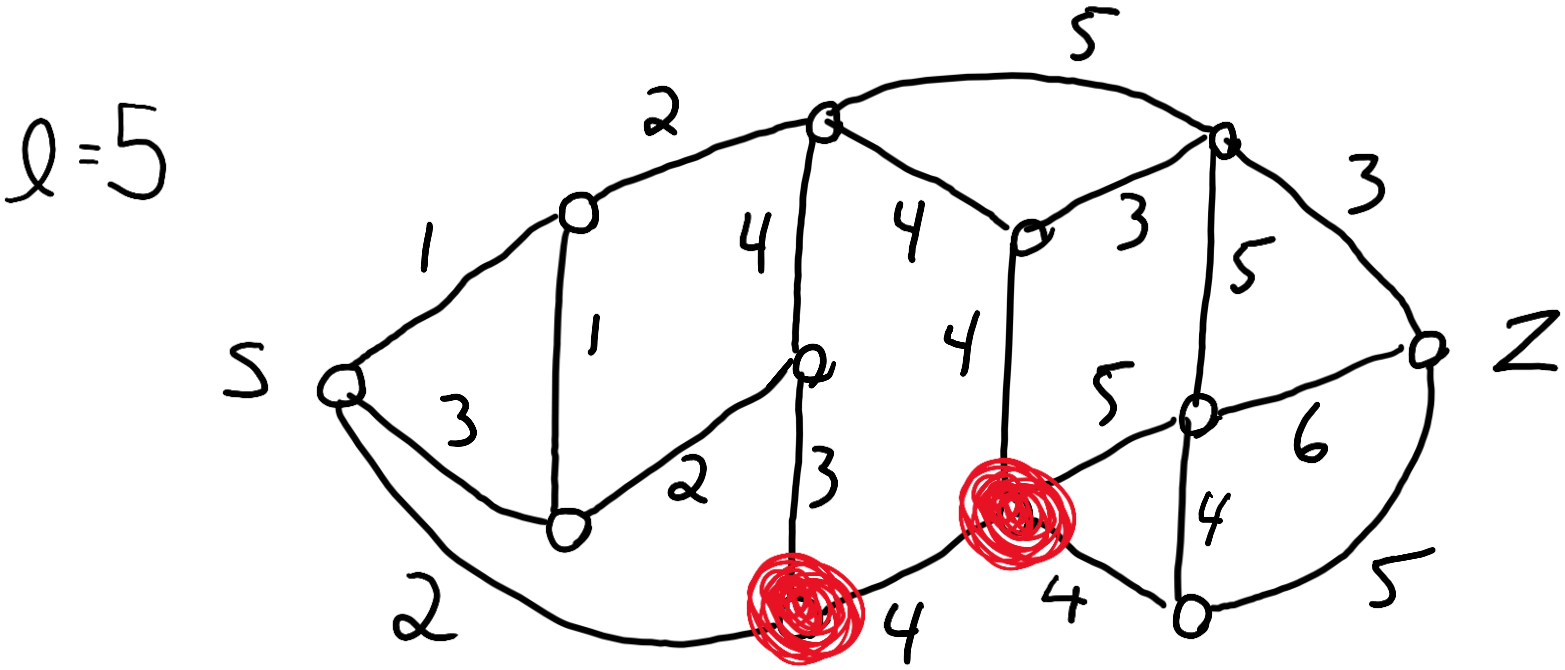
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z



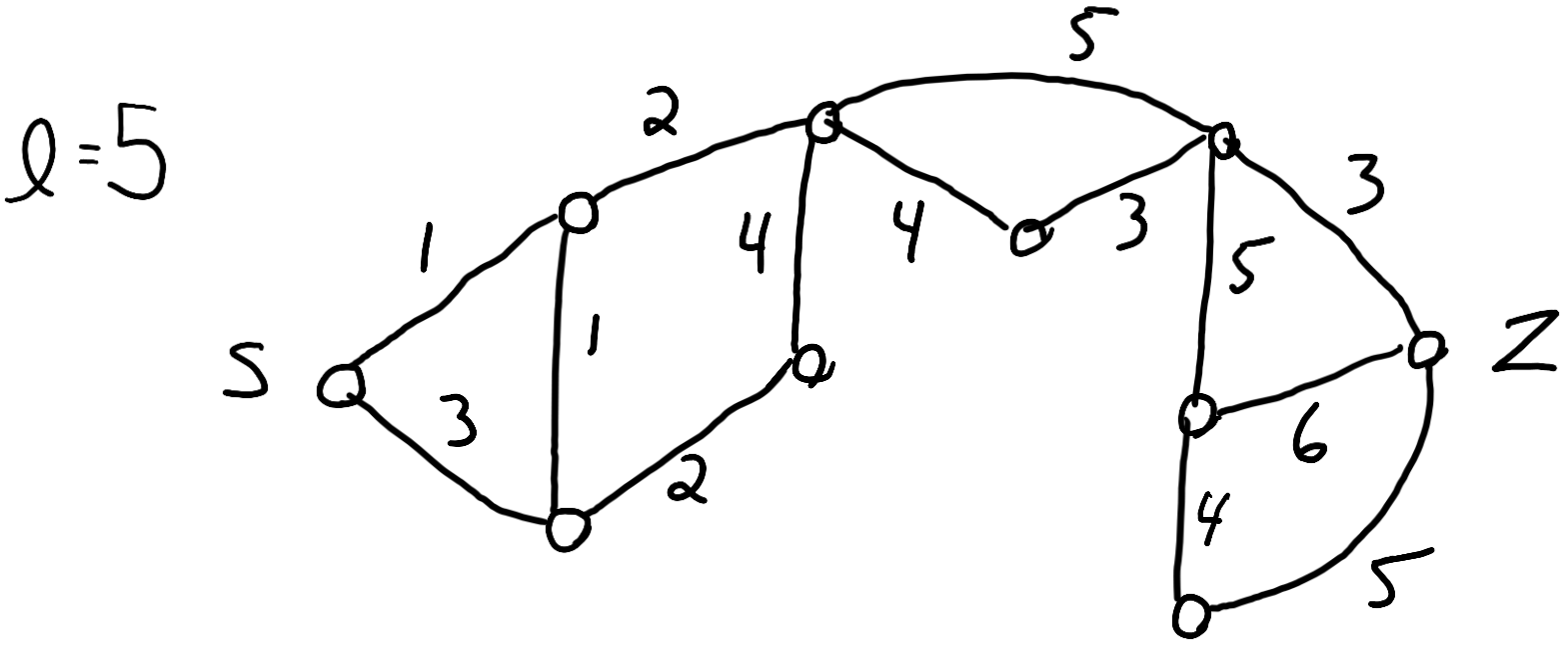
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



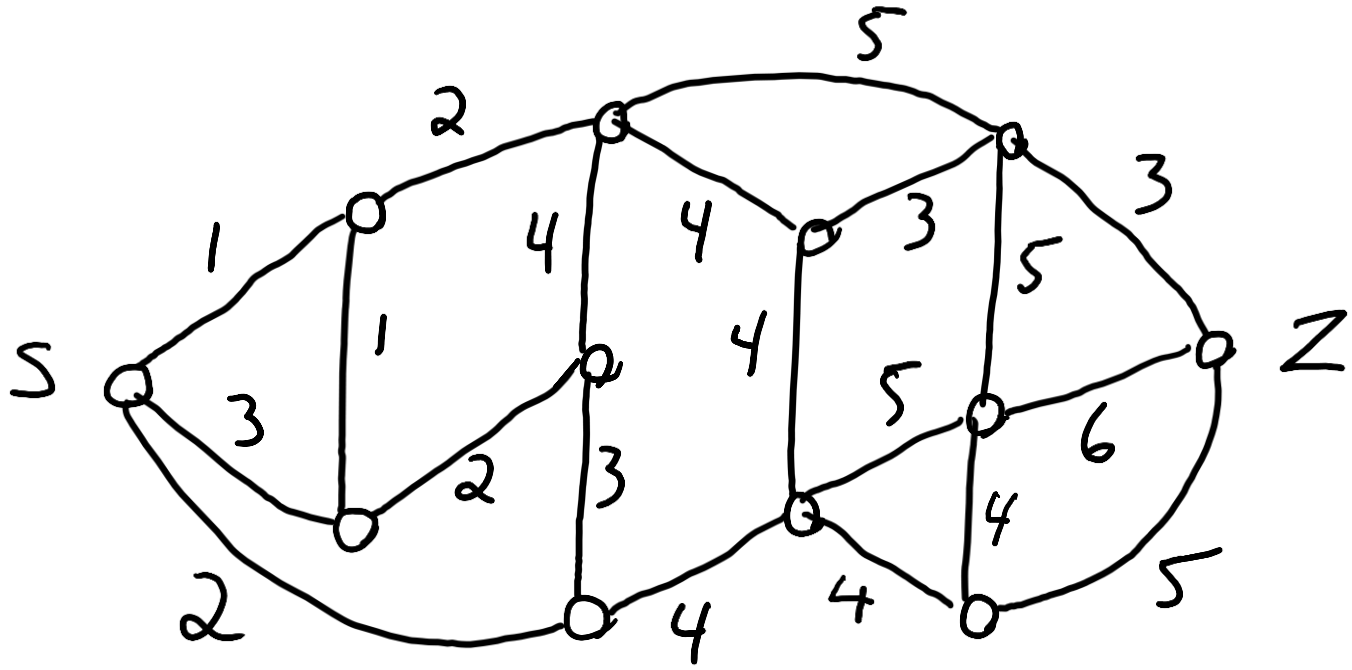
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



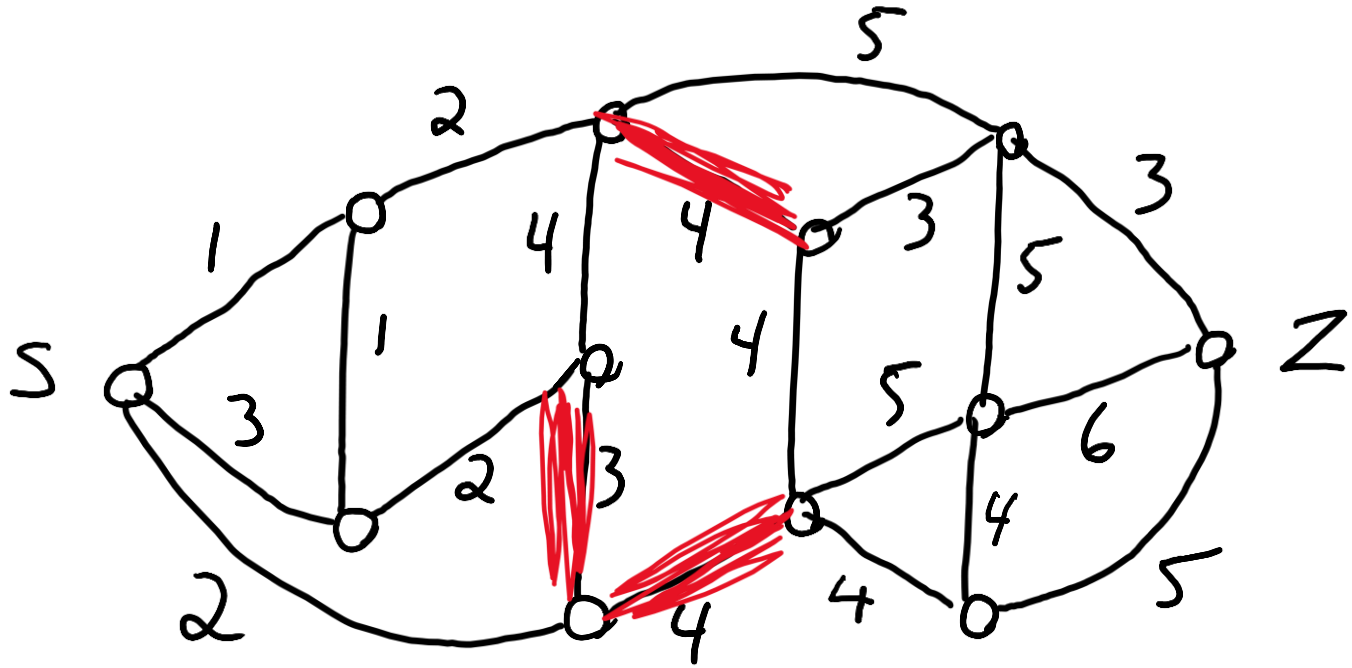
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



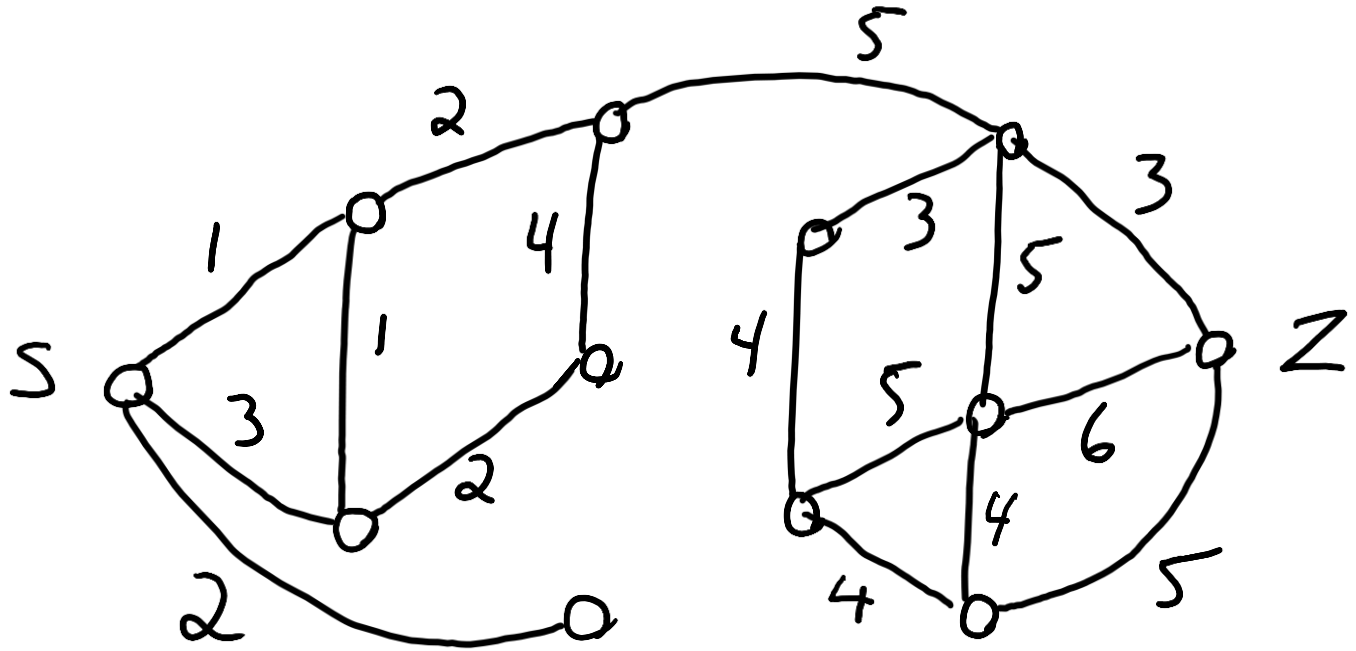
- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.
- (s, z, ℓ)-temporal cut
  - Set of edges whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.
- (s, z, ℓ)-temporal cut
  - Set of edges whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



- (s, z)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z
- (s, z, ℓ)-temporal separator
  - Set of vertices whose removal deletes all temporal paths from s to z of traveling time at most ℓ.
- (s, z, ℓ)-temporal cut
  - Set of edges whose removal deletes all temporal paths from s to z of traveling time at most ℓ.



The (s, z, ℓ)-temporal separator problem

Input: temporal graph G, vertices s, z, integer ℓ.

Find: an (s, z, ℓ)-temporal separator of minimum size.

The (s, z, ℓ)-temporal cut problem

Input: temporal graph G, vertices s, z, integer ℓ.

Find: an (s, z, ℓ)-temporal cut of minimum size.

Some past results:

- (s, z)-temporal **cut** can be solved in poly time [Berman 1996]
- (s, z)-temporal separator is NP-hard [Kempe et al. 2002]
- More result on (s, z)-temporal separator in [Zschoche et al. 2023]

Some past results:

- (s, z)-temporal **cut** can be solved in poly time [Berman 1996]
- (s, z)-temporal separator is NP-hard [Kempe et al. 2002]
- More result on (s, z)-temporal separator in [Zschoche et al. 2023]

Table 1
Overview on our results. Herein, NP-c. abbreviates NP-complete, n and $|\mathcal{E}|$ denote the number of vertices and time-edges, respectively, $G_{\downarrow}$ refers to the underlying graph of an input temporal graph $\mathcal{G}$. The temporal core is the set of vertices incident to edges that are not permanently existing (see Definition 5.1).

(s, z)-Separation		Temporal	Strict Temporal
General (Section 3)	$2 \leq \tau \leq 4$:	NP-c.* (Theorem 3.1)	$\mathcal{O}(k \cdot \lvert\mathcal{E}\rvert)$ (Theorem 3.9)
	$5 \leq \tau$:	NP-c.* (Theorem 3.1)	NP-c.* (Theorem 3.1)
Planar $G_{\downarrow}$ (Section 4)	τ unbounded:	NP-c. (Theorem 4.1)	NP-c. (Theorem 4.1)
	τ constant:	*open*	$\mathcal{O}(\lvert\mathcal{E}\rvert \log \lvert\mathcal{E}\rvert)$ (Proposition 4.5)
Constant-size temporal core: (Section 5)		$n^{\mathcal{O}(1)} + \mathcal{O}(\lvert\mathcal{E}\rvert)$ (Theorem 5.2)	NP-c.* (Theorem 3.1)

* Theorem 3.1 also implies W[1]-hardness wrt. k.

(s, z, ℓ)-temporal separator introduced in [Harutyunyan, Koupayi, Pankratov 2023]

(s, z, ℓ)-temporal separator introduced in [Harutyunyan, Koupayi, Pankratov 2023]

- NP-hard even if $\ell = 1$
- Admits a τ^2 approximation, where $\tau =$ max edge time in G
- No $O(\log(n) + \log(\tau))$ approx under standard assumptions
- In P if $G - \{s, z\}$ is a tree or if G has branchwidth 2

(s, z, ℓ)-temporal separator introduced in [Harutyunyan, Koupayi, Pankratov 2023]

- NP-hard even if $\ell = 1$
- Admits a τ^2 approximation, where $\tau =$ max edge time in G
- No $O(\log(n) + \log(\tau))$ approx under standard assumptions
- In P if $G - \{s, z\}$ is a tree or if G has branchwidth 2
- **OPEN PROBLEM**: is (s, z, ℓ)-temporal separator in P on graphs of constant pathwidth?
  - If yes, would settle complexity on a Discrete Segment Covering (DISC-SC) problem.

Our results:

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

It is, however, in P on graphs of pathwidth 3 or less, for any ℓ.

Our results:

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

It is, however, in P on graphs of pathwidth 3 or less, for any ℓ.

Theorem: (s, z, ℓ)-temporal separator admits no $2^{\Omega(log^{1-\epsilon}(n))}$approx. unless NP⊆ZPP.

Our results:

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

It is, however, in P on graphs of pathwidth 3 or less, for any ℓ.

Theorem: (s, z, ℓ)-temporal separator admits no $2^{\Omega(log^{1-\epsilon}(n))}$approx. unless NP⊆ZPP.

Theorem: The strict variant of (s, z, ℓ)-temporal separator has an $\ell - 1$ approx., tight under the UGC.

Our results:

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

It is, however, in P on graphs of pathwidth 3 or less, for any ℓ.

Theorem: (s, z, ℓ)-temporal separator admits no $2^{\Omega(log^{1-\epsilon}(n))}$approx. unless NP⊆ZPP.

Theorem: The strict variant of (s, z, ℓ)-temporal separator has an $\ell - 1$ approx., tight under the UGC.

Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard, for all $\ell \geq 2$. It admits a $O(\log \ell)$-approx.

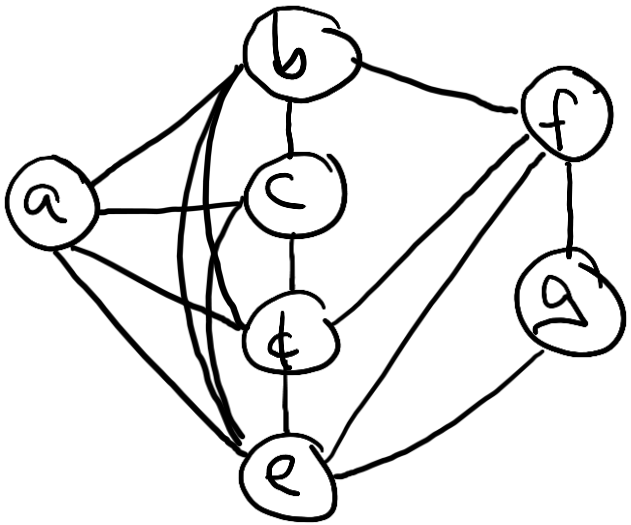
- Contrasts with (s, z)-temporal cut, which is in P.

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

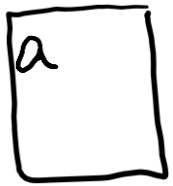
such that there are **always at most $k + 1$ vertices in memory**.

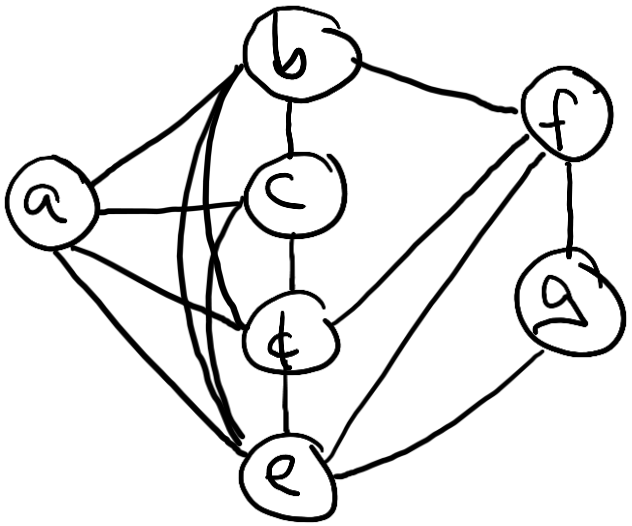


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

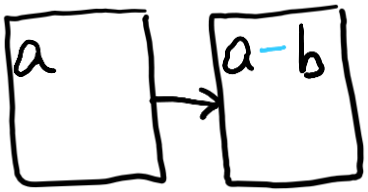


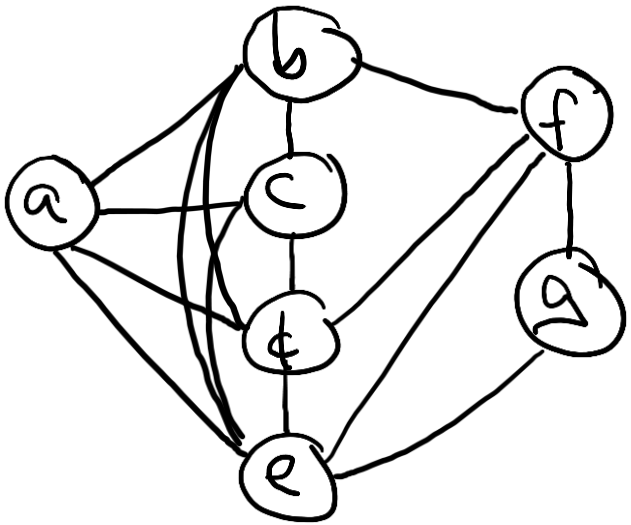


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

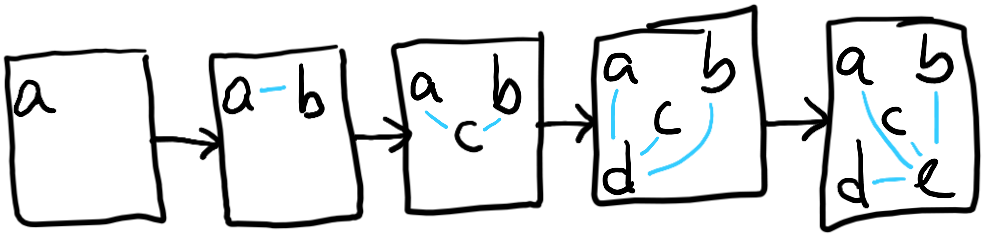


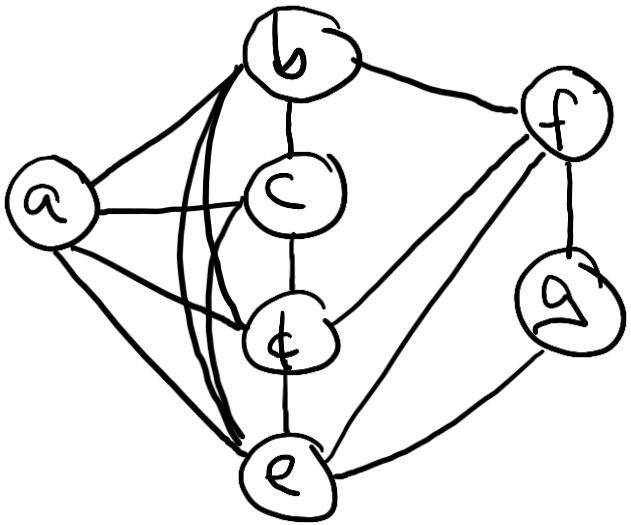


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

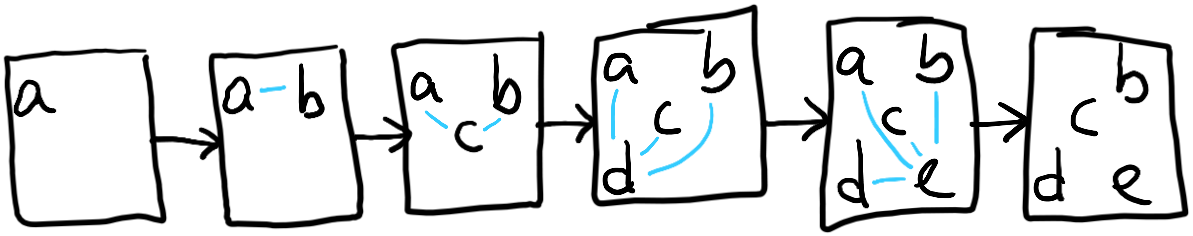


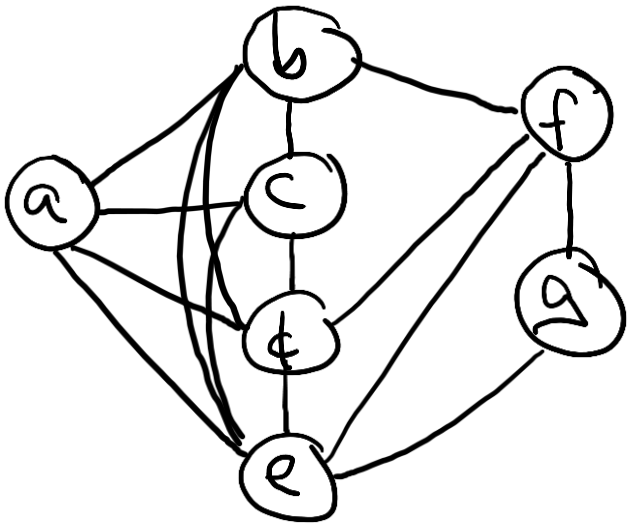


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

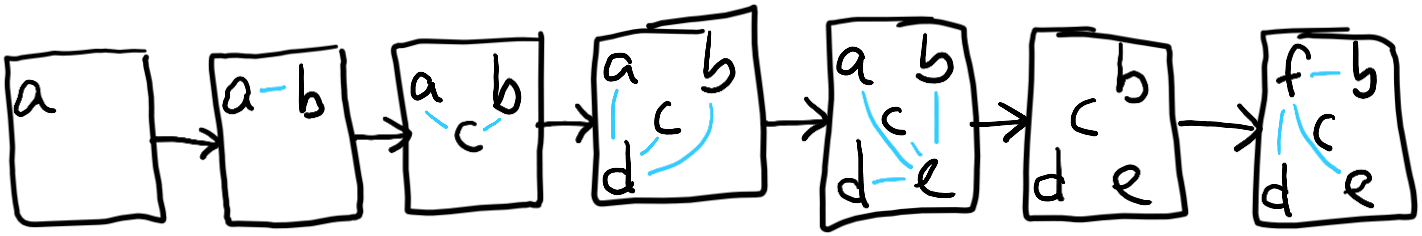


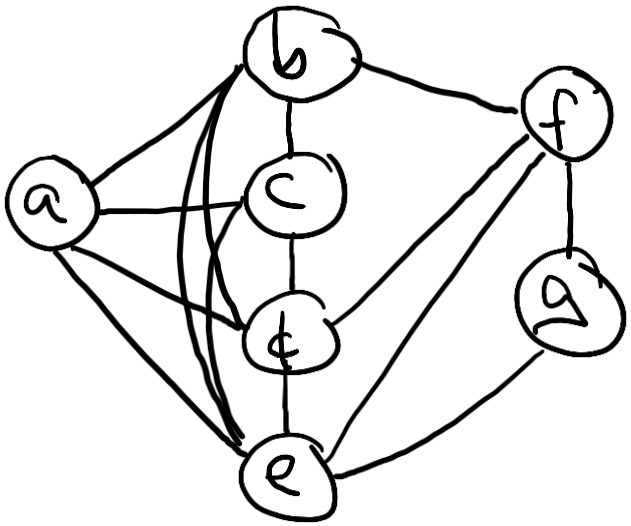


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

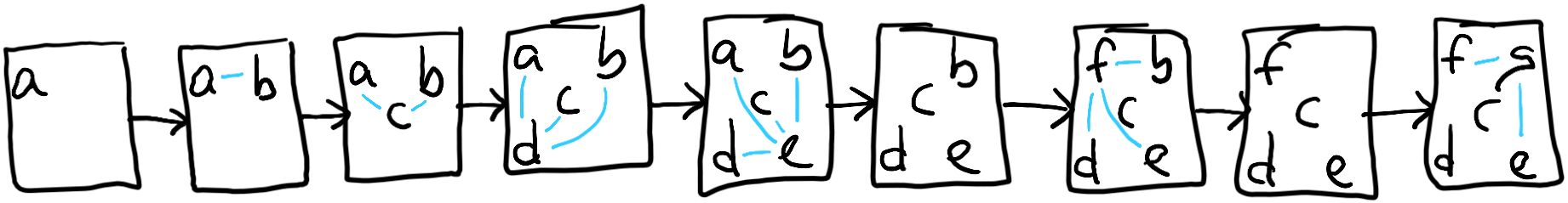


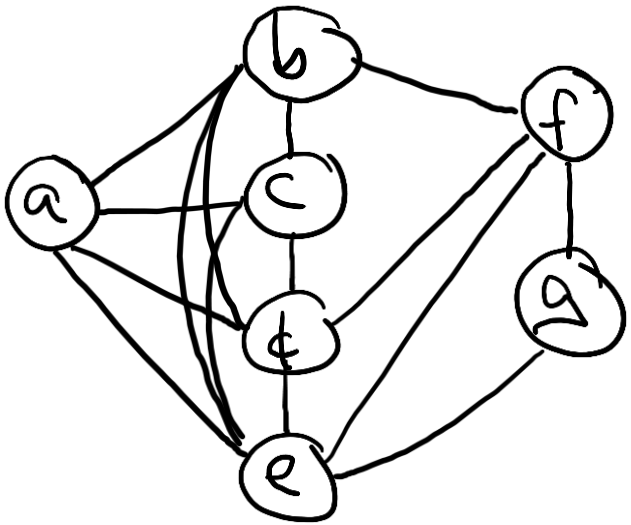


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.

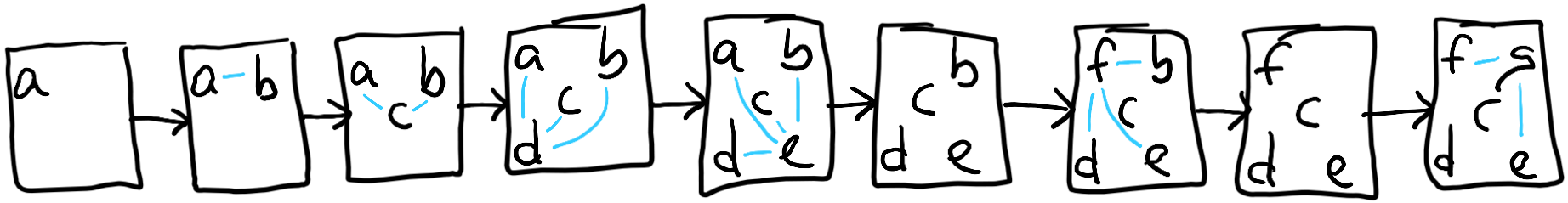


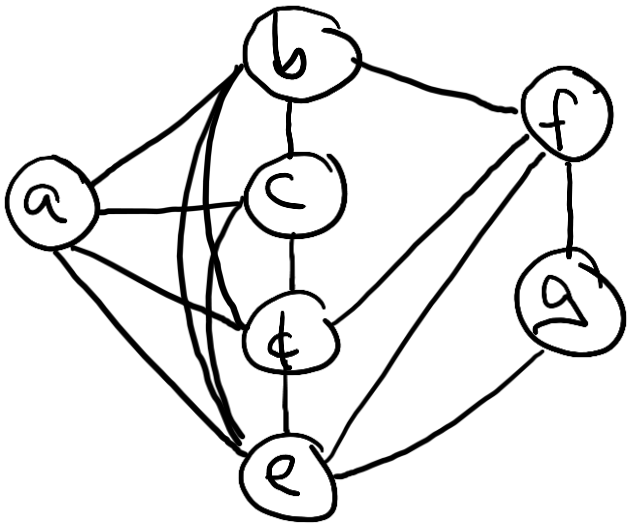


A graph G has **pathwidth k** or less if it can be built using the following computer instructions:

- create a vertex and store it in memory;
- add an edge between two vertices stored in memory;
- unload a vertex from memory (without ever loading it again);

such that there are **always at most $k + 1$ vertices in memory**.





Need 5 memory

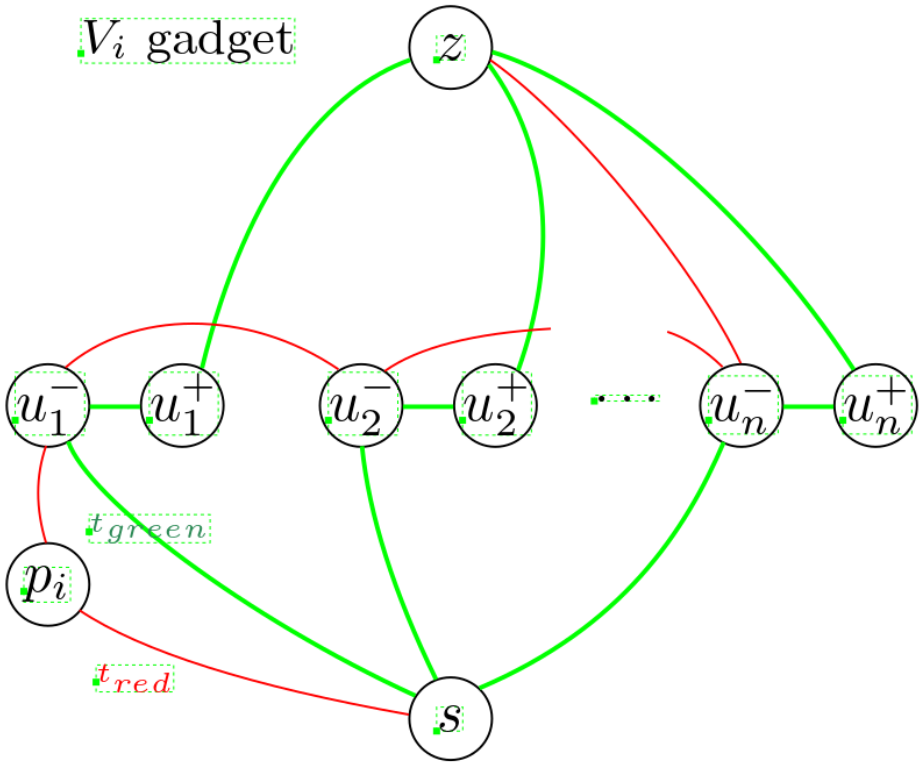
$\Rightarrow pw = 4$

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

Reminder: ℓ is the threshold of travel time.

$\ell = 1$: view each edge as colored
destroy monochromatic paths



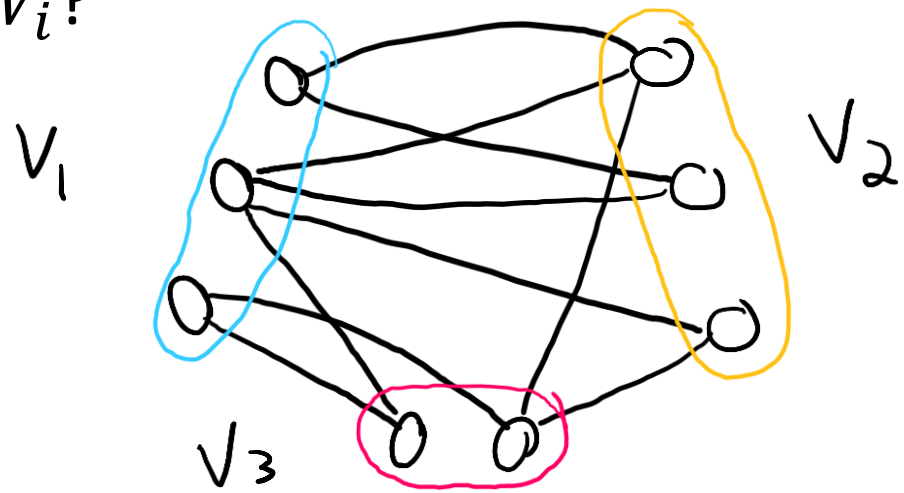
Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

Reduction from Multicolored Independent Set

Multicolored Independent Set (MulIS)

Input: graph $G_C = (V_1 \cup \cdots \cup V_k, E_C)$

Question: is there an independent set with one vertex per V_i?



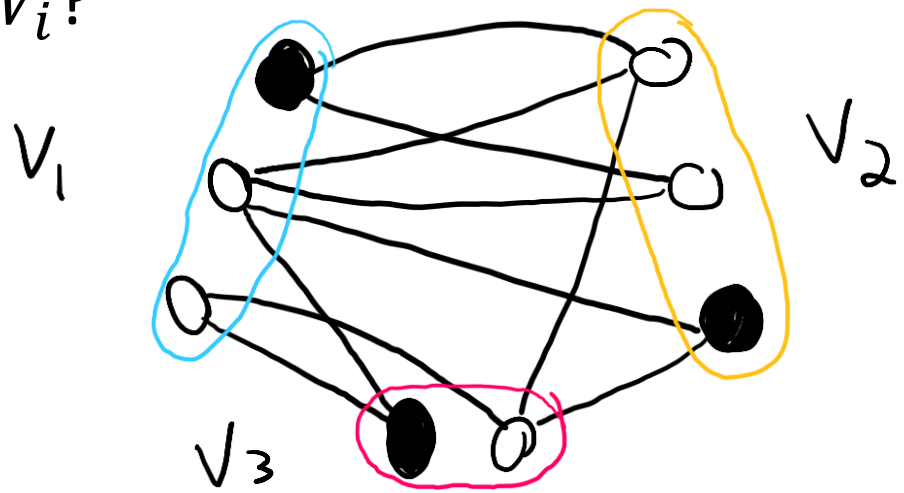
Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

Reduction from Multicolored Independent Set

Multicolored Independent Set (MulIS)

Input: graph $G_C = (V_1 \cup \cdots \cup V_k, E_C)$

Question: is there an independent set with one vertex per V_i?

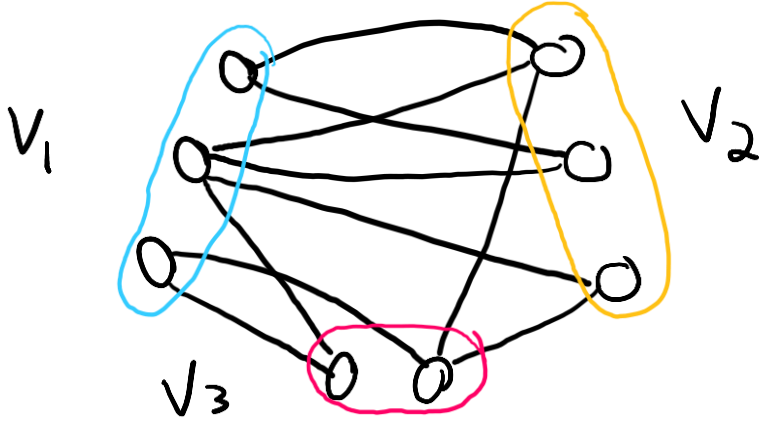


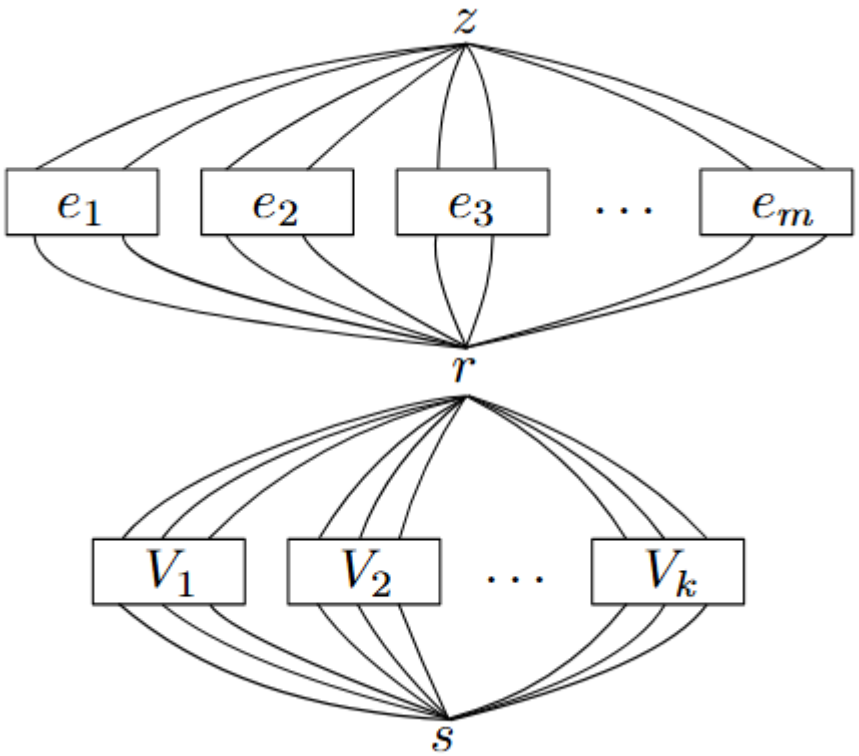
Multicolored Independent Set (MulIS)

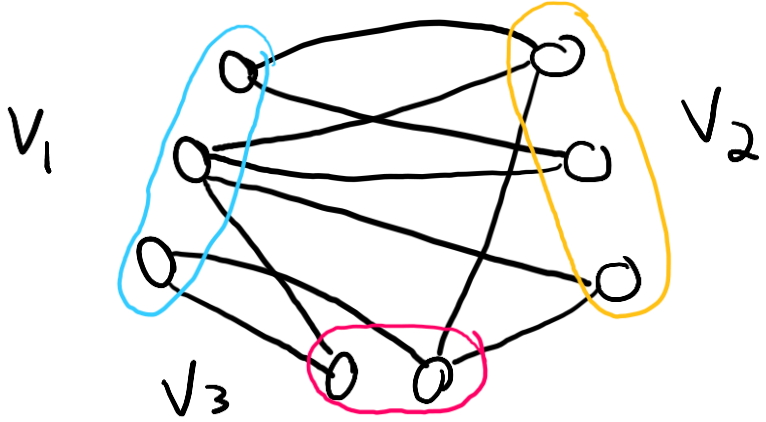
Input: graph $G_C = (V_1 \cup \cdots \cup V_k, E_C)$

Question: is there an independent set with one vertex per V_i?

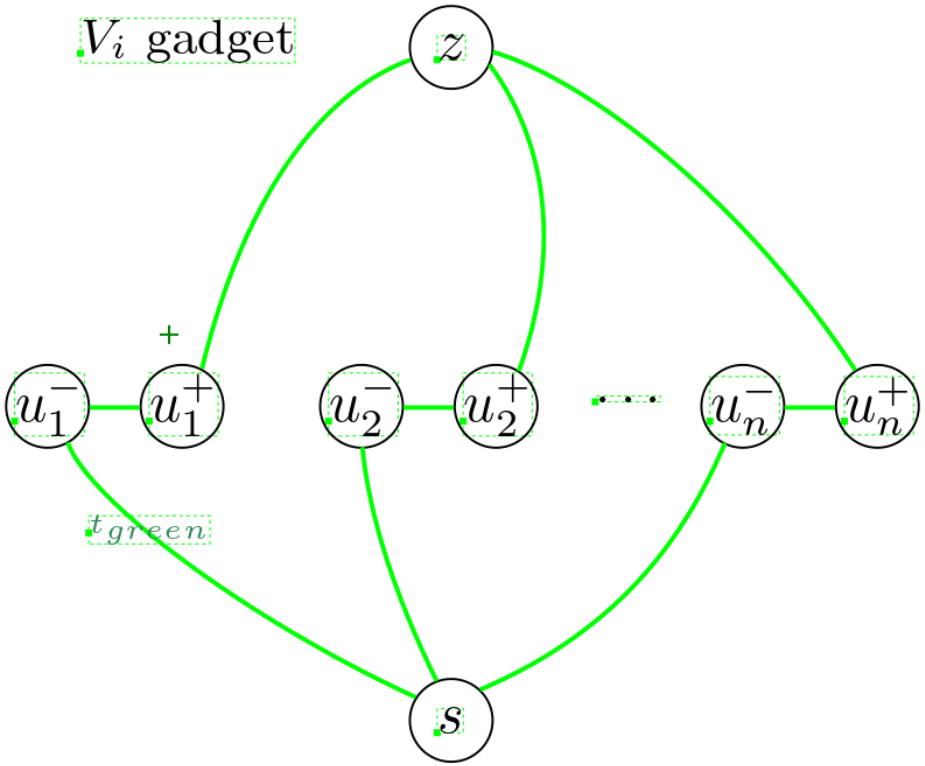
Goal: from MulIS instance G_C, create temporal graph G of pathwidth 4, such that G_C has a MulIS iff G has an $(s, z, 1)$-separator of size $|V(G_C)| + |E(G_C)|$.

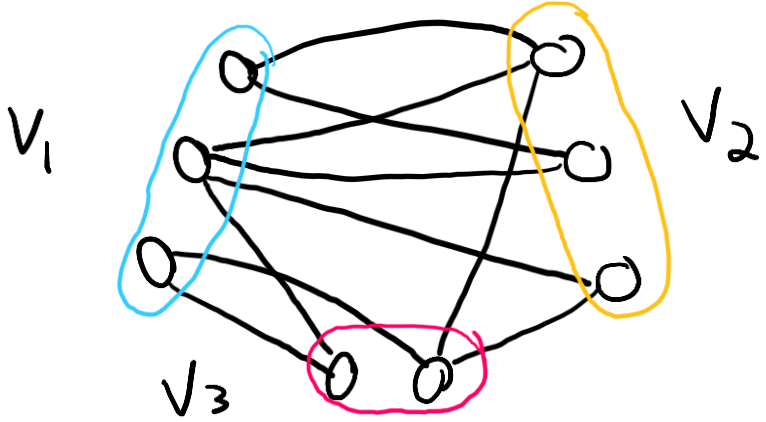

V_1
V_2
V_3


z
e_1
e_2
e_3
$\cdots$
e_m
r
V_1
V_2
$\cdots$
V_k
s



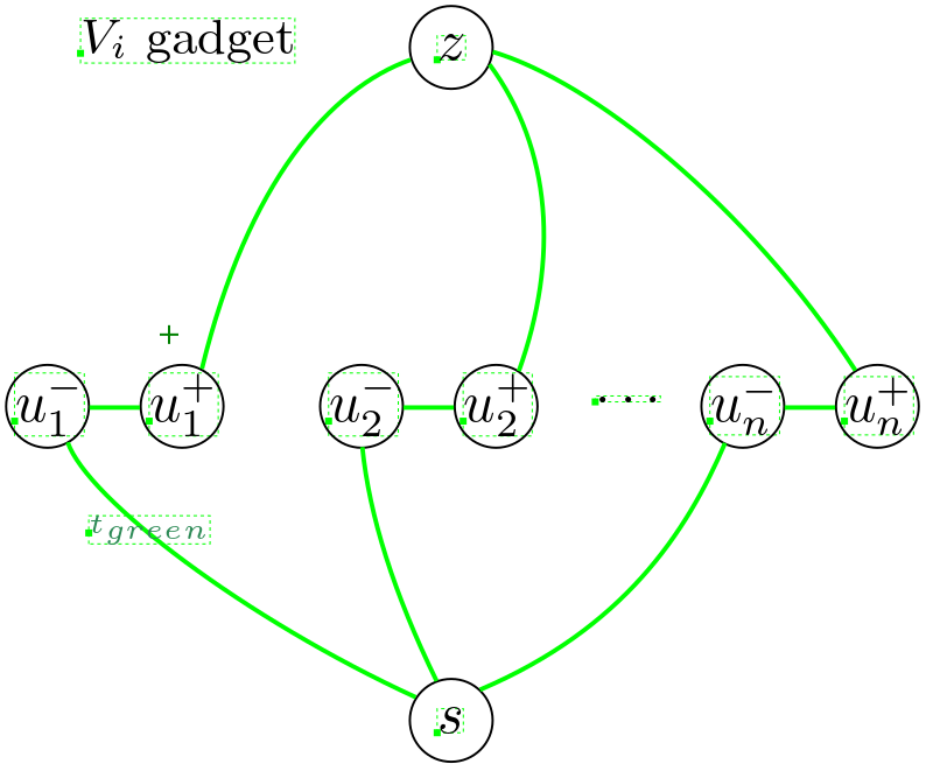
For each color class V_i and each $u \in V_i$, add vertices u^+, u^- and a path of travel time $\ell = 1$ enforcing us to delete one of the two.

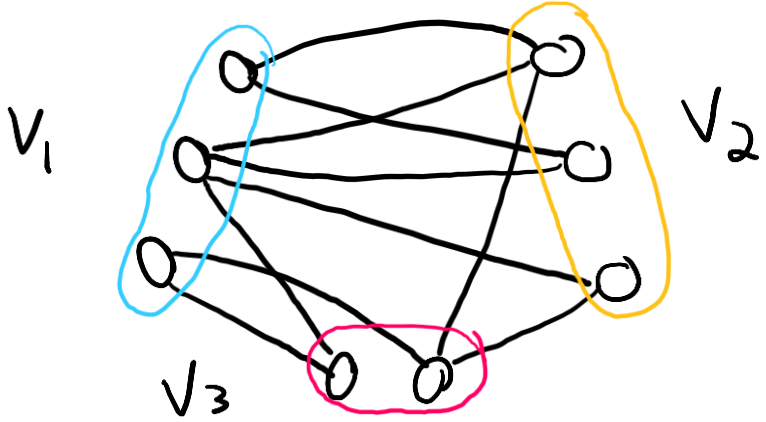




For each color class V_i and each $u \in V_i$, add vertices u^+, u^- and a path of travel time $\ell = 1$ enforcing us to delete one of the two.

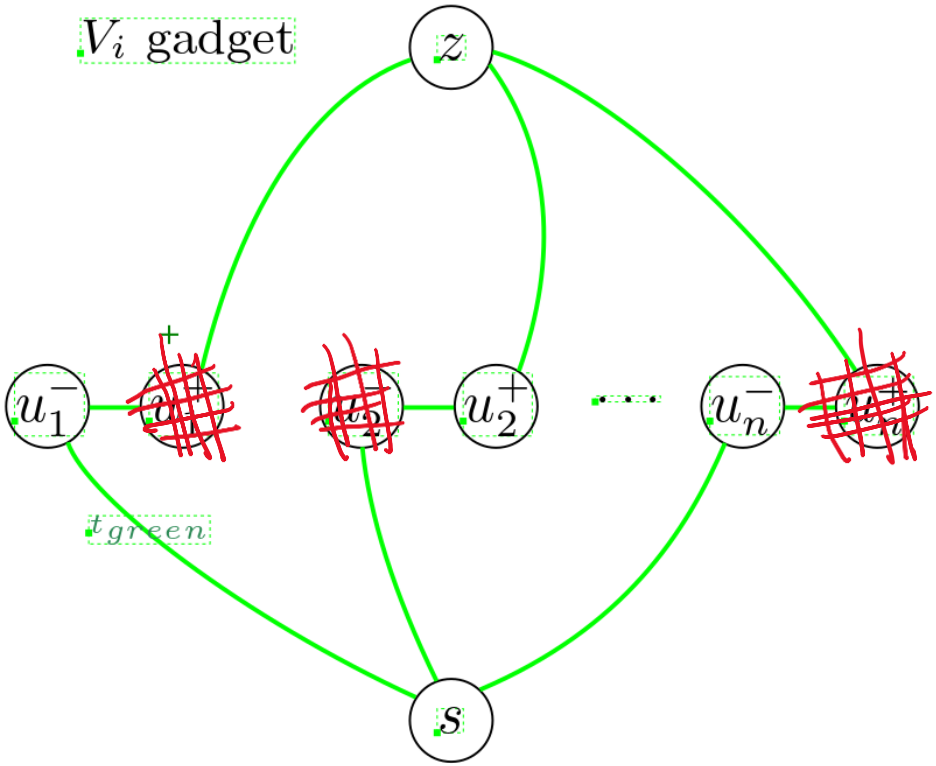
Deleting u^- and keeping u^+ means "I want u in the MulIS".

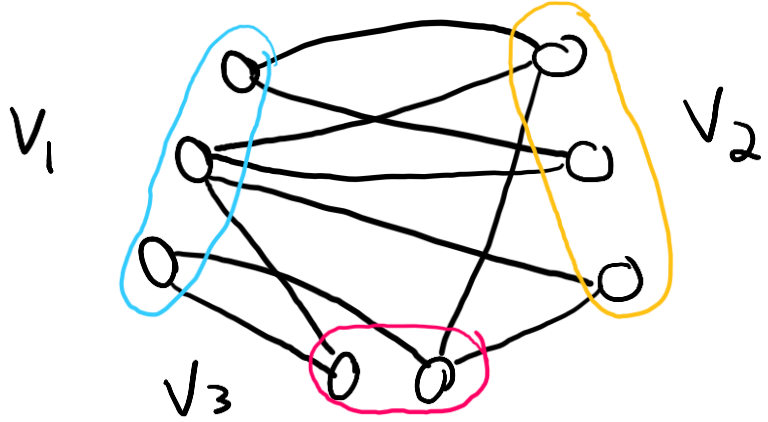




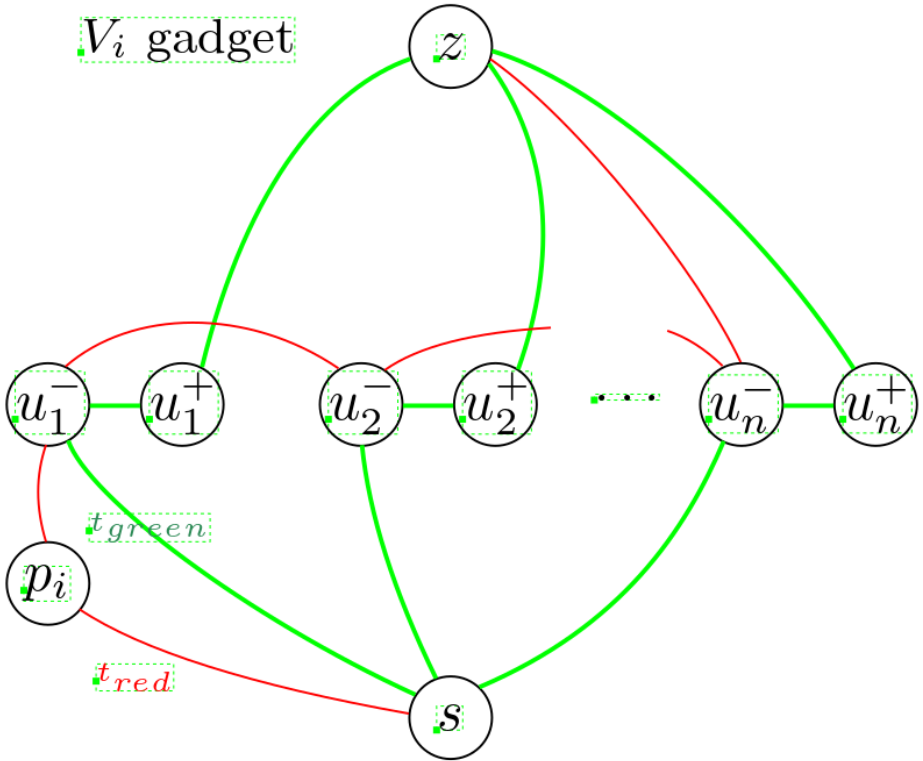
For each color class V_i and each $u \in V_i$, add vertices u^+, u^- and a path of travel time $\ell = 1$ enforcing us to delete one of the two.

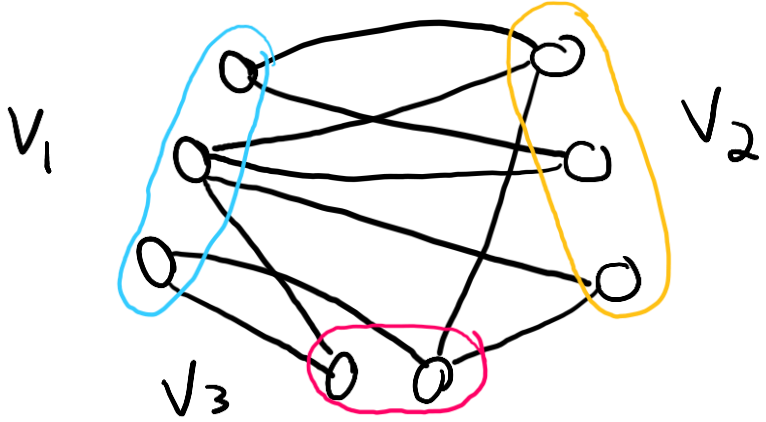
Deleting u^- and keeping u^+ means "I want u in the MulIS".



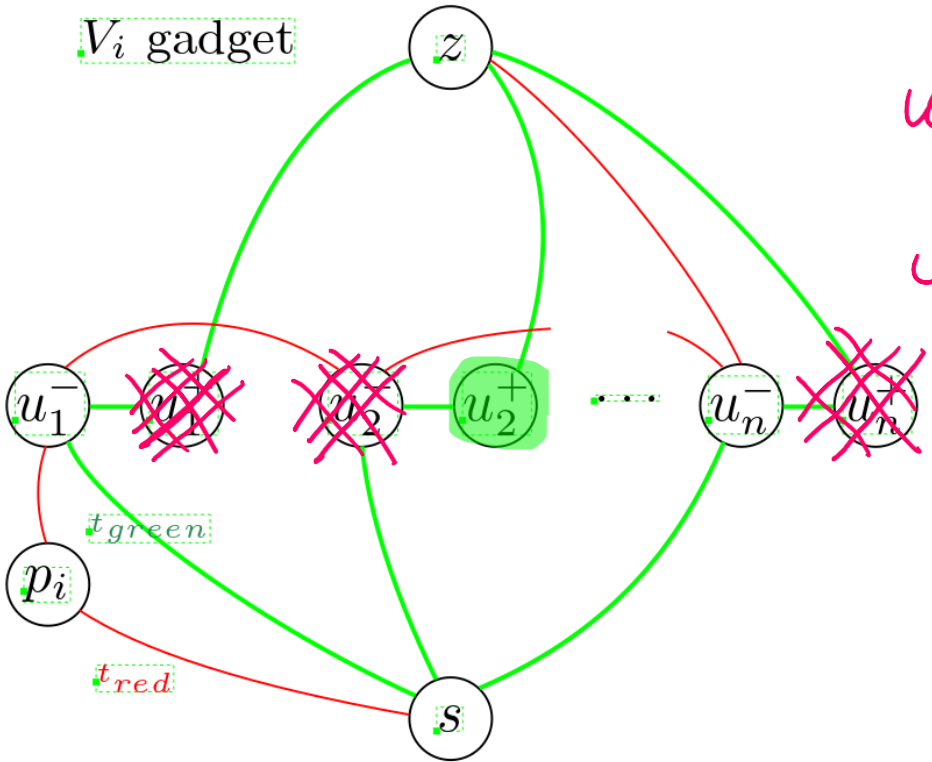


Add a path of travel time $\ell = 1$ between the u^-, enforcing the deletion of one (ignore p_i).





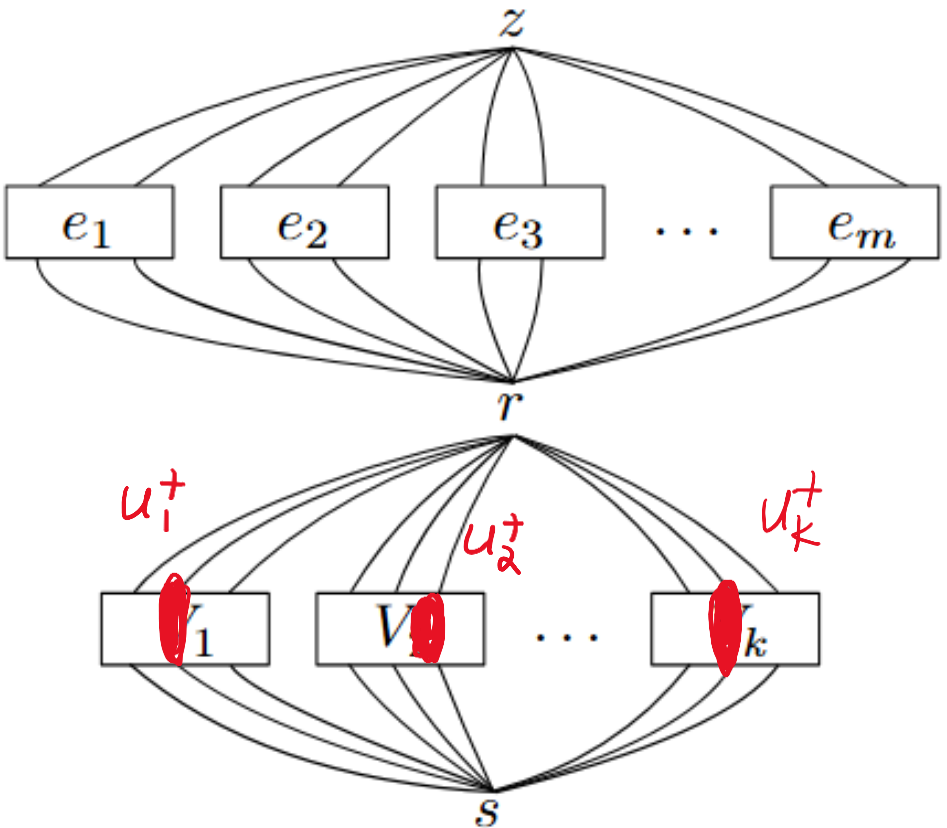
Add a path of travel time $\ell = 1$ between the u^-, enforcing the deletion of one (ignore p_i).

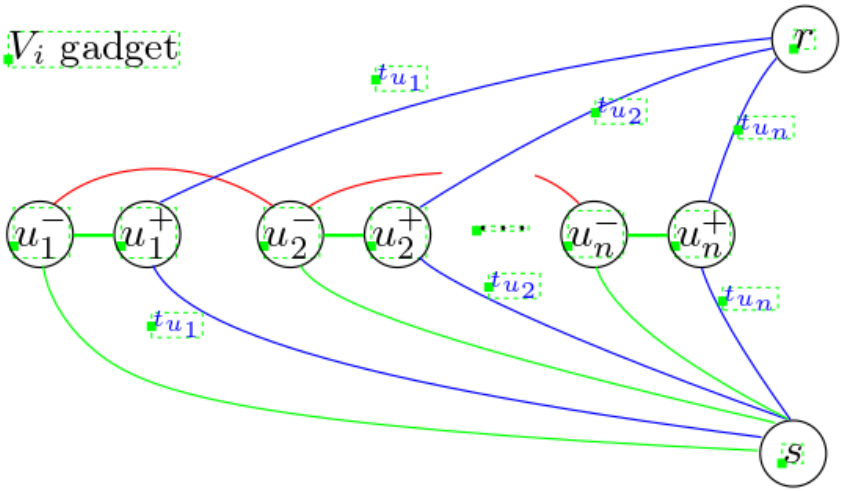


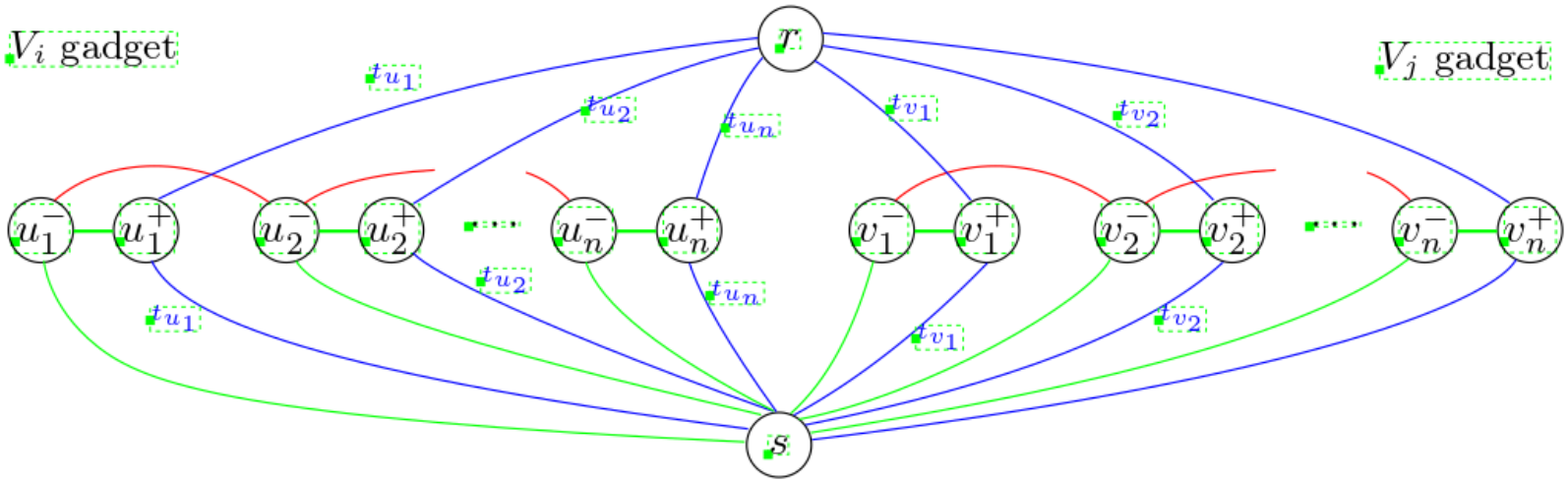
u_2^+ is kept
$\Rightarrow$
u_2 chosen in indset

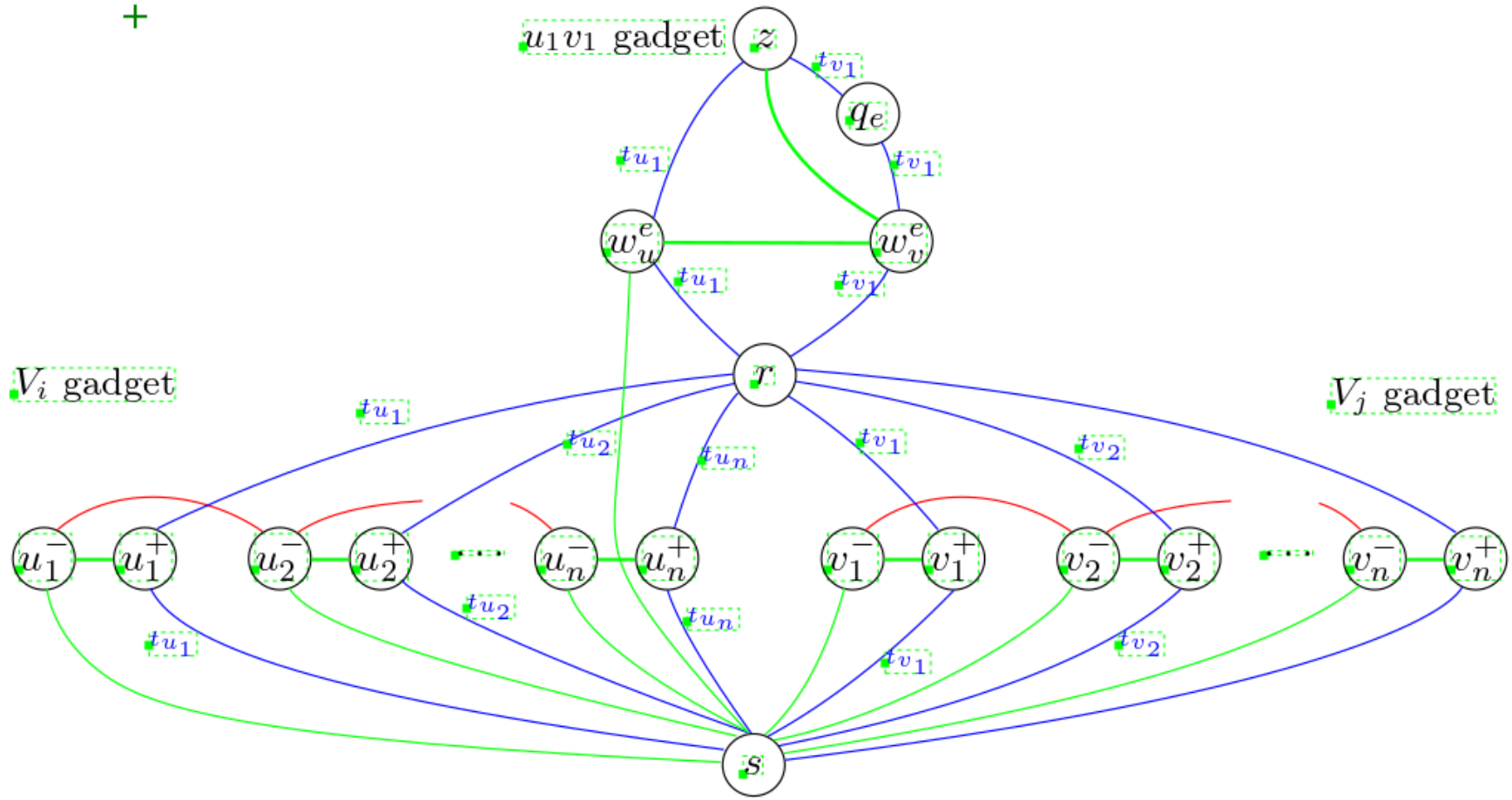
New vertex r: each chosen vertex u has a path of travel time $\ell = 1$ to this r, with its specific time t_u.

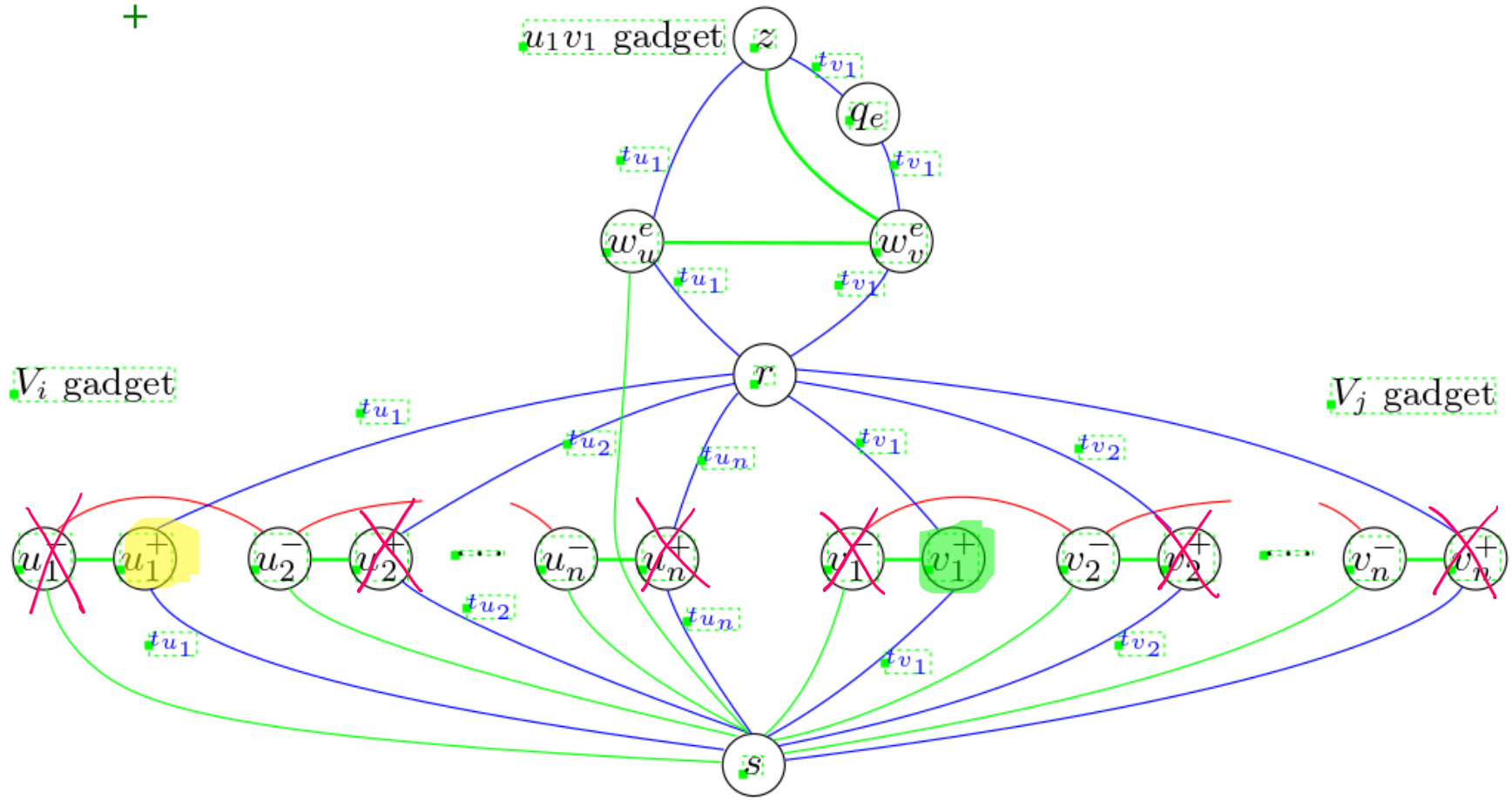
Add an edge verification gadget. Designed so that we must delete one vertex, but if $uv \in E$ and we chose both u, v we must delete two.



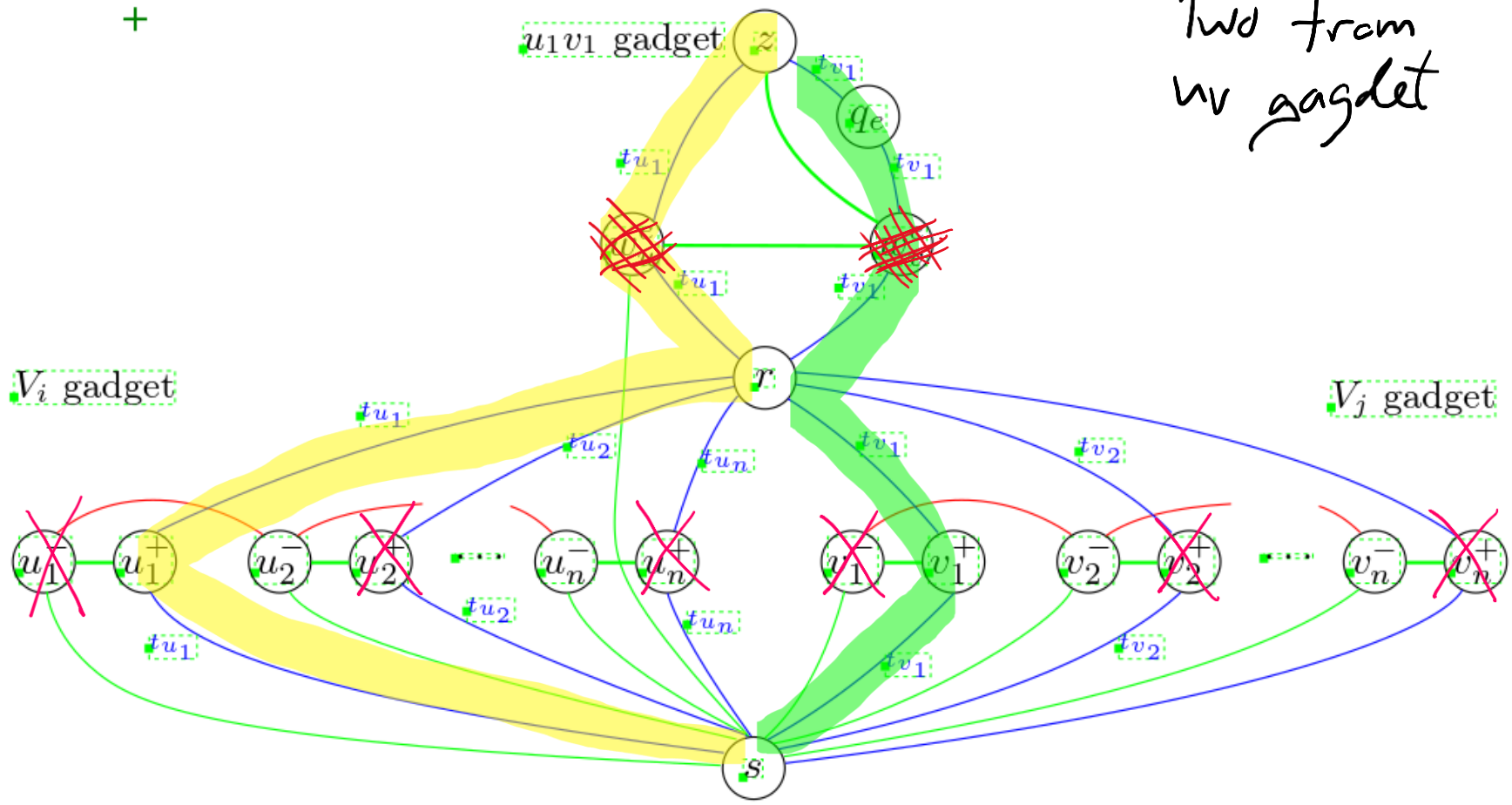

V_i gadget
r
t_{u_1}
t_{u_2}
t_{u_n}
u_1^-
u_1^+
u_2^-
u_2^+
u_n^-
u_n^+
t_{u_2}
t_{u_n}
t_{u_1}
s


V_i gadget
V_j gadget
r
s
u_1^-
u_1^+
u_2^-
u_2^+
u_n^-
u_n^+
v_1^-
v_1^+
v_2^-
v_2^+
v_n^-
v_n^+
t_{u_1}
t_{u_2}
t_{u_n}
t_{v_1}
t_{v_2}

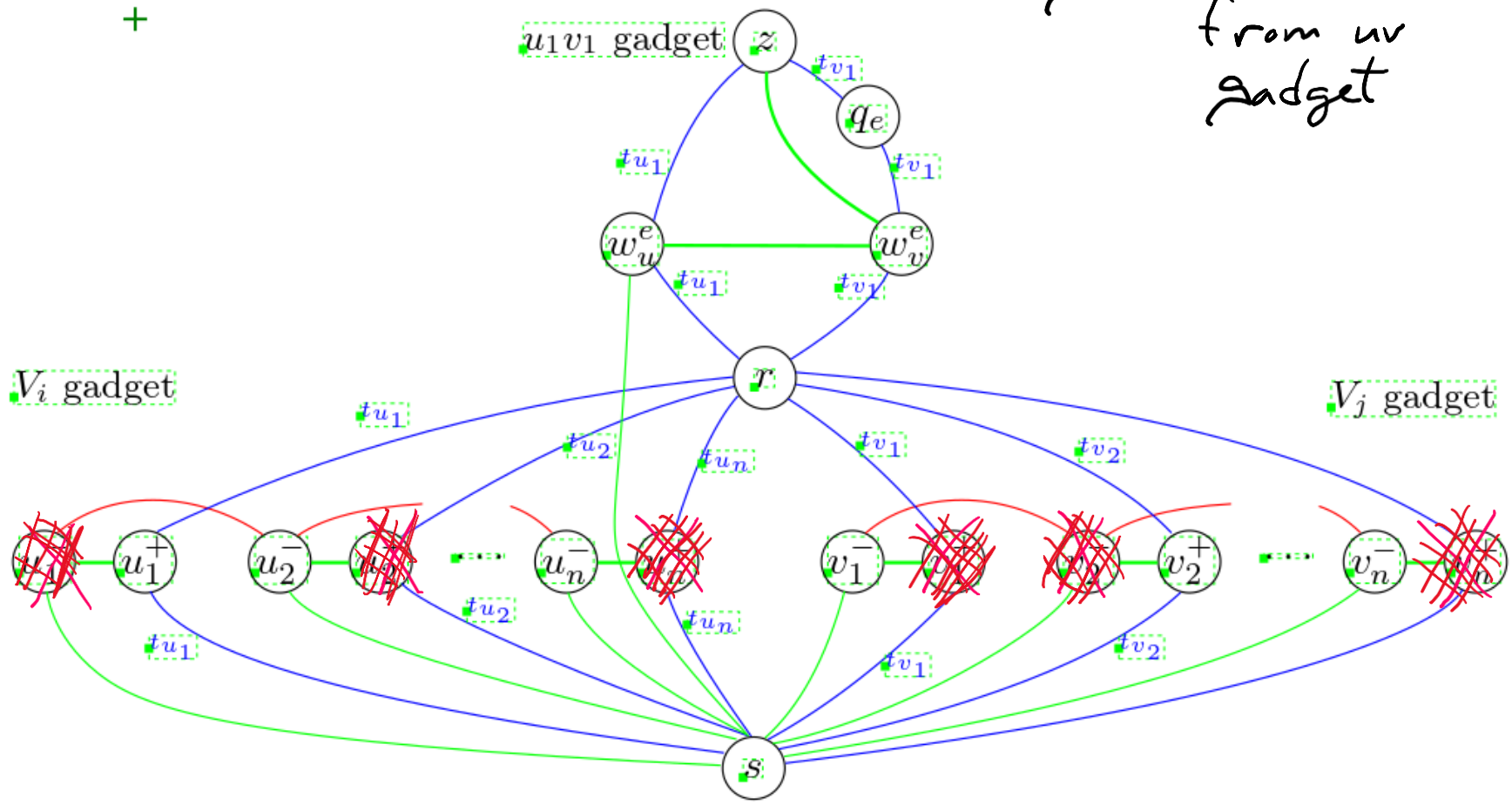

u_1v_1 gadget
z
t_{v_1}
q_e
t_{u_1}
t_{v_1}
w_u^e
w_v^e
t_{u_1}
t_{v_1}
r
V_i gadget
V_j gadget
t_{u_1}
t_{u_2}
t_{u_n}
t_{v_1}
t_{v_2}
u_1^-
u_1^+
u_2^-
u_2^+
u_n^-
u_n^+
v_1^-
v_1^+
v_2^-
v_2^+
v_n^-
v_n^+
t_{u_2}
t_{u_n}
t_{u_1}
t_{v_1}
t_{v_2}
s


u_1v_1 gadget
z
t_{v_1}
q_e
t_{u_1}
t_{v_1}
w_u^e
w_v^e
t_{u_1}
t_{v_1}
V_i gadget
r
V_j gadget
t_{u_1}
t_{u_2}
t_{u_n}
t_{v_1}
t_{v_2}
u_1^-
u_1^+
u_2^-
u_2^+
u_n^-
u_n^+
v_1^-
v_1^+
v_2^-
v_2^+
v_n^-
v_n^+
t_{u_2}
t_{u_n}
t_{u_1}
t_{v_1}
t_{v_2}
s

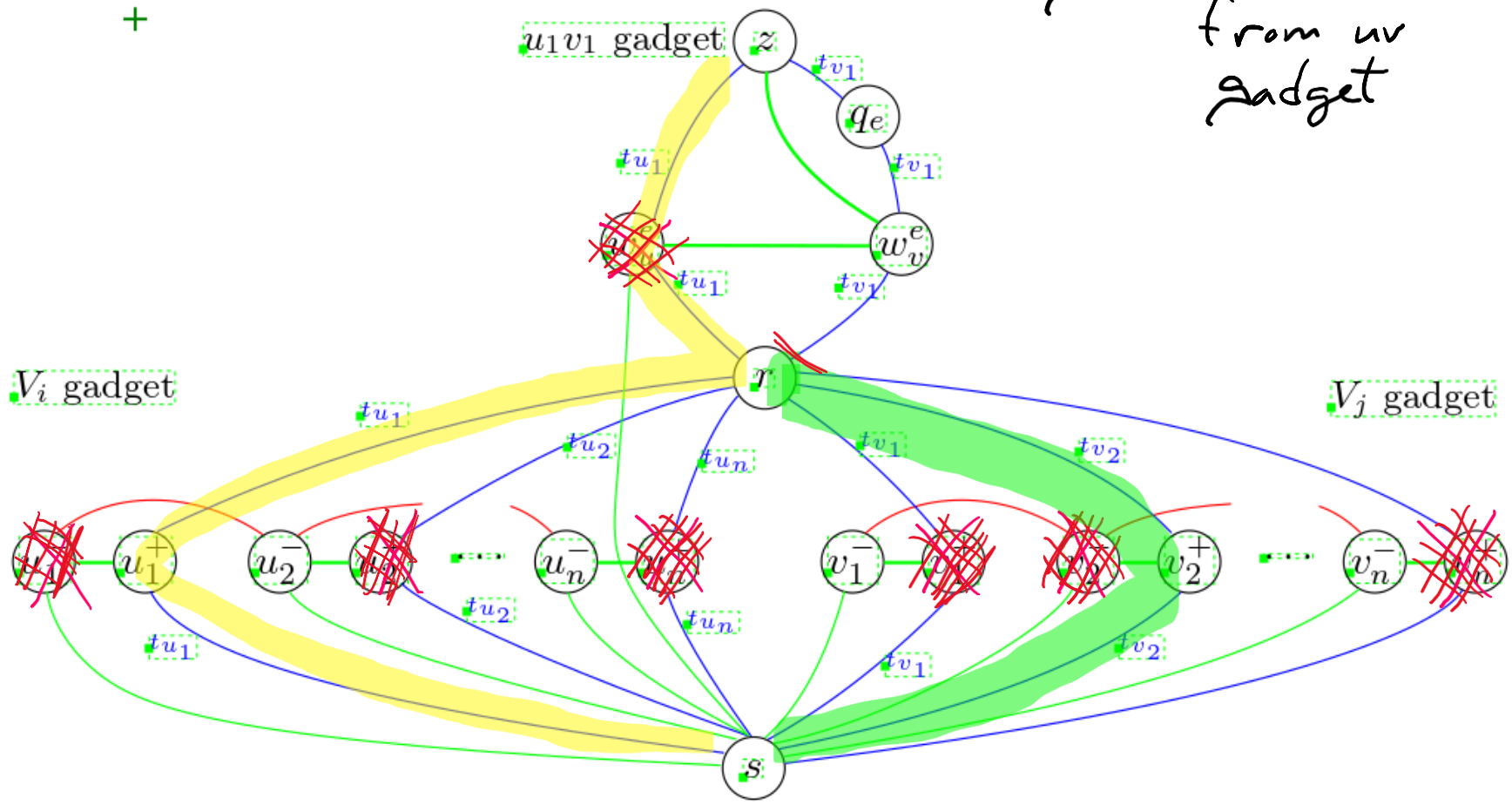
Bad = must delete two from uv gadget



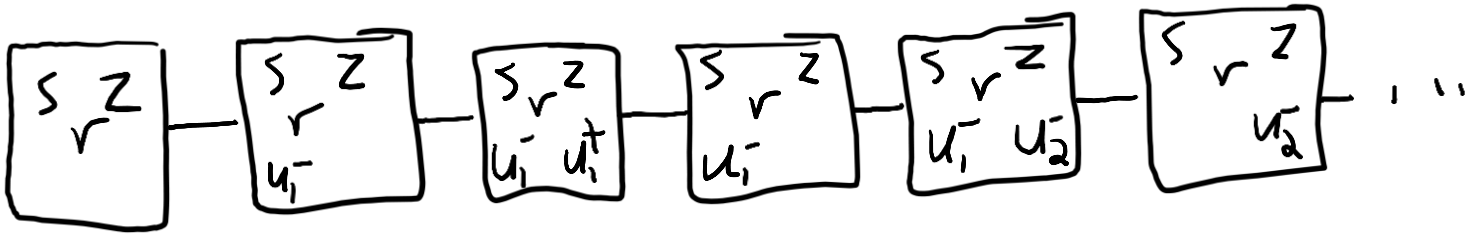
good = delete one from uv gadget

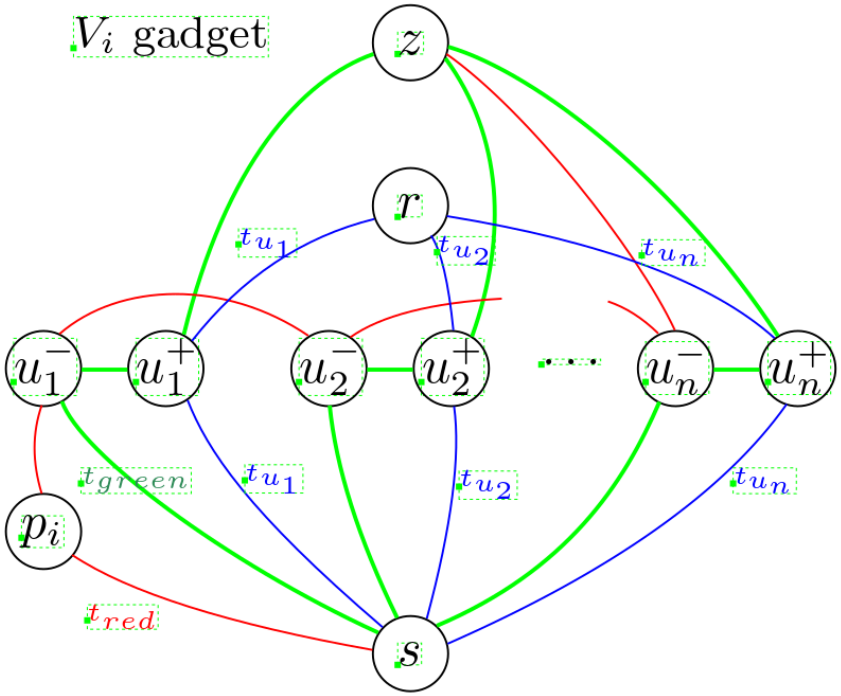


good = delete one from uv gadget



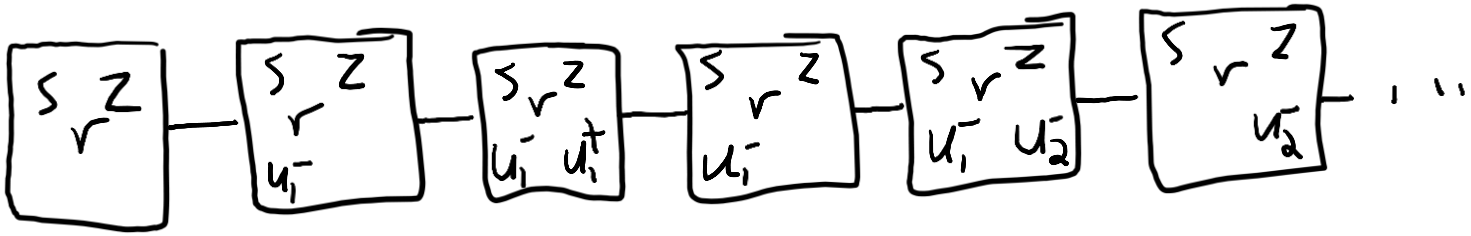
This graph has pathwidth 4 because ...

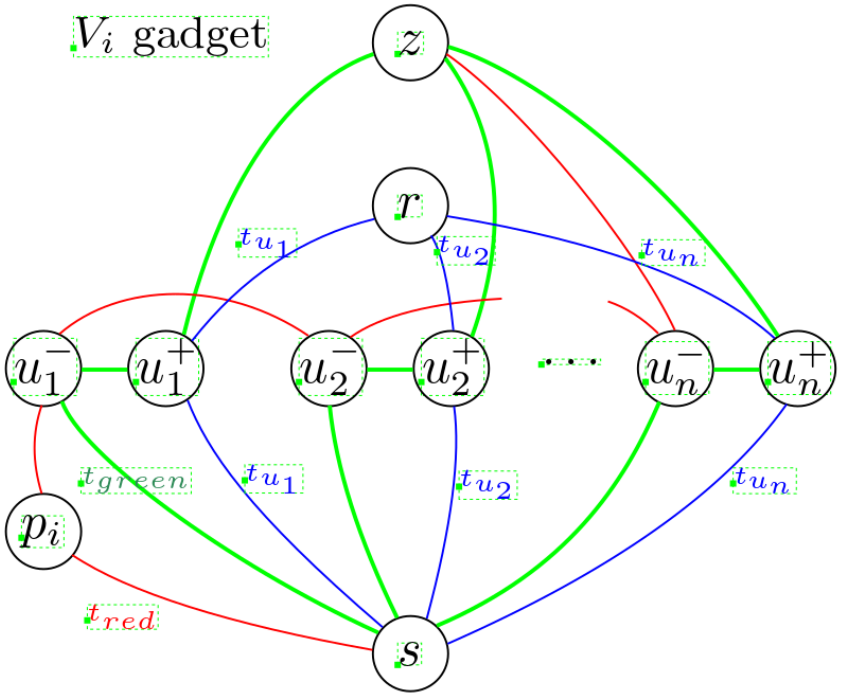




This graph has pathwidth 4 because we can:

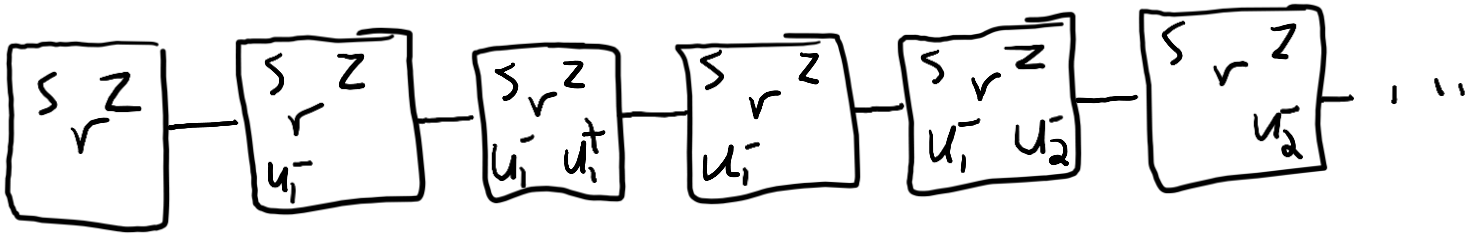
- Create s, z, r in memory and keep them there forever.
- Each V_i gadget can be built with at most two more vertices in memory at any time.
- Same with edge gadgets.

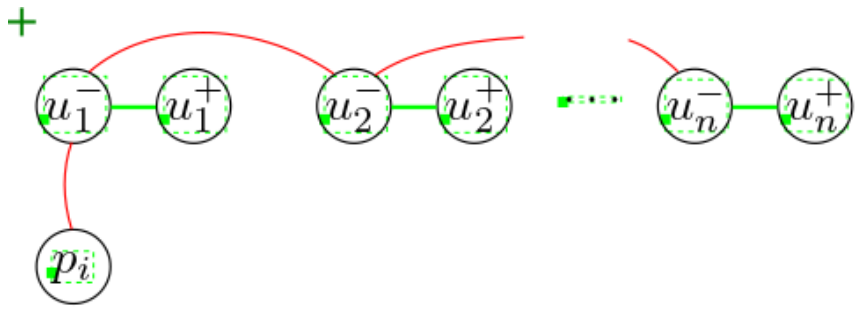




This graph has pathwidth 4 because we can:

- Create s, z, r in memory and keep them there forever.
- Each V_i gadget can be built with at most two more vertices in memory at any time.
- Same with edge gadgets.



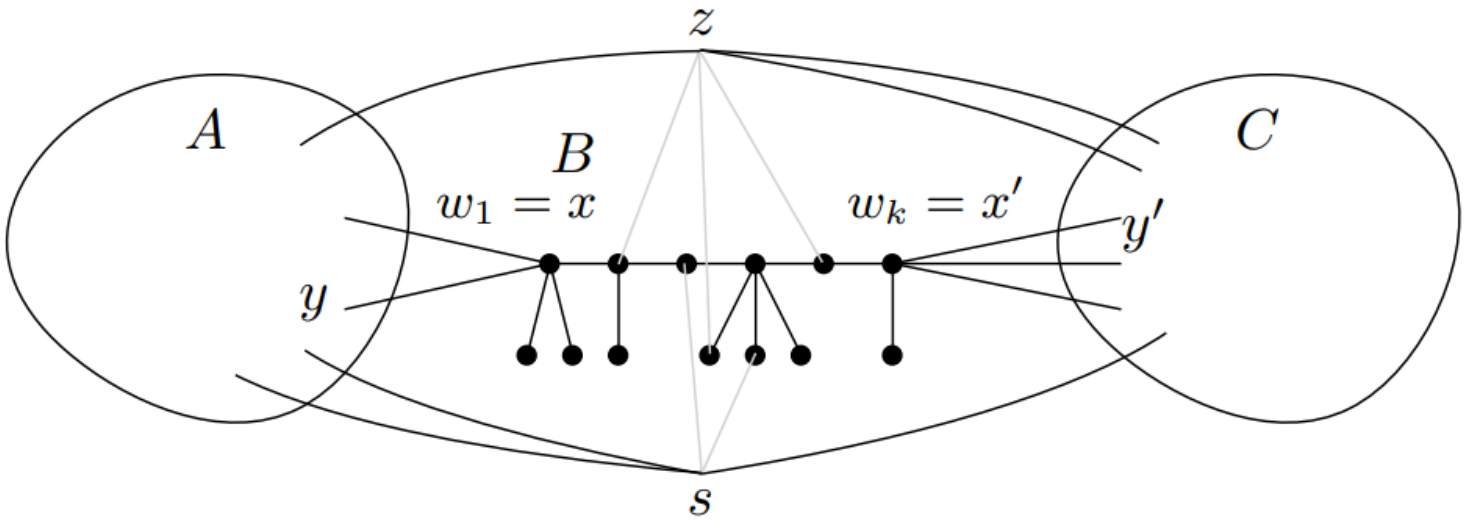


Theorem: (s, z, ℓ)-temporal separator is NP-hard on graphs of pathwidth 4, even when $\ell = 1$.

It is, however, in P on graphs of pathwidth 3 or less, for any ℓ.

Lemma: If G has pathwidth 3, then either it has an (s, z, ℓ)-separator of size at most 3, or $V(G)$ can be partitioned into three sets $\{A, B, C\}$ such that:

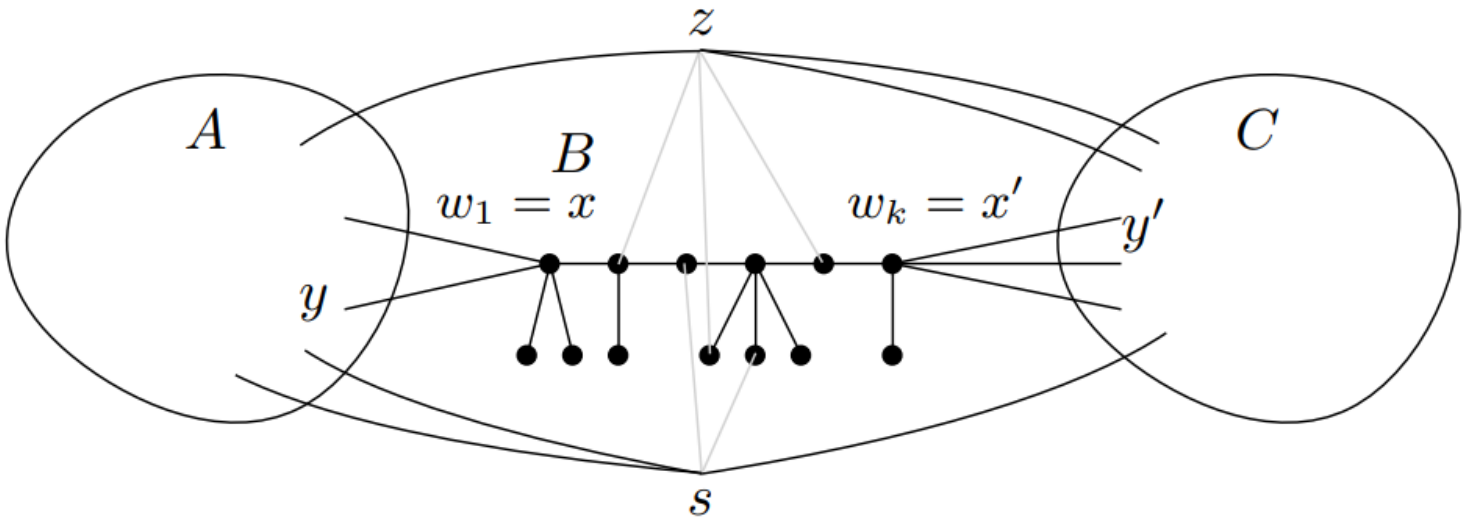
- Any min (s, z, ℓ)-separator has at most one vertex in A and in C.
- B induces a caterpillar, and only the ends have neighbors in A or C.



Lemma: If G has pathwidth 3, then either it has an (s, z, ℓ)-separator of size at most 3, or $V(G)$ can be partitioned into three sets $\{A, B, C\}$ such that:

- Any min (s, z, ℓ)-separator has at most one vertex in A and in C.
- B induces a caterpillar, and only the ends have neighbors in A or C.

Idea: try all $O(n^3)$ sets of size at most 3. If none is a separator, use the structure. Guess the vertex in the separator from A and from C, then handle the B caterpillar in polytime.



Theorem: (s, z, ℓ)-temporal separator admits no $2^{\Omega(log^{1-\epsilon}(n))}$approx. unless NP⊆ZPP.

Theorem: (s, z, ℓ)-temporal separator admits no $2^{\Omega(log^{1-\epsilon}(n))}$approx. unless NP⊆ZPP.

Reduction from Directed Multicut, which has that inapproximability.

Directed Multicut

Input: directed graph $D = (V, A)$, pairs of vertices $(s_1, z_1), \ldots, (s_k, z_k)$.

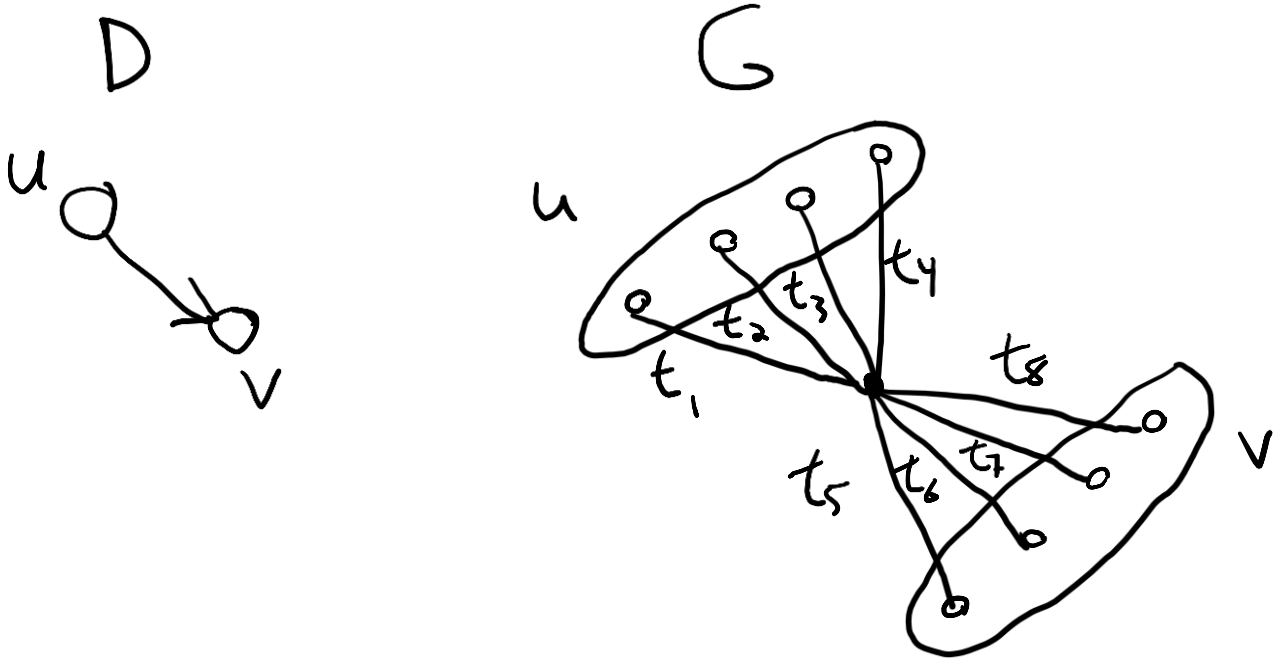
Find: min number of arcs to delete to disconnect each s_i from z_i.

Reduction from Directed Multicut.

Main differences with our problem: we delete vertices (not arcs), and we are undirected.

Idea from given D:

- Each v has m copies $v_1, \ldots, v_m$. $(m = |E(G)|)$
- Each arc (u, v) becomes a vertex $x_{u,v}$. Add edges $u\ x_{u,v}$ and $x_{u,v}\ v$.
- Use time to enforce same direction as the arc.

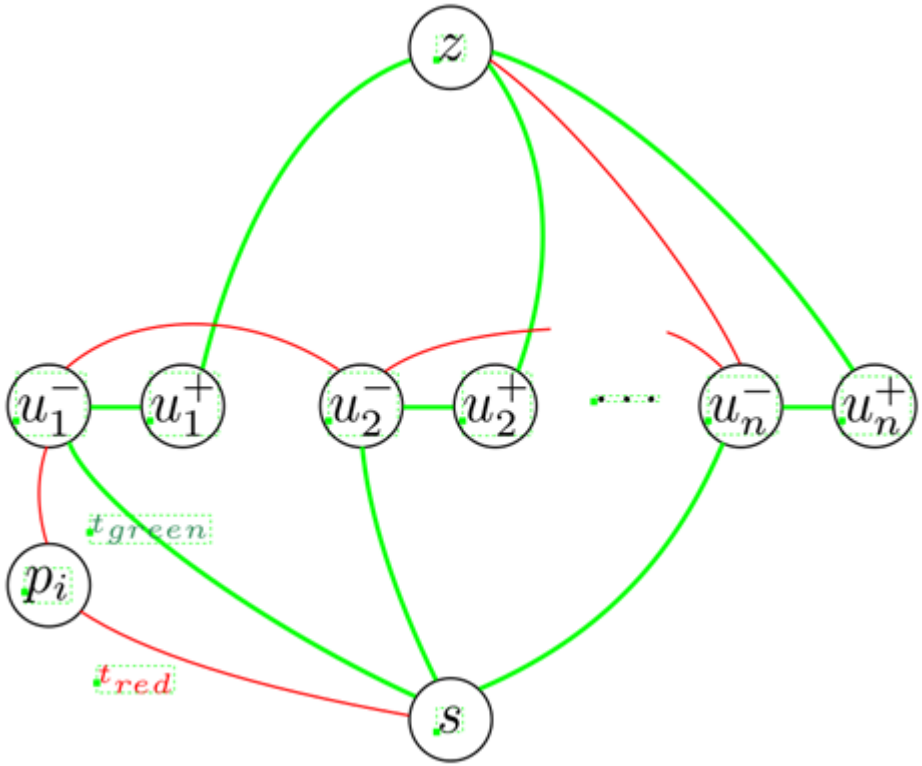


Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard when $\ell \geq 2$. It admits a $O(\log \ell)$-approx.

Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard when $\ell \geq 2$. It admits a $O(\log \ell)$-approx.

Warm up: (s, z, ℓ)-temporal cut is in P when $\ell = 1$.

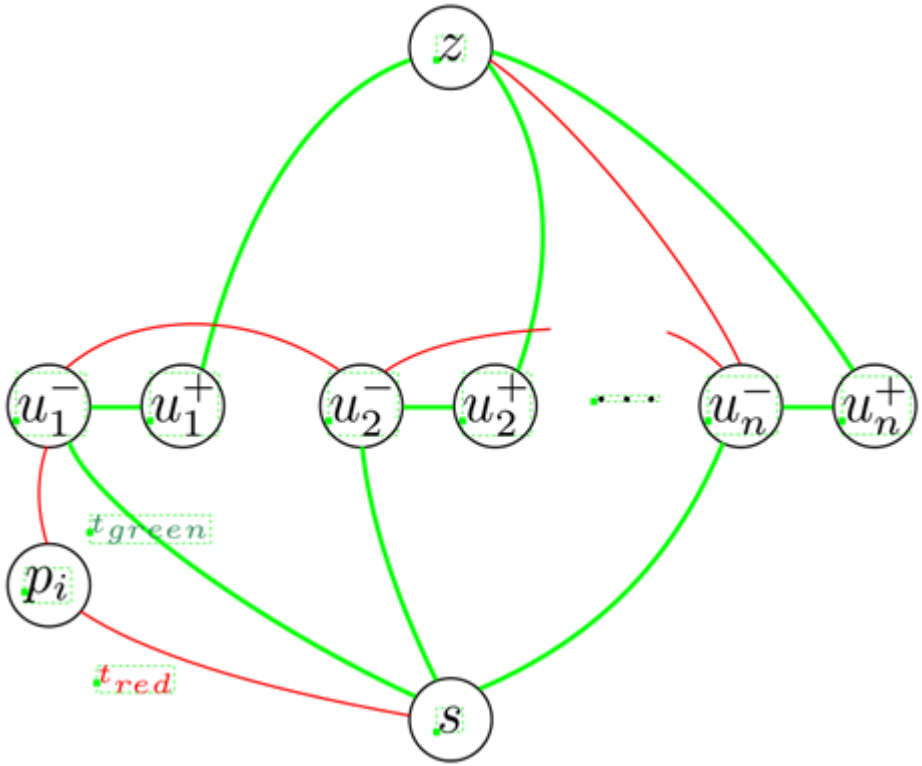
Just compute a min-cut for each time separately.



Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard when $\ell \geq 2$. It admits a $O(\log \ell)$-approx.

Warm up: (s, z, ℓ)-temporal cut is in P when $\ell = 1$.

Doesn't work when $\ell = 2$.



Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard when $\ell \geq 2$. It admits a $O(\log \ell)$-approx.

When $\ell = 2$, reduction from Multiway Cut, which is APX-hard.

Multiway Cut

Input: graph $G = (V, E)$, vertices $s_1, \ldots, s_k$

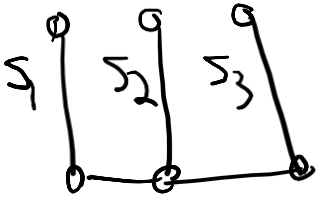
Find: min number of edges to delete so that all s_i's are disconnected from each other.

Multiway Cut

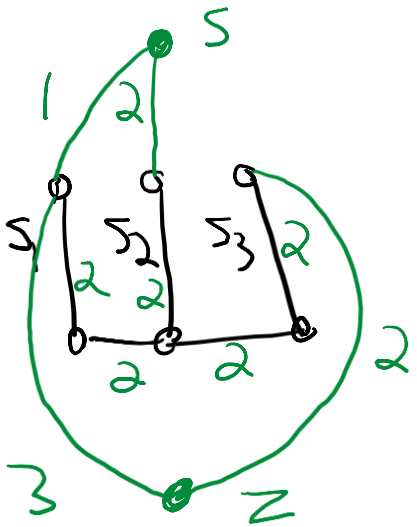
Input: graph $G = (V, E)$, vertices $s_1, \ldots, s_k$

Find: min number of edges to delete so that all s_i's are disconnected from each other.

Multiway cut



$(s, z, 2)$-terminal sep

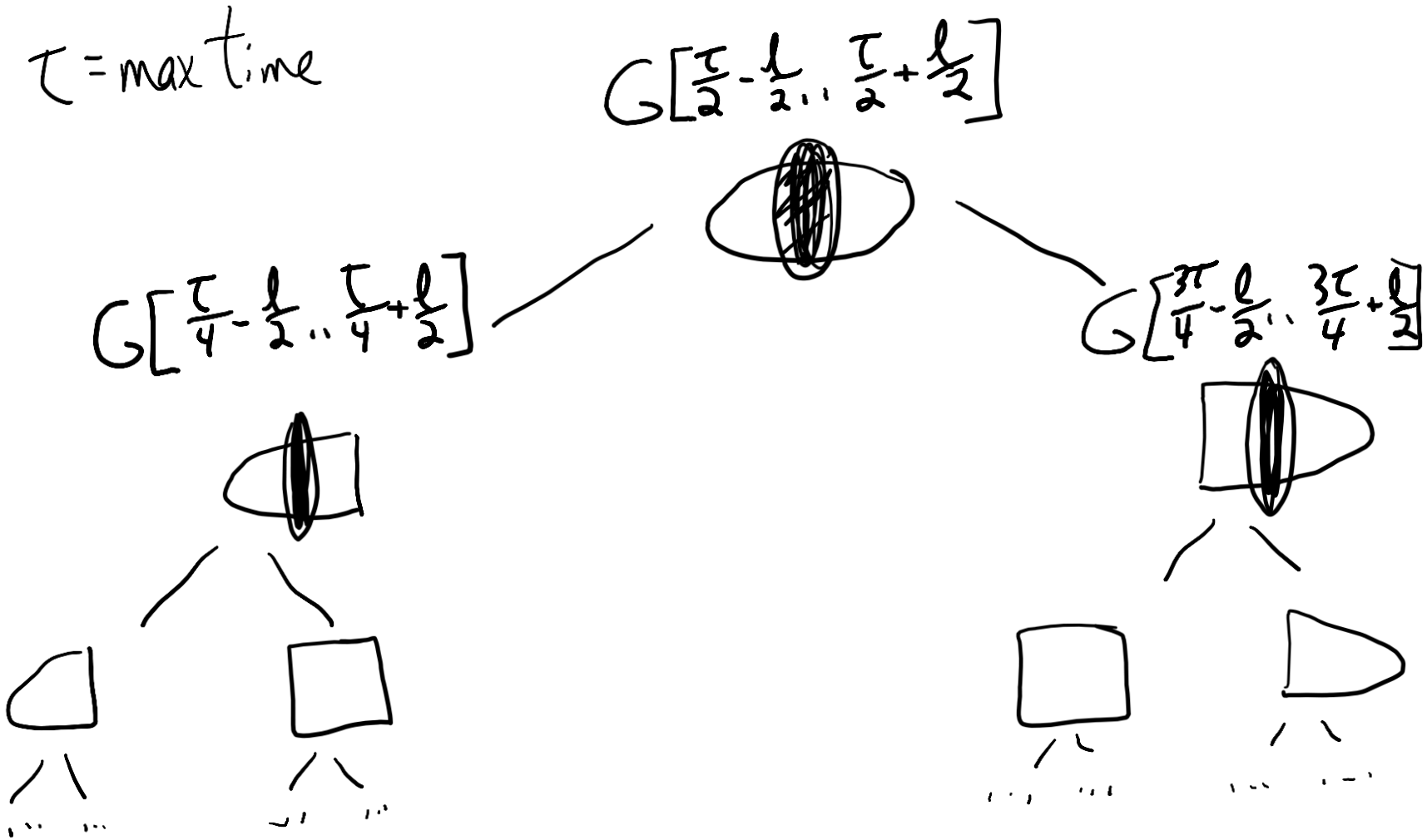


$\Leftrightarrow$

Edge cuts correspond

Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard. It admits a $O(\log \ell)$-approx.

Theorem: The (s, z, ℓ)-temporal **cut** problem is APX-hard. It admits a $O(\log \ell)$-approx.



Open problems

- Is (s, z, ℓ)-temporal separator in P on graphs of treewidth 3?
  - pw=4 result implies hardness for tw≥4
- Is there an $O(\ell)$-approx for (s, z, ℓ)-temporal separator?
- Is there an $O(1)$-approx for (s, z, ℓ)-temporal cut?
- Is the strict variant of (s, z, ℓ)-temporal cut in P?