Kernelizing Temporal Exploration Problems

Emmanuel Arrighi[†] Fedor V. Fomin[*] Petr Golovach[*] Petra Wolf[*]

Algorithmic Aspects of Temporal Graphs VI

[*]University of Bergen, Norway

[†]University of Trier, Germany

Temporal Graph

Graphs that vary over time.

Definition (Temporal graphs)

A temporal graph $\mathcal{G}$ over a set of vertices V is a sequence $\mathcal{G} = (G_1, G_2, \ldots, G_L)$ of graphs such that for all $t \in [L]$, $V(G_t) = V$.

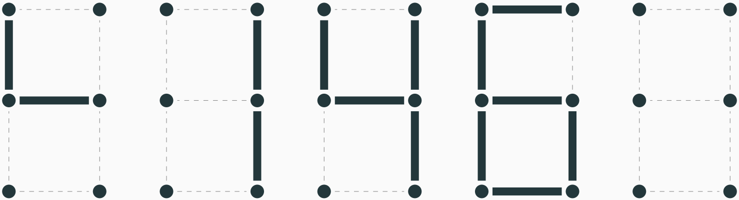
- Lifetime L
- Snapshot graph G_t, $t \in [L]$
- Underlying graph $G = (V, E)$ with $E = \bigcup_{t \in [L]} E(G_t)$
- $\mathfrak{m} = \sum_{t \in [L]} |E(G_t)|$

Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

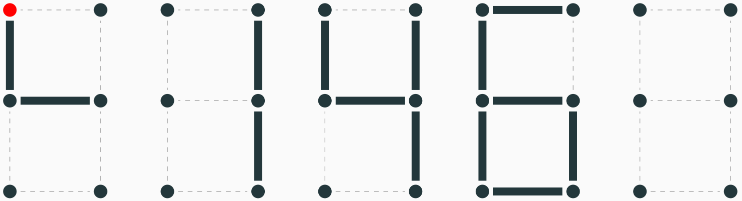


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

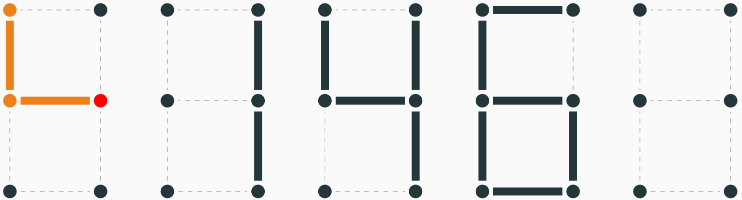


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

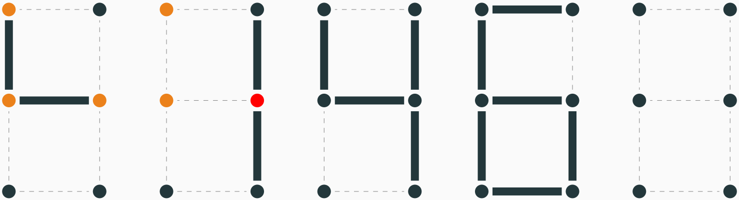


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

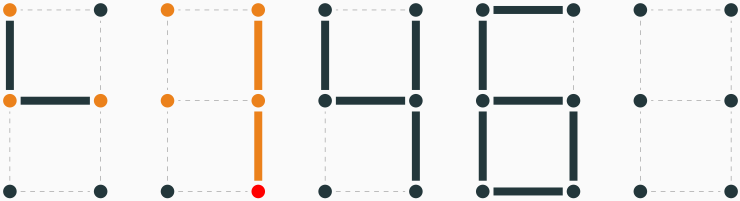


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

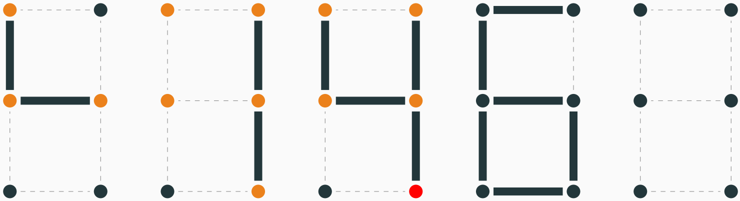


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

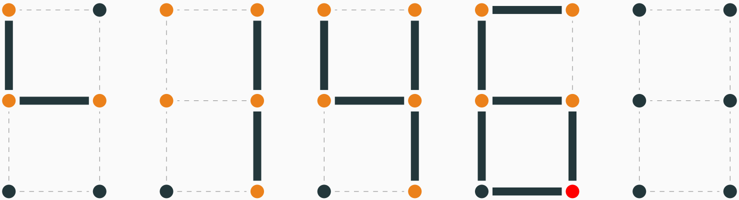


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

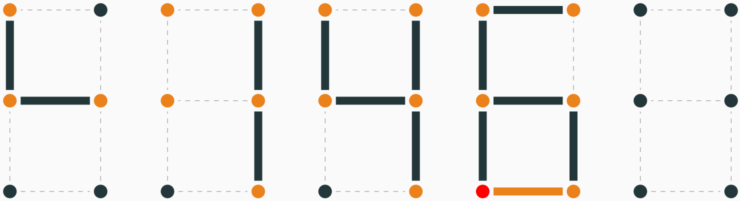


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:

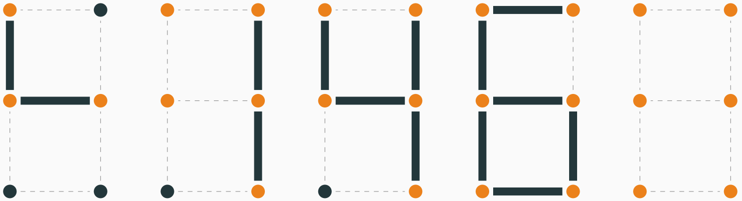


Non-Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Non-strict temporal walk: cross arbitrary many edges per time-step.

Example:



Temporal Exploration

Definition (Non-Strict Temporal Exploration (NS-TEXP))

Input: Temporal graph $\mathcal{G} = (G_1, G_2, \ldots, G_L)$, vertex $v \in V(\mathcal{G})$.
Question: Does there exist a *non-strict temporal walk* in $\mathcal{G}$ that starts in v and visits all vertices in $V(\mathcal{G})$?

Temporal Exploration

Definition (Non-Strict Temporal Exploration (NS-TEXP))

Input: Temporal graph $\mathcal{G} = (G_1, G_2, \ldots, G_L)$, vertex $v \in V(\mathcal{G})$.
Question: Does there exist a *non-strict temporal walk* in $\mathcal{G}$ that starts in v and visits all vertices in $V(\mathcal{G})$?

k-ARB NS-TEXP: visit at least k vertices.

Temporal Exploration

Definition (Non-Strict Temporal Exploration (NS-TEXP))

Input: Temporal graph $\mathcal{G} = (G_1, G_2, \ldots, G_L)$, vertex $v \in V(\mathcal{G})$.
Question: Does there exist a *non-strict temporal walk* in $\mathcal{G}$ that starts in v and visits all vertices in $V(\mathcal{G})$?

k-ARB NS-TEXP: visit at least k vertices.

Definition (Weighted k-arb NS-TEXP)

Input: Temporal graph $\mathcal{G} = (G_1, G_2, \ldots, G_L)$, vertex $v \in V(\mathcal{G})$, weight function $w \colon V \to \mathbb{N}$, integer k.
Question: Does there exist a *non-strict temporal walk* in $\mathcal{G}$ that starts in v and visits vertices $\{v_1, v_2, \ldots, v_\ell\} \subseteq V(\mathcal{G})$ with weight $\sum_{1 \le i \le \ell} w(v_i) \ge k$?

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.
- [Erlebach & Spooner, SIROCCO 20] introduced $\mathrm{NS\text{-}TEXP}$ and showed NP-hardness.

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.
- [Erlebach & Spooner, SIROCCO 20] introduced NS-TEXP and showed NP-hardness.
- [Erlebach & Spooner, SAND 22] NS-TEXP is FPT in L and k-ARB NS-TEXT is FPT in k.

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.
- [Erlebach & Spooner, SIROCCO 20] introduced NS-TEXP and showed NP-hardness.
- [Erlebach & Spooner, SAND 22] NS-TEXP is FPT in L and k-ARB NS-TEXT is FPT in k.

  Question

  - Is NS-TEXP FPT/XP in parameter maximal number of connected components per time-step?

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.
- [Erlebach & Spooner, SIROCCO 20] introduced $\mathrm{NS\text{-}TEXP}$ and showed NP-hardness.
- [Erlebach & Spooner, SAND 22] $\mathrm{NS\text{-}TEXP}$ is FPT in L and k-$\textsc{arb}$ $\mathrm{NS\text{-}TEXT}$ is FPT in k.

  Question

  - Is $\mathrm{NS\text{-}TEXP}$ FPT/XP in parameter maximal number of connected components per time-step?
  - Is k-$\textsc{arb}$ $\mathrm{NS\text{-}TEXT}$ FPT in L?

Problem History

- [Michail & Spirakis, TCS 16] introduced Strict Temporal Exploration (cross only one edge per time-step) and showed NP-hardness.
- [Erlebach & Spooner, SIROCCO 20] introduced NS-TEXP and showed NP-hardness.
- [Erlebach & Spooner, SAND 22] NS-TEXP is FPT in L and k-ARB NS-TEXT is FPT in k.

  Question

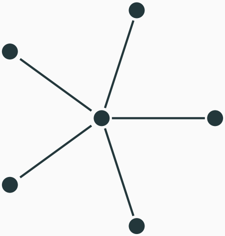
  - Is NS-TEXP FPT/XP in parameter maximal number of connected components per time-step? - No!
  - Is k-ARB NS-TEXT FPT in L? - No!

Parameterized Complexity of k-arb NS-TEXP

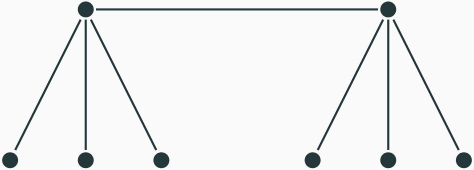
Param	FPT	Kernel
n	FPT in k (Erlebach & Spooner, 2022)	no poly kernel
L	W[1]-hard	no poly kernel
k	$\mathcal{O}^*((2e)^k k^{\log k})$ (Erlebach & Spooner, 2022)	no poly kernel
$L+k$	FPT in k (Erlebach & Spooner, 2022)	no poly kernel
γ	in P for ≤ 2 (Erlebach & Spooner, 2022), NP-hard for ≥ 5	-
$L+\gamma$	$\mathcal{O}(\gamma^L n^{\mathcal{O}(1)})$	no poly kernel for $\gamma \geq 6$
$k+\gamma$	FPT in k (Erlebach & Spooner, 2022)	no poly kernel

Two easy examples as a warm up

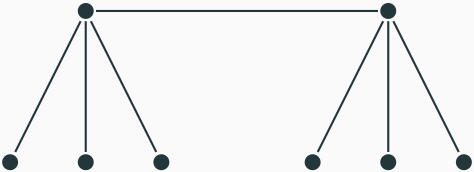
Two easy examples as a warm up



Easy?



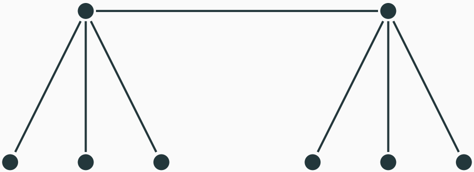
Easy?



Theorem

NS-TEXP is NP-complete for temporal graphs where the underlying graph consists of two stars connected with a bridge.

Easy?



Theorem

NS-TEXP is NP-complete for temporal graphs where the underlying graph consists of two stars connected with a bridge.

Theorem

NS-TEXP is NP-complete for temporal graphs where each edge appears only once.

Temporal Sparsity

Lemma

If the underlying graph is a tree, and every edge appears only once, then we can find a maximum weight non-strict temporal walk from a vertex x to a vertex y in polynomial time.

Temporal Sparsity

Lemma

If the underlying graph is a tree, and every edge appears only once, then we can find a maximum weight non-strict temporal walk from a vertex x to a vertex y in polynomial time.

Definition

$$p(\mathcal{G}) = \mathfrak{m} - n + 1$$

Temporal Sparsity

Lemma

If the underlying graph is a tree, and every edge appears only once, then we can find a maximum weight non-strict temporal walk from a vertex x to a vertex y in polynomial time.

Definition

$$p(\mathcal{G}) = \mathfrak{m} - n + 1$$

Notations:

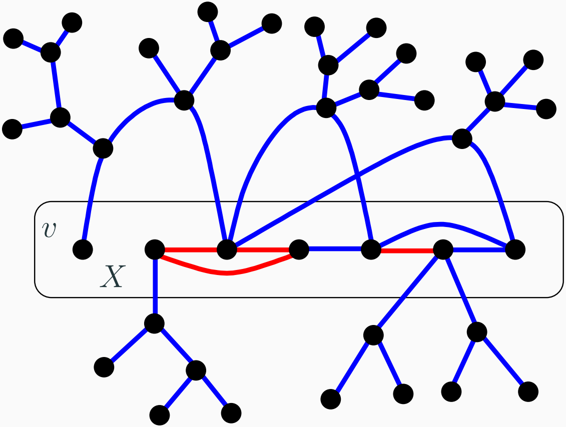
- Blue edges: edges that appear only once
- Red edges: edges that appear at least twice.

Kernel in $p(\mathcal{G})$: Bounded Feedback Edge Set

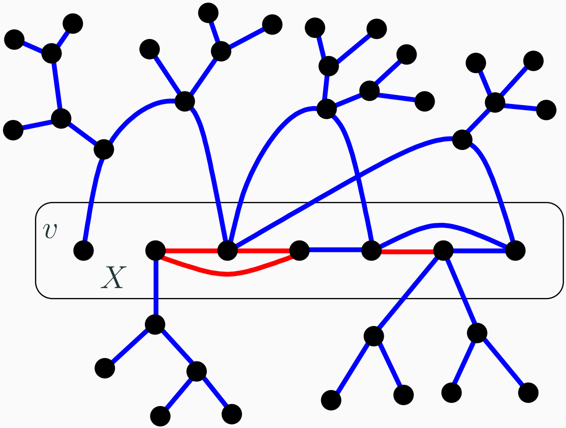
Lemma

Let $\mathcal{G}$ be a temporal graph. Then, the underlying graph G of $\mathcal{G}$ has a feedback edge set S of size at most p such that all the red edges are in S.

Kernel in $p(\mathcal{G})$: Structure of the Underlying Graph

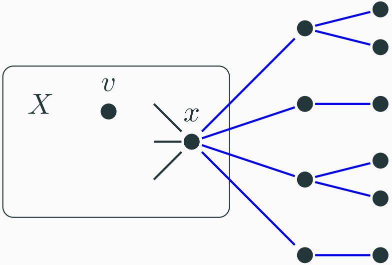


Kernel in $p(\mathcal{G})$: Structure of the Underlying Graph

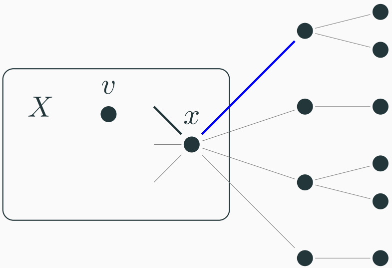


$$|X| \leq 4p$$

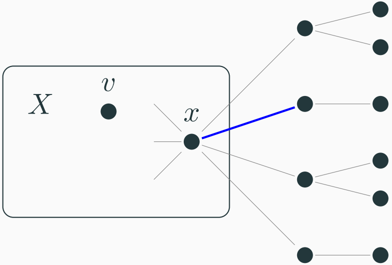
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



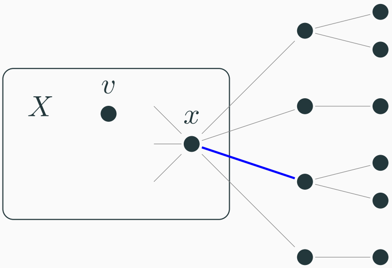
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



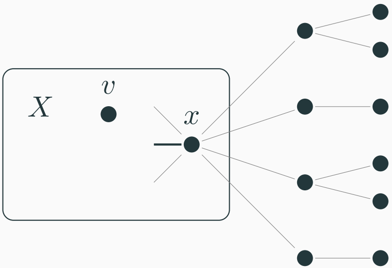
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



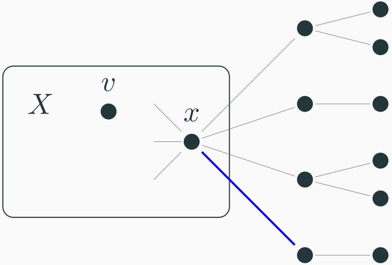
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



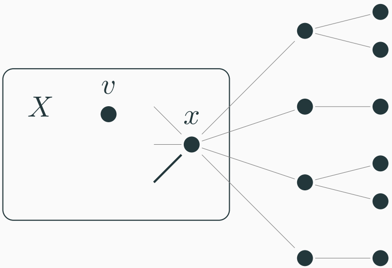
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



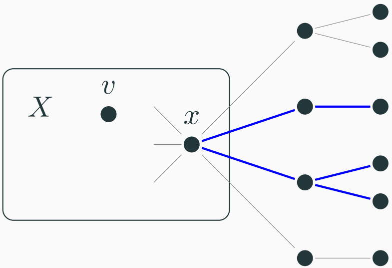
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



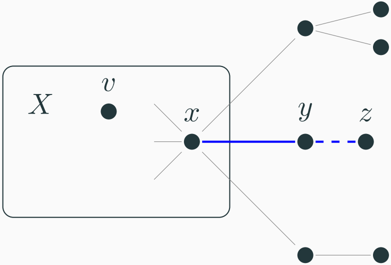
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



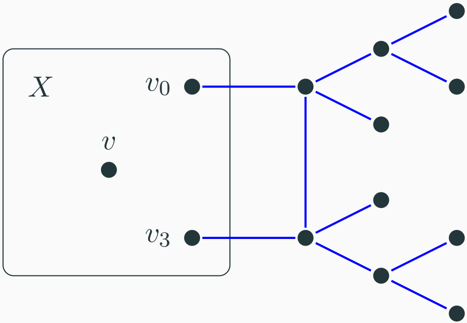
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



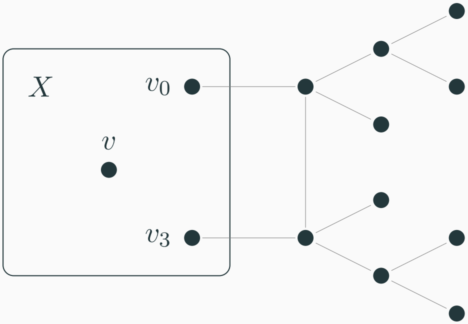
Kernel in $p(\mathcal{G})$: Reduction Rule (Tree compression)



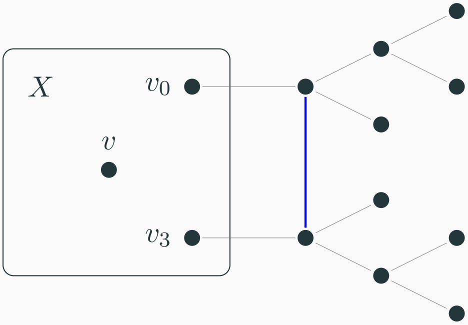
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



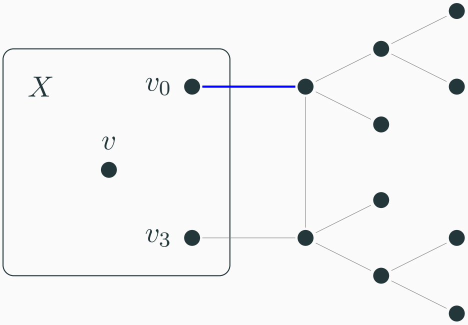
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



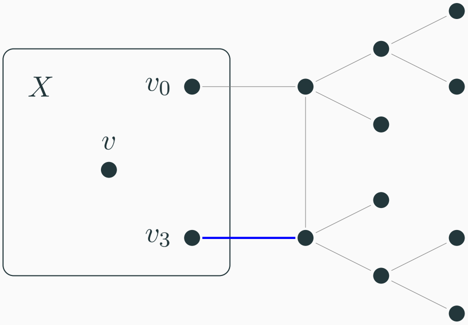
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



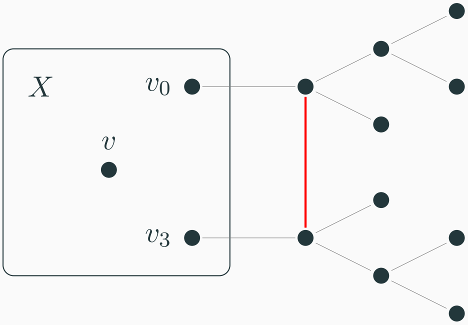
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



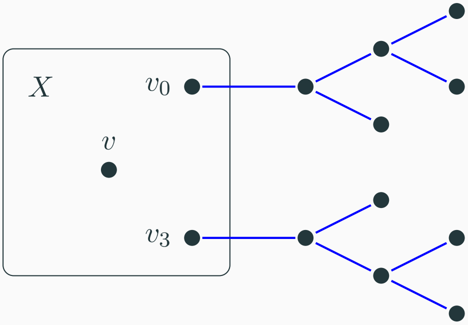
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



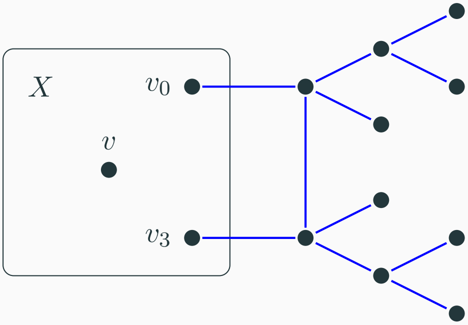
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



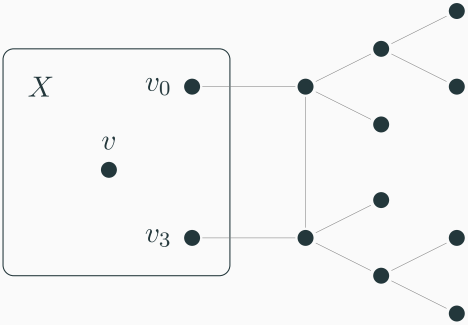
Kernel in $p(\mathcal{G})$: Reduction Rule (Non usable edge)



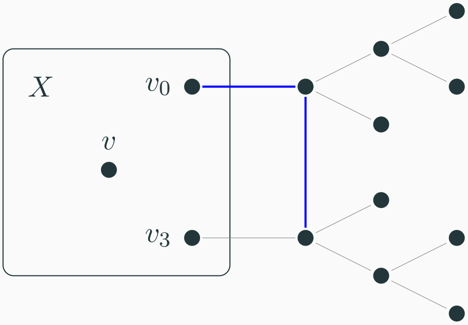
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



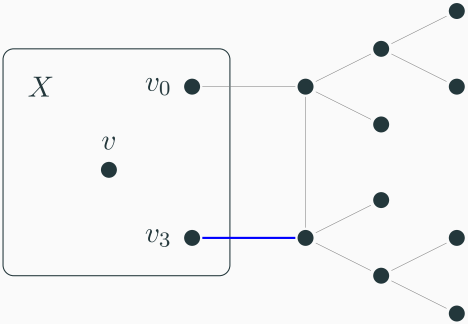
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



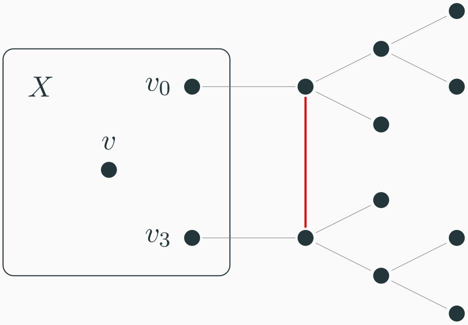
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



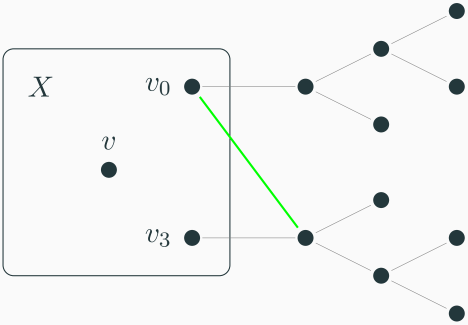
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



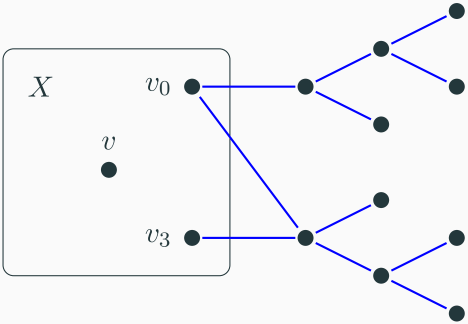
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



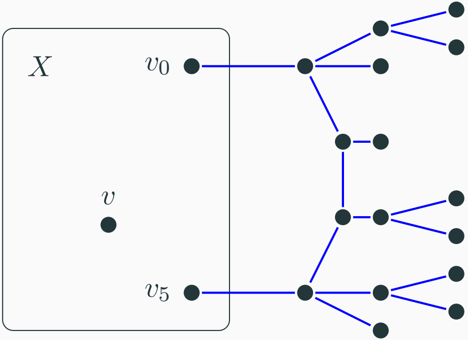
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



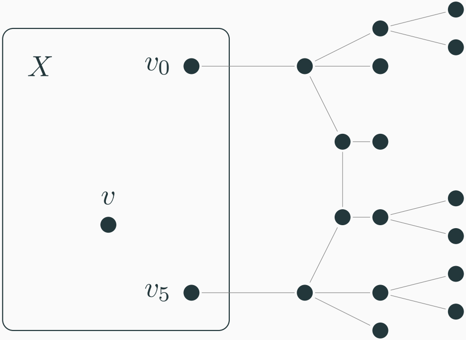
Kernel in $p(\mathcal{G})$: Reduction Rule (Short cut)



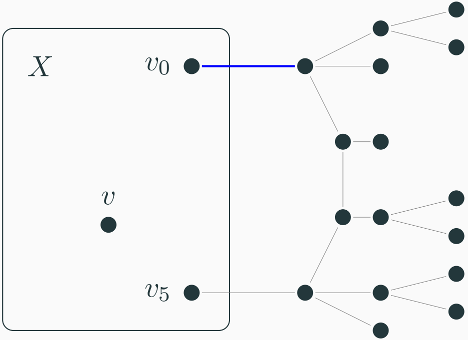
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



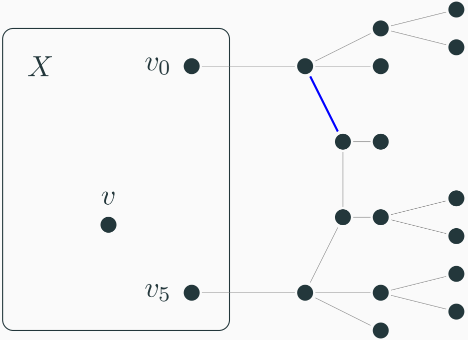
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



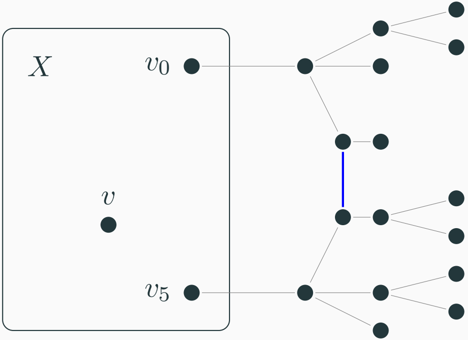
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



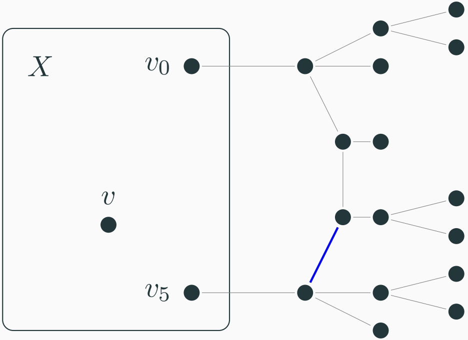
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



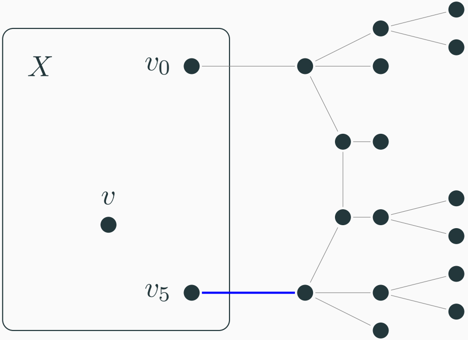
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



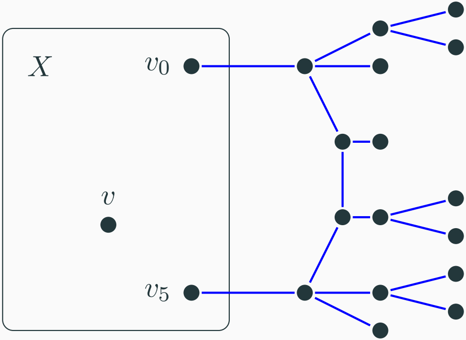
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



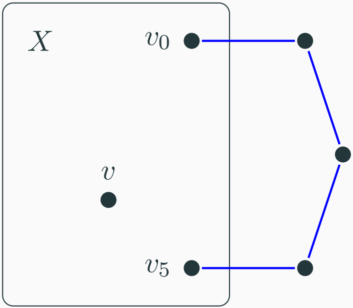
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



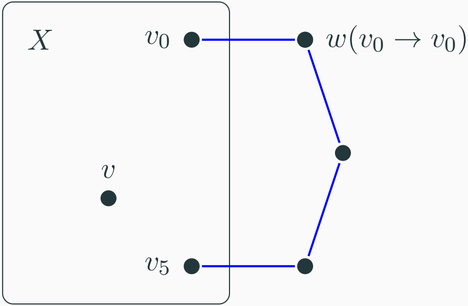
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



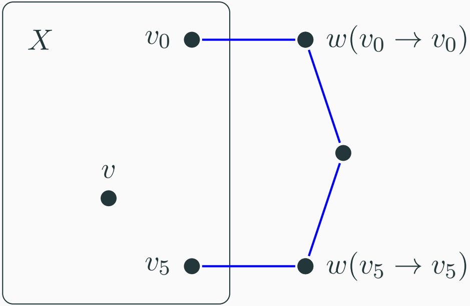
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



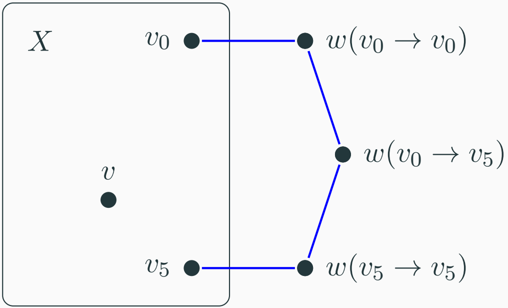
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



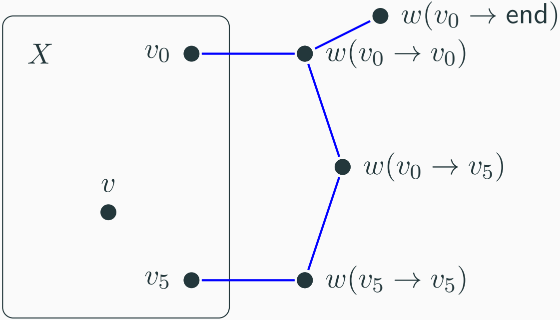
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



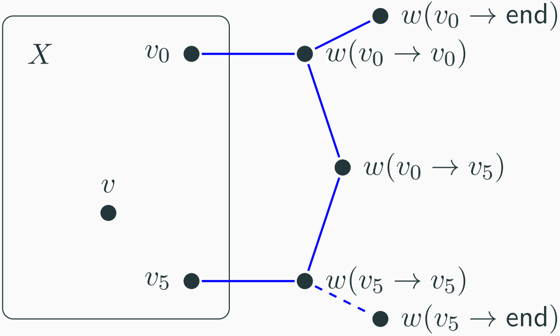
Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



Kernel in $p(\mathcal{G})$: Reduction Rule (Long path)



Kernel in $p(\mathcal{G})$: Frank and Tardos magic algorithm

Proposition (Frank & Tardos, 1987)

There is an algorithm that, given a vector $\mathrm{w} \in \mathbb{Q}^r$ and an integer N, in polynomial time finds a vector $\overline{\mathrm{w}} \in \mathbb{Z}^r$ with $\|\overline{\mathrm{w}}\|_\infty \leq 2^{4r^3} N^{r(r+2)}$ such that $\mathrm{sign}(\mathrm{w} \cdot b) = \mathrm{sign}(\overline{\mathrm{w}} \cdot b)$ for all vectors $b \in \mathbb{Z}^r$ with $\|b\|_1 \leq N - 1$.

Kernel in $p(\mathcal{G})$: Frank and Tardos magic algorithm

Proposition (Frank & Tardos, 1987)

There is an algorithm that, given a vector $\mathrm{w} \in \mathbb{Q}^r$ and an integer N, in polynomial time finds a vector $\overline{\mathrm{w}} \in \mathbb{Z}^r$ with $\|\overline{\mathrm{w}}\|_\infty \leq 2^{4r^3} N^{r(r+2)}$ such that $\mathrm{sign}(\mathrm{w} \cdot b) = \mathrm{sign}(\overline{\mathrm{w}} \cdot b)$ for all vectors $b \in \mathbb{Z}^r$ with $\|b\|_1 \leq N - 1$.

Reduction Rule (Weights reduction)

Apply this algorithm for $\mathrm{w} = (k, w(v_0), \ldots, w(v_n))$ and $N = r + 1$ and find the vector $\overline{\mathrm{w}} = (\overline{w}_0, \ldots, \overline{w}_n)$. Set $k := \overline{w}_0$ and set $w(v_i) := \overline{w}_i$ for $i \in [n]$.

Reduction Rule (Livetime)

For all $t \in [L]$, if G_t has no edge, then remove G_t from $\mathcal{G}$.

Kernel in $p(\mathcal{G})$

Theorem

Weighted k-arb NS-TEXP *admits a kernel of size* $\mathcal{O}(p^4)$ *for connected underlying graphs such that for the output instance* $(\mathcal{G} = (G_1, \ldots, G_L), w, v, k)$, $\mathcal{G}$ *has* $\mathcal{O}(p)$ *vertices and edges, and* $L \in \mathcal{O}(p)$.

Conclusion

- Adaption for Strict Temporal Exploration possible for many of our result including the kernel.

Conclusion

- Adaption for Strict Temporal Exploration possible for many of our result including the kernel.
- Our parameter applicable to other problems?
- What are good parameters for temporal graphs?

Thank you!

Kernelizing Temporal Exploration Problems

References

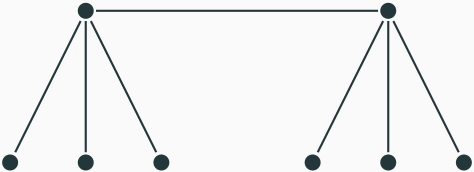
Erlebach, Thomas, & Spooner, Jakob T. 2022.
Parameterized temporal exploration problems.
CoRR, **abs/2212.01594**.

Frank, András, & Tardos, Éva. 1987.
An application of simultaneous Diophantine approximation in combinatorial optimization.
Comb., **7**(1), 49–65.

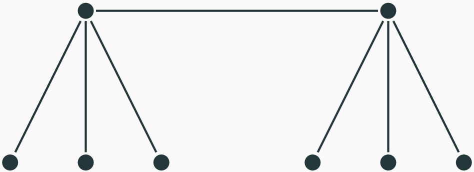
Our contributions: k-arb NS-TEXP

Param	FPT	Kernel
p	$2^{\mathcal{O}(p)}(nL)^{\mathcal{O}(1)}$	$\mathcal{O}(p^4)^\star$
n	FPT in k (Erlebach & Spooner, 2022)	no poly kernel
L	W[1]-hard	no poly kernel
k	$\mathcal{O}^*((2e)^k k^{\log k})$ (Erlebach & Spooner, 2022)	no poly kernel
$L+k$	FPT in k (Erlebach & Spooner, 2022)	no poly kernel
γ	in P for ≤ 2 (Erlebach & Spooner, 2022), NP-hard for ≥ 5	-
$L+\gamma$	$\mathcal{O}(\gamma^L n^{\mathcal{O}(1)})$	no poly kernel for $\gamma \geq 6$
$k+\gamma$	FPT in k (Erlebach & Spooner, 2022)	no poly kernel

Easy?



Easy?



Theorem

NS-TEXP is NP-complete for temporal graphs where the underlying graph consists of two stars connected with a bridge.

Proof

Reduction from Monotone SAT (each clause either contains only positive or only negative literals)

Proof

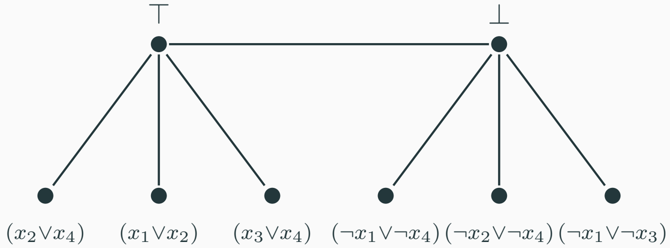
Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

$$\begin{aligned}\Phi = &(x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \\ &\wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)\end{aligned}$$

Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

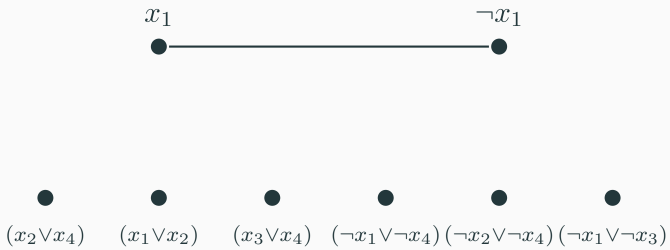
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \\ \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

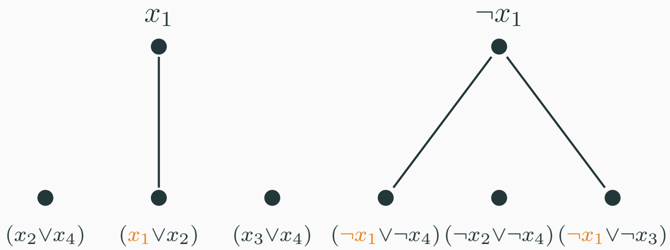
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \\ \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from Monotone SAT (each clause either contains only positive or only negative literals)

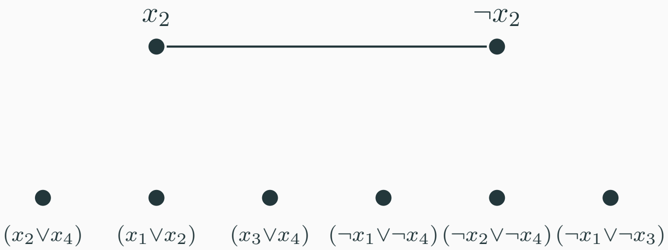
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

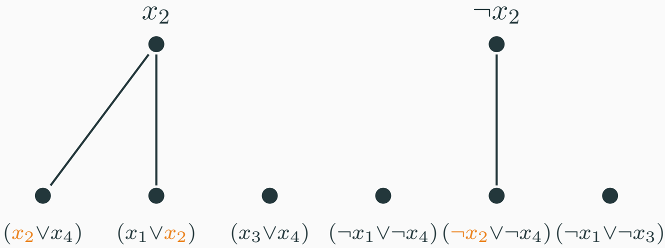
$$\begin{aligned}\Phi = &(x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4)\\ &\wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)\end{aligned}$$



Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

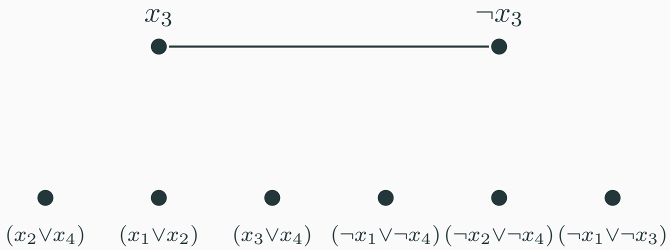
$$\begin{aligned}\Phi = &(x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \\ &\wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)\end{aligned}$$



Proof

Reduction from Monotone SAT (each clause either contains only positive or only negative literals)

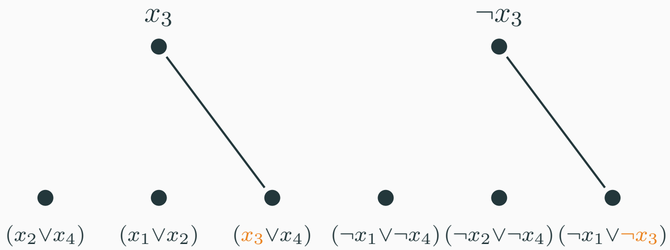
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

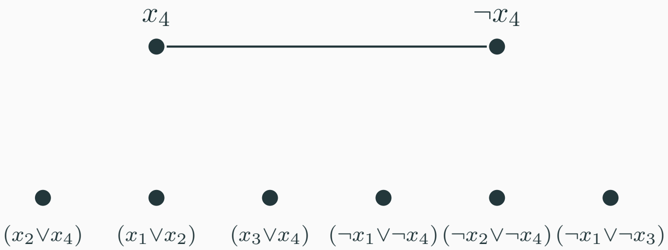
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from Monotone SAT (each clause either contains only positive or only negative literals)

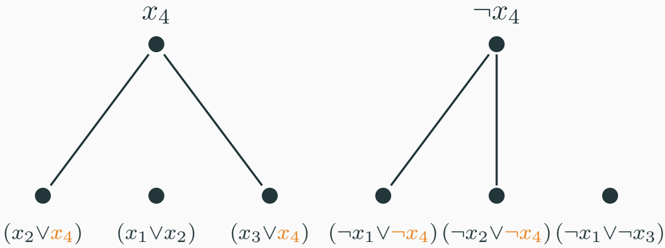
$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$



Proof

Reduction from MONOTONE SAT (each clause either contains only positive or only negative literals)

$$\Phi = (x_2 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$$

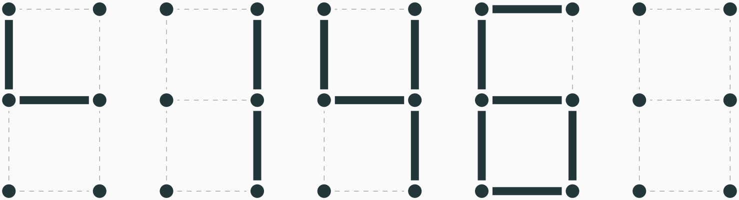


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

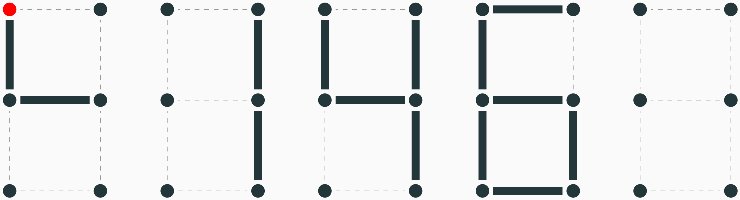


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

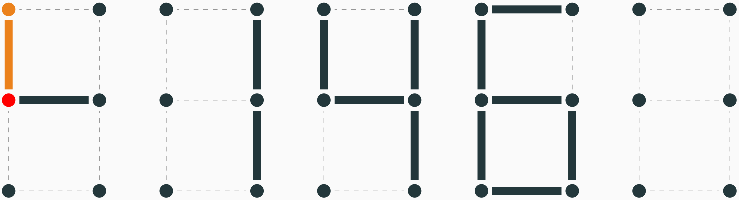


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

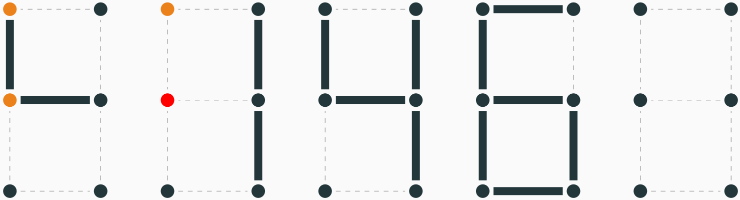


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

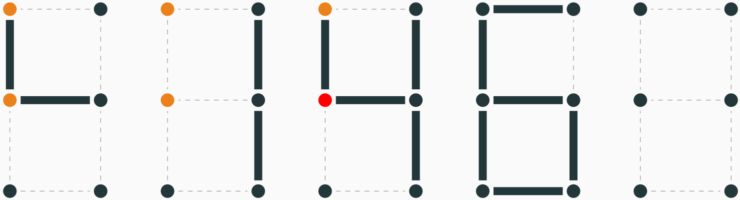


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

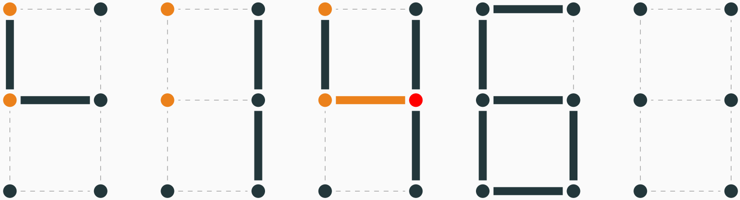


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

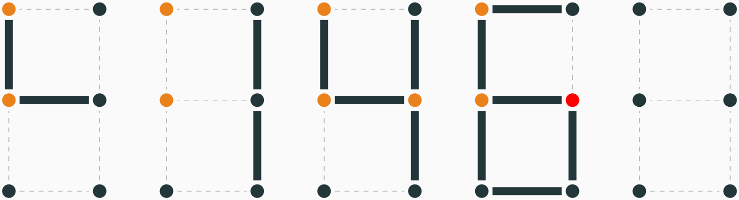


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:

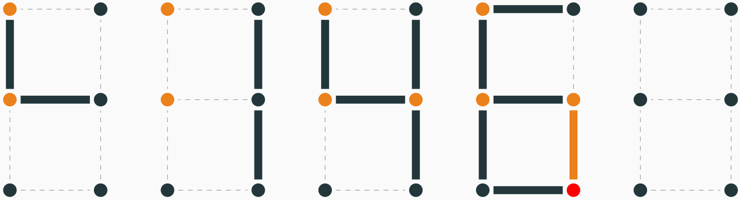


(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example:



(Non-)Strict Temporal Walk

Find temporal walks that starting in vertex v in time-step 0, visit every vertex of input temporal graph.

Strict temporal walk: cross at most one edge per time-step.

Example: