Path Covers of Temporal Graphs: When is Dilworth dynamic?

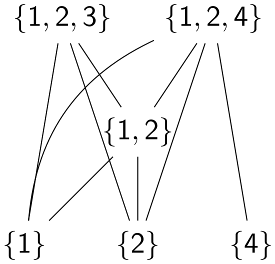
Dibyayan Chakraborty[1], Antoine Dailly[2], Florent Foucaud[2], Ralf Klasing[3]

LITIS Seminar - March 12th 2024

[1] School of Computing, University of Leeds
[2] LIMOS, Clermont-Ferrand
[3] LaBRI, Bordeaux

ANR GRALMECO and TEMPOGRAL

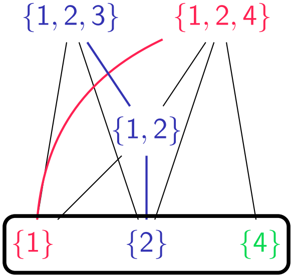
Introduction: Dilworth's theorem



Introduction: Dilworth's theorem

Theorem [Dilworth, 1950]

The minimum size of a chain partition of a finite poset is equal to the maximum size of an antichain of this poset.

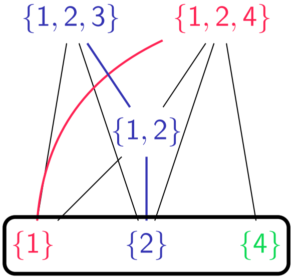


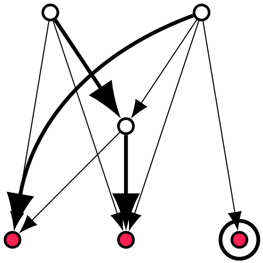
Introduction: Dilworth's theorem

Theorem [Dilworth, 1950]

The minimum size of a path partition of a transitive DAG is equal to the maximum size of an antichain of this DAG.

Restated for graphs...





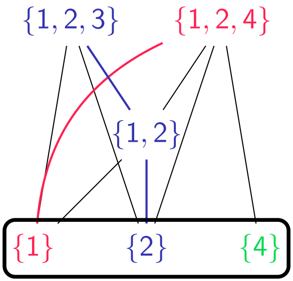
Introduction: Dilworth's theorem

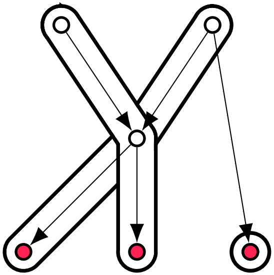
Theorem [Dilworth, 1950]

The minimum size of a path cover of a DAG is equal to the maximum size of an antichain of this DAG.

Restated for graphs...

... and covers.





Introduction: Dilworth's theorem

Theorem [Dilworth, 1950]

The minimum size of a path cover of a DAG is equal to the maximum size of an antichain of this DAG.

Restated for graphs...
... and covers.

Algorithms:

- Algorithmic proof (polynomial time) [Fulkerson, 1956]

Introduction: Dilworth's theorem

Theorem [Dilworth, 1950]

The minimum size of a path cover of a DAG is equal to the maximum size of an antichain of this DAG.

Restated for graphs...
... and covers.

Algorithms:

- Algorithmic proof (polynomial time) [Fulkerson, 1956]
- Many improvements since then, now quasi-linear [Caceres, ICALP 2023]

Introduction: Dilworth's theorem

Theorem [Dilworth, 1950]

The minimum size of a path cover of a DAG is equal to the maximum size of an antichain of this DAG.

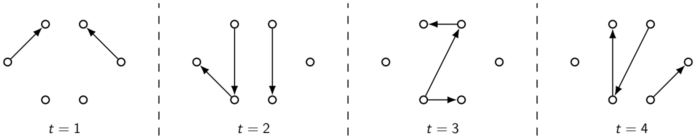
Restated for graphs...
... and covers.

Algorithms:

- Algorithmic proof (polynomial time) [Fulkerson, 1956]
- Many improvements since then, now quasi-linear [Caceres, ICALP 2023]
- NP-hard on general graphs

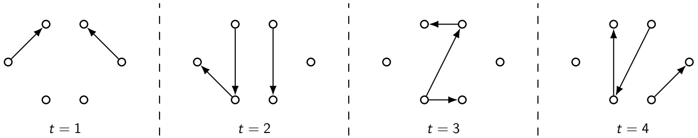
Introduction: temporal (di)graphs

$\mathcal{D} = (V, A_1, A_2, \ldots, A_k)$ [Ferreira & Viennot, 2002]

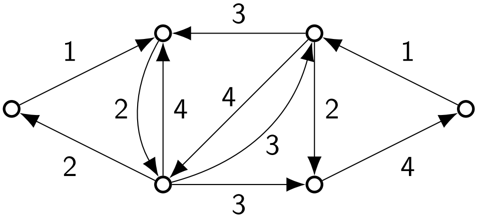


Introduction: temporal (di)graphs

$\mathcal{D} = (V, A_1, A_2, \ldots, A_k)$ [Ferreira & Viennot, 2002]

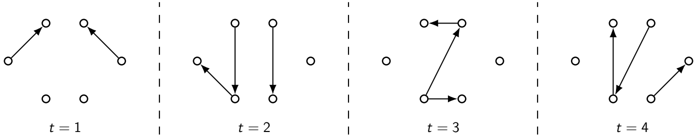


$\mathcal{D} = (V, A, \lambda)$ [Kempe, Kleinberg & Kumar, 2000]

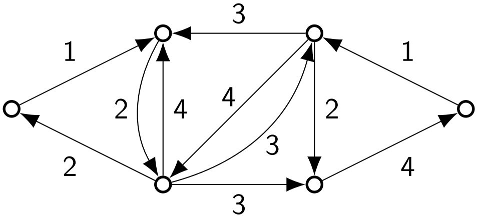


Introduction: temporal (di)graphs

$\mathcal{D} = (V, A_1, A_2, \ldots, A_k)$ [Ferreira & Viennot, 2002]



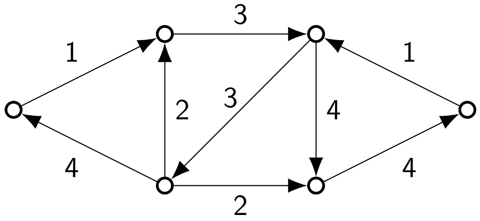
$\mathcal{D} = (V, A, \lambda)$ [Kempe, Kleinberg & Kumar, 2000]



Many results and applications in distributed algorithms, dynamic networks (transportation, social, biological...). More recently, gain of interest from the graph algorithms community.

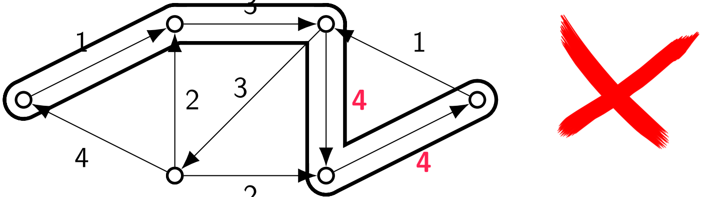
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).



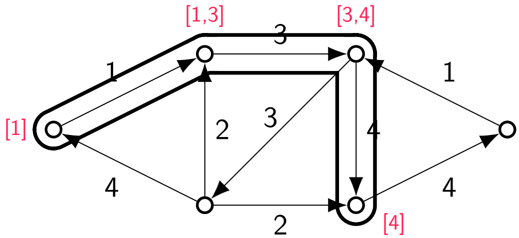
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.



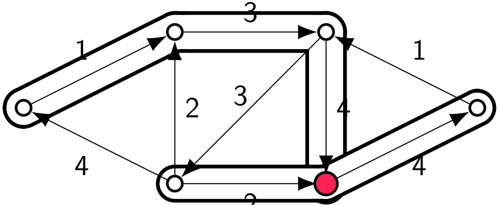
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.
- A temporal path occupies a vertex during interval $[t_1, t_2]$ if it reaches it at time t_1 and leaves it at time t_2.



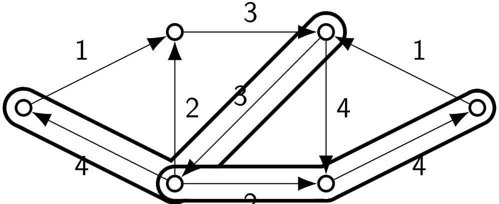
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.
- A temporal path occupies a vertex during interval $[t_1, t_2]$ if it reaches it at time t_1 and leaves it at time t_2.
- Two temporal paths intersect if they occupy the same vertex during non-disjoint intervals.



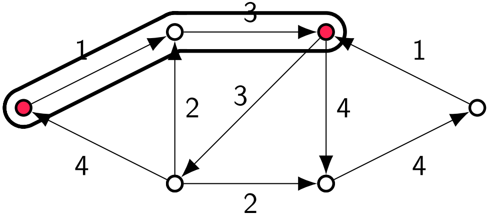
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.
- A temporal path occupies a vertex during interval $[t_1, t_2]$ if it reaches it at time t_1 and leaves it at time t_2.
- Two temporal paths intersect if they occupy the same vertex during non-disjoint intervals. They are temporally disjoint if they do not intersect.



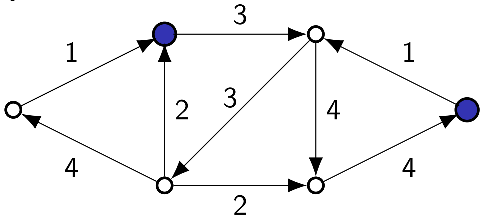
A few definitions for this talk

- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.
- A temporal path occupies a vertex during interval $[t_1, t_2]$ if it reaches it at time t_1 and leaves it at time t_2.
- Two temporal paths intersect if they occupy the same vertex during non-disjoint intervals. They are temporally disjoint if they do not intersect.
- Two vertices are temporally connected if there is a temporal path between them.



A few definitions for this talk

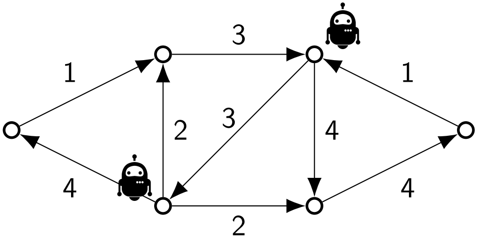
- A temporal DAG (resp. tree...) is a temporal (di)graph whose underlying (di)graph is a DAG (resp. tree...).
- (Directed) temporal path : strictly increasing time labels.
- A temporal path occupies a vertex during interval $[t_1, t_2]$ if it reaches it at time t_1 and leaves it at time t_2.
- Two temporal paths intersect if they occupy the same vertex during non-disjoint intervals. They are temporally disjoint if they do not intersect.
- Two vertices are temporally connected if there is a temporal path between them.
- A temporal antichain is a set of vertices who are pairwise not temporally connected.



Our incentive for temporally disjoint paths

History

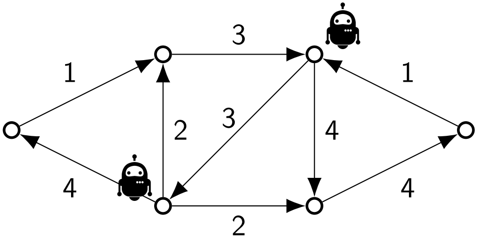
- Several papers on paths and journeys in temporal graphs
- Temporally disjoint paths are a good model for dynamic MULTI AGENT PATH FINDING [Stern *et al.*, 2019]



Our incentive for temporally disjoint paths

History

- Several papers on paths and journeys in temporal graphs
- Temporally disjoint paths are a good model for dynamic MULTI AGENT PATH FINDING [Stern *et al.*, 2019]



- TEMPORALLY DISJOINT WALKS: W[1]-hard and XP (number of walks) [Klobas *et al.*, IJCAI 2021]
- TEMPORALLY DISJOINT PATHS: NP-hard and W[1]-hard (number of vertices) on temporal stars [Kunz, Molter & Zehavi, IJCAI 2023]

A temporal Dilworth's theorem?

Dilworth property

In a (transitive) DAG, the minimum size of a path partition/cover is equal to the maximum size of a antichain.

A temporal Dilworth's theorem?

Temporal Dilworth property

In a temporal DAG, the minimum size of a temporal path partition/cover is equal to the maximum size of a temporal antichain.

A temporal Dilworth's theorem?

Temporal Dilworth property

In a temporal DAG, the minimum size of a temporal path partition/cover is equal to the maximum size of a temporal antichain.

Two problems:

Temporal Path Cover (TPC)

Temporal Path Partition/Temporally Disjoint Path Cover (TD-PC)

A temporal Dilworth's theorem?

Temporal Dilworth property

In a temporal DAG, the minimum size of a temporal path partition/cover is equal to the maximum size of a temporal antichain.

Two problems:

Temporal Path Cover (TPC)

Temporal Path Partition/Temporally Disjoint Path Cover (TD-PC)

Two questions:

Which temporal DAGs have the Dilworth property?

$\Rightarrow$ **Combinatorial** aspect

What is the complexity of those problems?

$\Rightarrow$ **Algorithmic** aspect

Our results

Temporal class	TPC	TD-PC
Oriented paths	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$
Rooted trees	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n^2)$
Oriented trees	$\mathcal{O}(\ell n^2 + n^3)$	NP-hard
DAGs*	NP-hard	NP-hard
Digraphs	XP (tw and $t_{\max}$) $n^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))}$	FPT (tw and $t_{\max}$) $2^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))} n$

* planar, subcubic, bipartite, girth 10, $\ell = 1$, $t_{\max} = 2$

$$n = \text{number of vertices}$$
$$\ell = \text{number of (unsorted) time labels per arc}$$
$$t_{\max} = \text{total number of time-steps}$$

For those specific classes, polynomial-time $\Leftrightarrow$ Dilworth property.

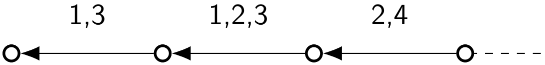
Temporal lines

Theorem [CDFK, 2024+]

Temporal oriented lines have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n)$.

Algorithm

Take a maximum-length temporal path containing a leaf.



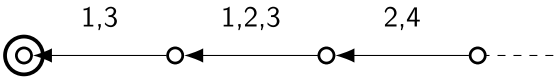
Temporal lines

Theorem [CDFK, 2024+]

Temporal oriented lines have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n)$.

Algorithm

Take a maximum-length temporal path containing a leaf.



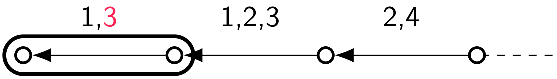
Temporal lines

Theorem [CDFK, 2024+]

Temporal oriented lines have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n)$.

Algorithm

Take a maximum-length temporal path containing a leaf.



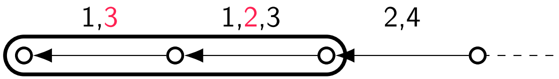
Temporal lines

Theorem [CDFK, 2024+]

Temporal oriented lines have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n)$.

Algorithm

Take a maximum-length temporal path containing a leaf.



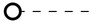
Temporal lines

Theorem [CDFK, 2024+]

Temporal oriented lines have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n)$.

Algorithm

Take a maximum-length temporal path containing a leaf.

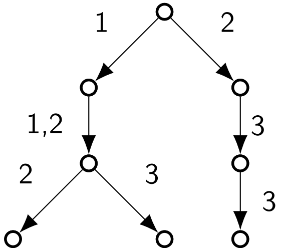


Iterate. The successive leaves are a temporal antichain!

Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.



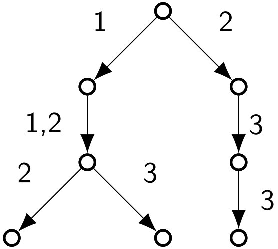
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines



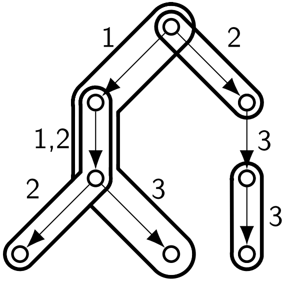
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines



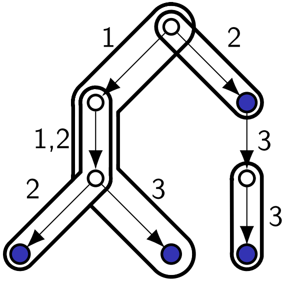
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines: successive leaves are a temporal antichain



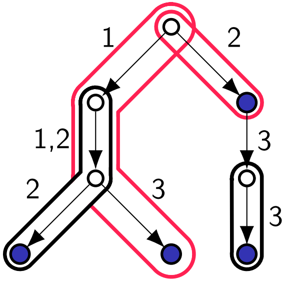
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines: successive leaves are a temporal antichain
- Starting from the root, resolve conflicts: if two paths intersect, either they start at the same vertex



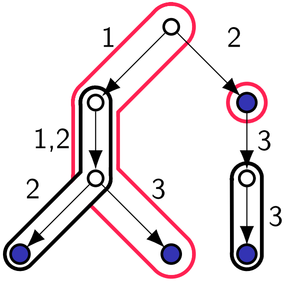
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines: successive leaves are a temporal antichain
- Starting from the root, resolve conflicts: if two paths intersect, either they start at the same vertex



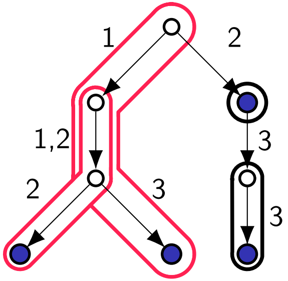
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

- Same principle as lines: successive leaves are a temporal antichain
- Starting from the root, resolve conflicts: if two paths intersect, either they start at the same vertex, or one starts before (in the underlying tree) the other



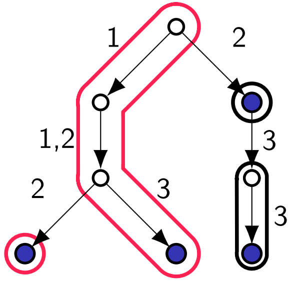
Temporal rooted trees

Theorem [CDFK, 2024+]

Temporal rooted trees have the Dilworth property, and we can solve TPC and TD-PC in time $\mathcal{O}(\ell n^2)$.

Algorithm

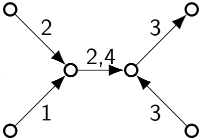
- Same principle as lines: successive leaves are a temporal antichain
- Starting from the root, resolve conflicts: if two paths intersect, either they start at the same vertex, or one starts before (in the underlying tree) the other



Path Cover of temporal oriented trees (1) Back to static

Theorem [CDFK, 2024+]

Temporal oriented trees have the Dilworth property for TPC, and we can solve TPC in time $\mathcal{O}(\ell n^2 + n^3)$.

$\mathcal{T} =$ 

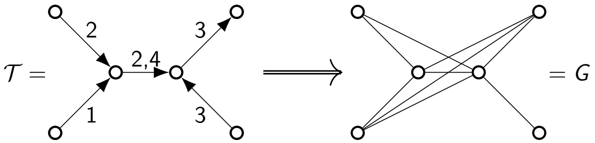
Path Cover of temporal oriented trees (1) Back to static

Theorem [CDFK, 2024+]

Temporal oriented trees have the Dilworth property for TPC, and we can solve TPC in time $\mathcal{O}(\ell n^2 + n^3)$.

Algorithm

- Construct an auxiliary connectivity graph: two vertices are adjacent $\Leftrightarrow$ they are temporally connected in the tree



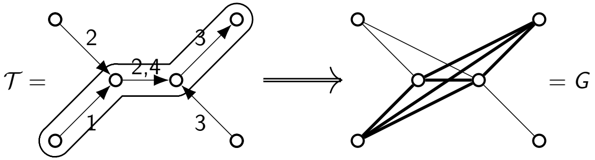
Path Cover of temporal oriented trees (1) Back to static

Theorem [CDFK, 2024+]

Temporal oriented trees have the Dilworth property for TPC, and we can solve TPC in time $\mathcal{O}(\ell n^2 + n^3)$.

Algorithm

- Construct an auxiliary connectivity graph: two vertices are adjacent $\Leftrightarrow$ they are temporally connected in the tree



Lemma

Clique in G $\Leftrightarrow$ Temporal Path in $\mathcal{T}$.

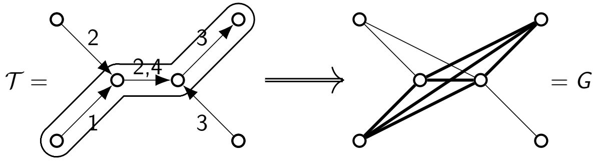
Path Cover of temporal oriented trees (1) Back to static

Theorem [CDFK, 2024+]

Temporal oriented trees have the Dilworth property for TPC, and we can solve TPC in time $\mathcal{O}(\ell n^2 + n^3)$.

Algorithm

- Construct an auxiliary connectivity graph: two vertices are adjacent $\Leftrightarrow$ they are temporally connected in the tree



Lemma

Clique Cover in G $\Leftrightarrow$ Temporal Path Cover in $\mathcal{T}$.

Path Cover of temporal oriented trees (2) Holes

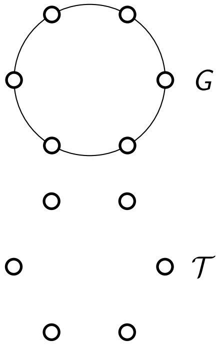
Lemma

There are no holes in the connectivity graph.

Path Cover of temporal oriented trees (2) Holes

Lemma

There are no holes in the connectivity graph.



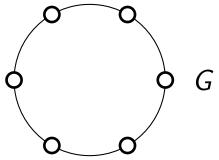
Path Cover of temporal oriented trees (2) Holes

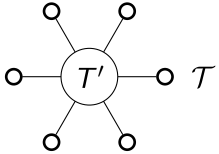
Lemma

There are no holes in the connectivity graph.

Claim

The vertices of the hole are leaves of a connected subtree T'.





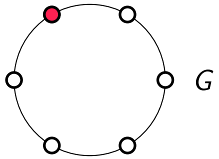
Path Cover of temporal oriented trees (2) Holes

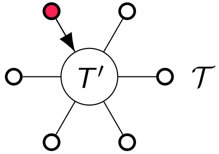
Lemma

There are no holes in the connectivity graph.

Claim

The vertices of the hole are leaves of a connected subtree T'.





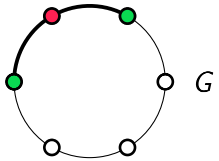
Path Cover of temporal oriented trees (2) Holes

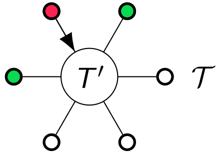
Lemma

There are no holes in the connectivity graph.

Claim

The vertices of the hole are leaves of a connected subtree T'.





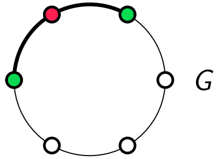
Path Cover of temporal oriented trees (2) Holes

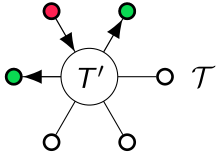
Lemma

There are no holes in the connectivity graph.

Claim

The vertices of the hole are leaves of a connected subtree T'.





Path Cover of temporal oriented trees (2) Holes

Lemma

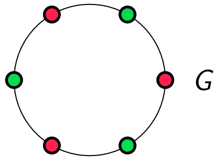
There are no holes in the connectivity graph.

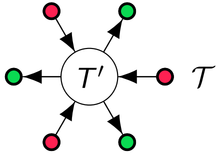
Claim

The vertices of the hole are leaves of a connected subtree T'.

Claim

Alternating between in-arcs and out-arcs from and to T'.





Path Cover of temporal oriented trees (2) Holes

Lemma

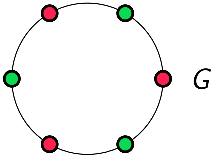
There are no holes in the connectivity graph.

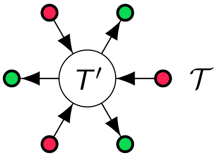
Claim

The vertices of the hole are leaves of a connected subtree T'.

Claim

Alternating between in-arcs and out-arcs from and to T'. $\Rightarrow$ No odd hole





Path Cover of temporal oriented trees (2) Holes

Lemma

There are no holes in the connectivity graph.

Claim

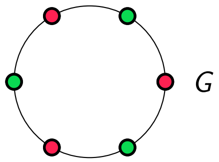
The vertices of the hole are leaves of a connected subtree T'.

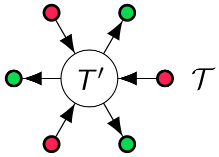
Claim

Alternating between in-arcs and out-arcs from and to T'. $\Rightarrow$ No odd hole

Claim

No even hole either (using Helly property and vertex-intersection of temporal paths).





Path Cover of temporal oriented trees (3) Antiholes

Lemma

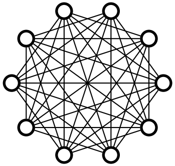
There are no antiholes in the connectivity graph.

Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

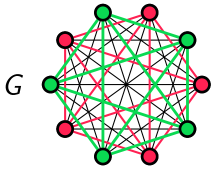
G



Path Cover of temporal oriented trees (3) Antiholes

Lemma

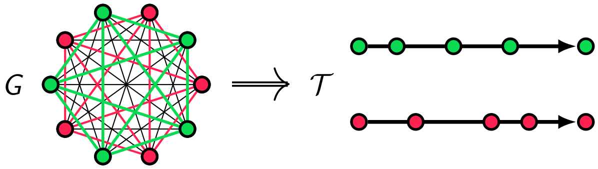
There are no antiholes in the connectivity graph.



Path Cover of temporal oriented trees (3) Antiholes

Lemma

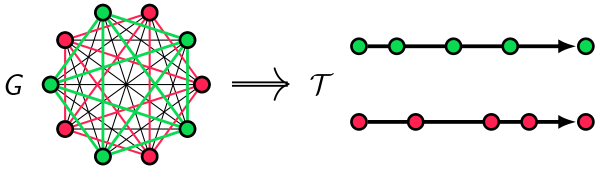
There are no antiholes in the connectivity graph.



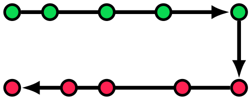
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



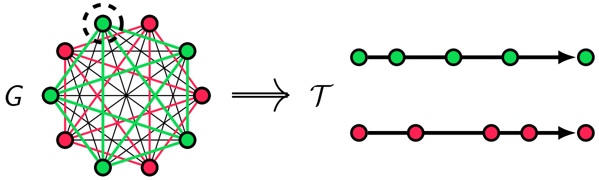
Case 1

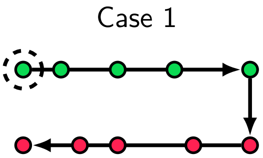


Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

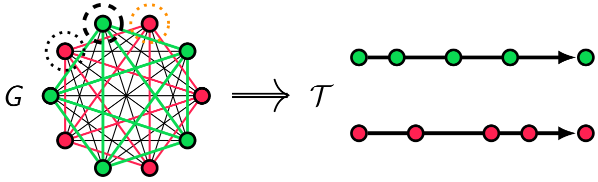




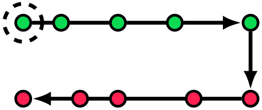
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



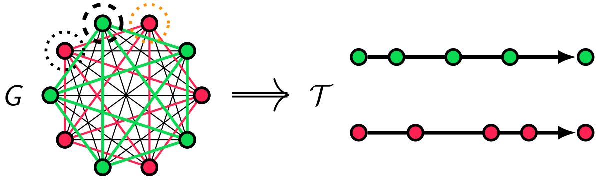
Case 1



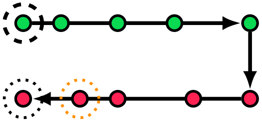
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



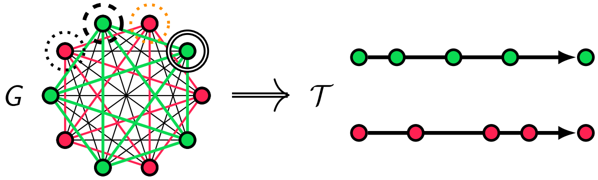
Case 1



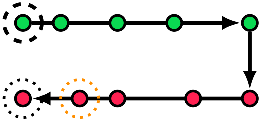
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



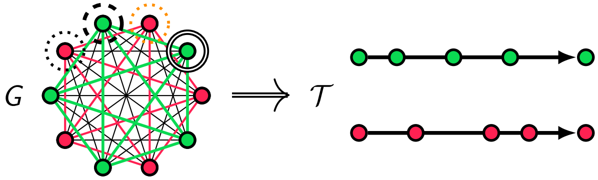
Case 1



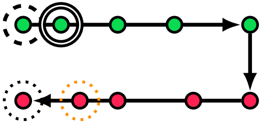
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



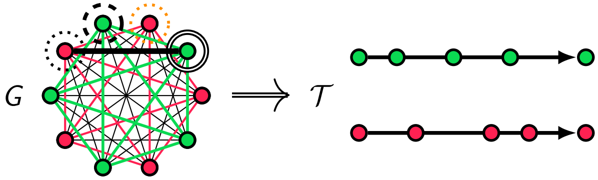
Case 1



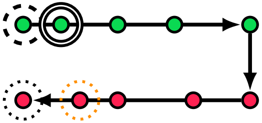
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.



Case 1

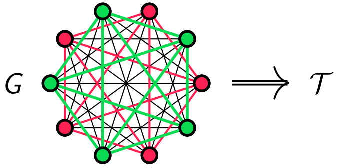


$\Rightarrow$ The temporal path of this edge does not exist in $\mathcal{T}$.

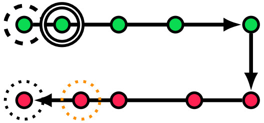
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

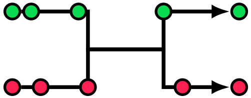


Case 1



$\Rightarrow$ The temporal path of this edge does not exist in $\mathcal{T}$.

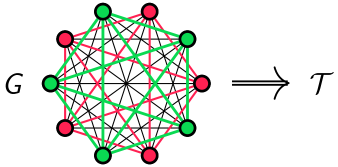
Case 2



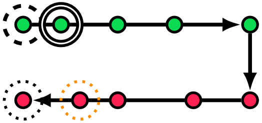
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

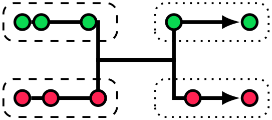


Case 1



$\Rightarrow$ The temporal path of this edge does not exist in $\mathcal{T}$.

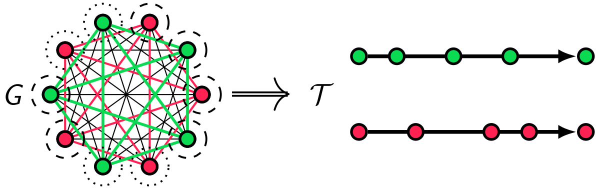
Case 2



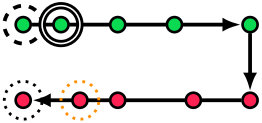
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

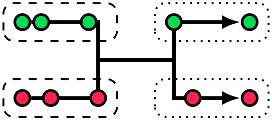


Case 1



$\Rightarrow$ The temporal path of this edge does not exist in $\mathcal{T}$.

Case 2

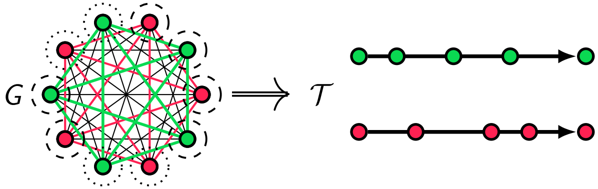


$\Rightarrow$ Each has to be complete bipartite in G

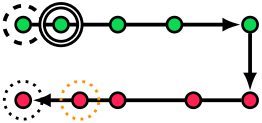
Path Cover of temporal oriented trees (3) Antiholes

Lemma

There are no antiholes in the connectivity graph.

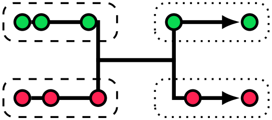


Case 1



⇒ The temporal path of this edge does not exist in $\mathcal{T}$.

Case 2



⇒ Each has to be complete bipartite in G, only possible if order $\leq$ 7, which we then manage.

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect)

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

Theorem [Hayward, Spinrad & Sritharan, 2000]

There is a $\mathcal{O}(mn)$ algorithm for Clique Cover in weakly chordal graphs with n vertices and m edges.

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

Theorem [Hayward, Spinrad & Sritharan, 2000]

There is a $\mathcal{O}(mn)$ algorithm for Clique Cover in weakly chordal graphs with n vertices and m edges.

$\Rightarrow$ Connectivity graph in $\mathcal{O}(n^2\ell)$, then [HSS00] in $\mathcal{O}(n^2 \times n)$.

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

Theorem [Hayward, Spinrad & Sritharan, 2000]

There is a $\mathcal{O}(mn)$ algorithm for Clique Cover in weakly chordal graphs with n vertices and m edges.

$\Rightarrow$ Connectivity graph in $\mathcal{O}(n^2\ell)$, then [HSS00] in $\mathcal{O}(n^2 \times n)$.

Remark

The connectivity graph is not chordal (it may contain C_4).

Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

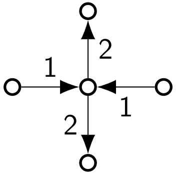
Theorem [Hayward, Spinrad & Sritharan, 2000]

There is a $\mathcal{O}(mn)$ algorithm for Clique Cover in weakly chordal graphs with n vertices and m edges.

$\Rightarrow$ Connectivity graph in $\mathcal{O}(n^2\ell)$, then [HSS00] in $\mathcal{O}(n^2 \times n)$.

Remark

The connectivity graph is not chordal (it may contain C_4).



Path Cover of temporal oriented trees (4) Conclusion

Lemmas

The connectivity graph is (hole,antihole)-free $\Rightarrow$ It is **weakly chordal** (subclass of perfect) $\Rightarrow$ Dilworth property!

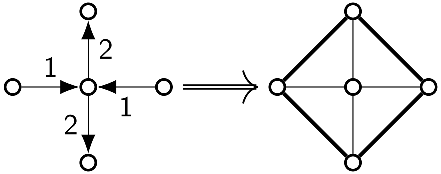
Theorem [Hayward, Spinrad & Sritharan, 2000]

There is a $\mathcal{O}(mn)$ algorithm for Clique Cover in weakly chordal graphs with n vertices and m edges.

$\Rightarrow$ Connectivity graph in $\mathcal{O}(n^2\ell)$, then [HSS00] in $\mathcal{O}(n^2 \times n)$.

Remark

The connectivity graph is not chordal (it may contain C_4).



Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from Unary Bin Packing (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B

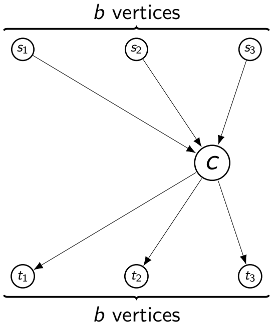
Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from UNARY BIN PACKING (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B



Each $(s_i, t_i) \equiv$ one bin

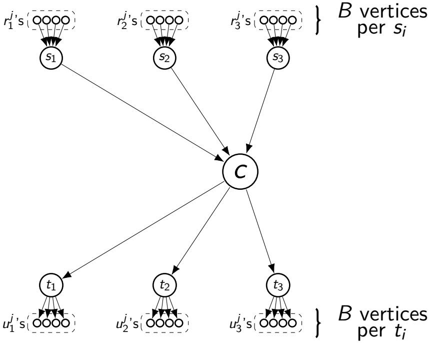
Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from Unary Bin Packing (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B



Each $(s_i, t_i) \equiv$ one bin

Each bin must be filled

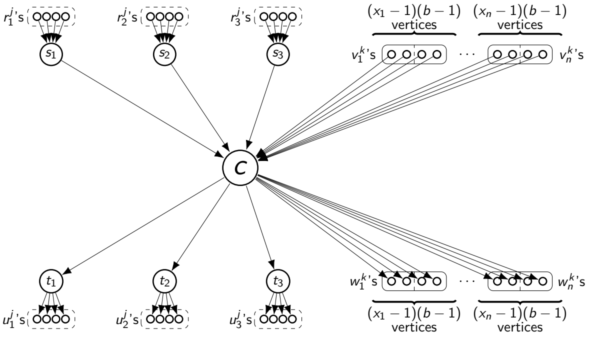
Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from Unary Bin Packing (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B



Each $(s_i, t_i) \equiv$ one bin

Each bin must be filled

$(v_i, w_i) \equiv$ bins unused by item i

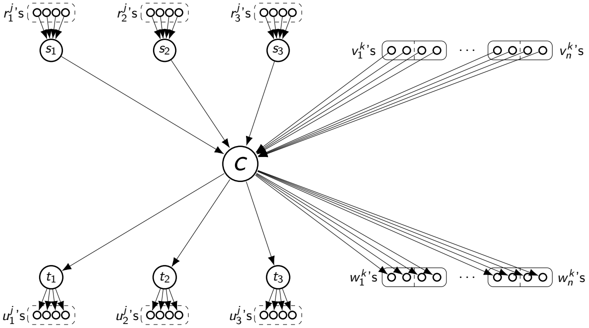
Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from Unary Bin Packing (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B



Each $(s_i, t_i) \equiv$ one bin

Each bin must be filled

$(v_i, w_i) \equiv$ bins unused by item i

We create **temporal layers** for each item i

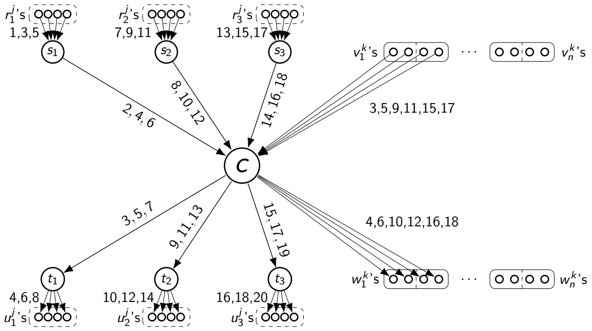
Path Partition of temporal oriented trees: NP-hardness

Theorem [CDFK, 2024+]

TD-PC is NP-hard on temporal oriented trees.

Reduction from UNARY BIN PACKING (inspired by [KMZ23])

Items of size $x_1, \ldots, x_n$; b bins of size B



Each $(s_i, t_i) \equiv$ one bin

Each bin must be filled

$(v_i, w_i) \equiv$ bins unused by item i

We create **temporal layers** for each item i

Here, $x_1 = 3$

What about the Dilworth property in those NP-hard cases?

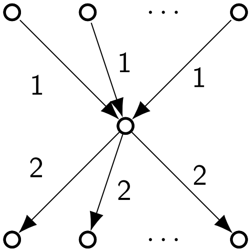
Theorem [CDFK, 2024+]

Temporal oriented trees do not have the TD-Dilworth property.

What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

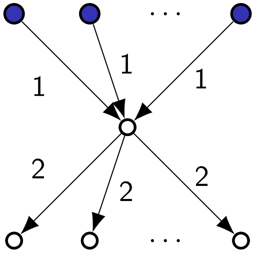
Temporal oriented trees do not have the TD-Dilworth property.



What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

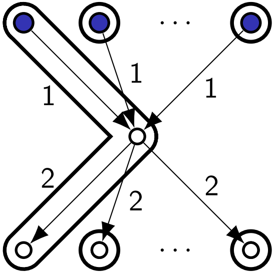
Temporal oriented trees do not have the TD-Dilworth property.



What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

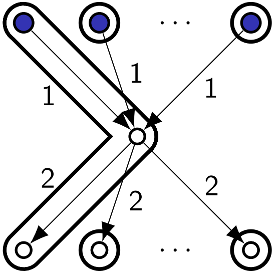
Temporal oriented trees do not have the TD-Dilworth property.



What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

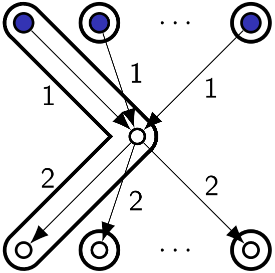
Temporal oriented trees do not have the TD-Dilworth property. Furthermore, the gap between antichain and partition can be arbitrarily large in temporal DAGs.

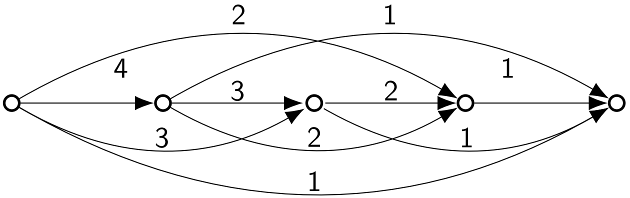


What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

Temporal oriented trees do not have the TD-Dilworth property. Furthermore, the gap between antichain and partition can be arbitrarily large in temporal DAGs.

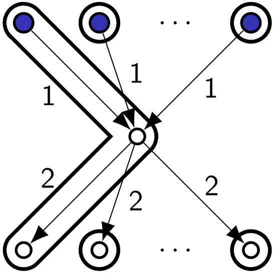


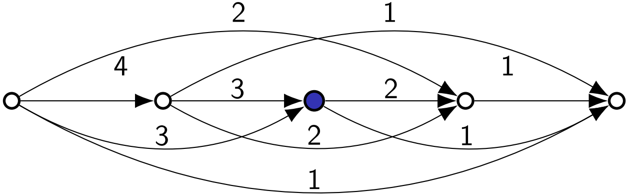


What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

Temporal oriented trees do not have the TD-Dilworth property. Furthermore, the gap between antichain and partition can be arbitrarily large in temporal DAGs.

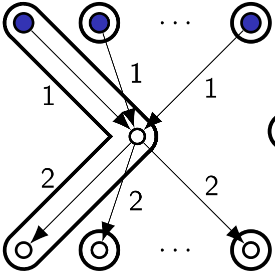


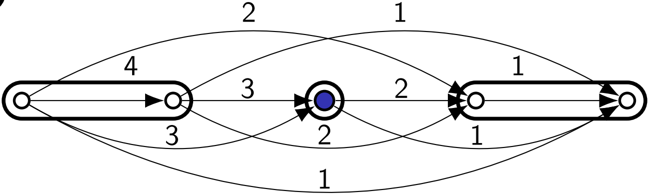


What about the Dilworth property in those NP-hard cases?

Theorem [CDFK, 2024+]

Temporal oriented trees do not have the TD-Dilworth property. Furthermore, the gap between antichain and partition can be arbitrarily large in temporal DAGs.





Parameterized complexity: reminder

Definition

For an input of size n with a parameter k and a computable function f:

- FPT algorithm $\Leftrightarrow f(k)n^{\mathcal{O}(1)}$
- XP algorithm $\Leftrightarrow n^{f(k)}$

Parameterized complexity: reminder

Definition

For an input of size n with a parameter k and a computable function f:

- FPT algorithm $\Leftrightarrow f(k)n^{\mathcal{O}(1)}$
- XP algorithm $\Leftrightarrow n^{f(k)}$

Parameter k bounded $\Rightarrow$ Efficient algorithm

Parameterized complexity: reminder

Definition

For an input of size n with a parameter k and a computable function f:

- FPT algorithm $\Leftrightarrow f(k)n^{\mathcal{O}(1)}$
- XP algorithm $\Leftrightarrow n^{f(k)}$

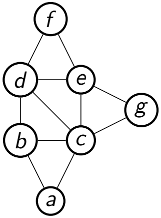
Parameter k bounded $\Rightarrow$ Efficient algorithm

Commonly used method

Decomposing an input graph and applying dynamic programming

Parameterized complexity: tree decompositions

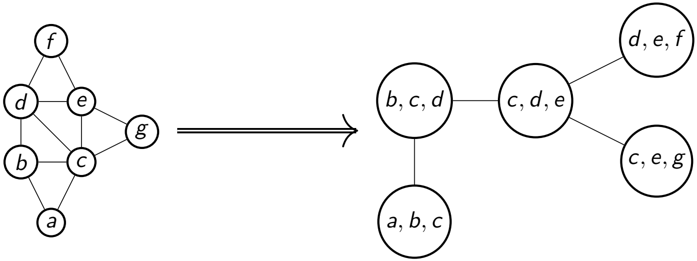
Tree decomposition of $G(V, E)$ [Halin, 1976] and others



Parameterized complexity: tree decompositions

Tree decomposition of $G(V, E)$ [Halin, 1976] and others

A tree T, each node $v \in$ T has a *bag* $X_v \subseteq V$, such that:

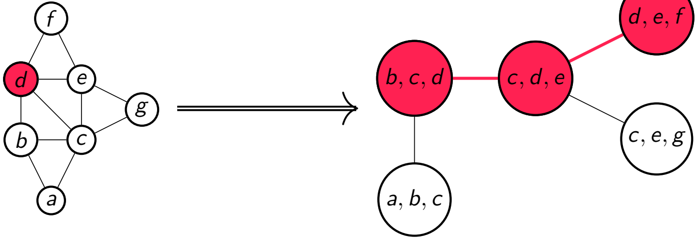


Parameterized complexity: tree decompositions

Tree decomposition of $G(V, E)$ [Halin, 1976] and others

A tree T, each node $v \in$ T has a *bag* $X_v \subseteq V$, such that:

- for $a \in V$, $\{v \;:\; a \in X_v\}$ is a connected subtree of T

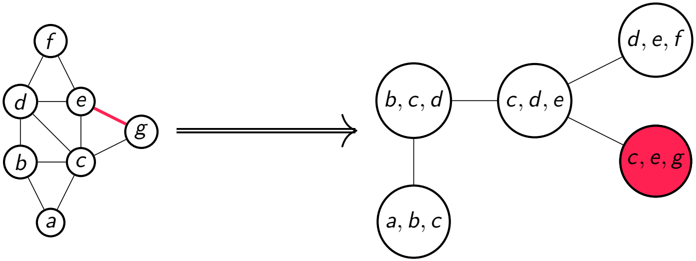


Parameterized complexity: tree decompositions

Tree decomposition of $G(V, E)$ [Halin, 1976] and others

A tree T, each node $v \in$ T has a *bag* $X_v \subseteq V$, such that:

- for $a \in V$, $\{v \ : \ a \in X_v\}$ is a connected subtree of T
- if $ab \in E$, then $\exists v$ such that $a, b \in X_v$



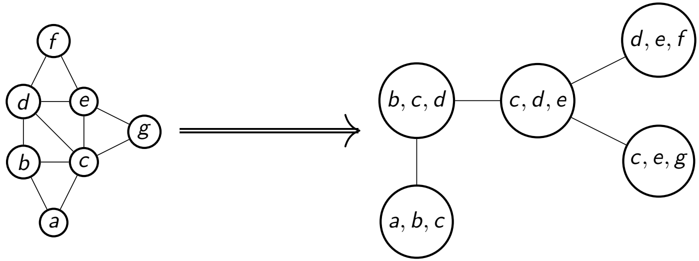
Parameterized complexity: tree decompositions

Tree decomposition of $G(V, E)$ [Halin, 1976] and others

A tree T, each node $v \in$ T has a *bag* $X_v \subseteq V$, such that:

- for $a \in V$, $\{v \ : \ a \in X_v\}$ is a connected subtree of T
- if $ab \in E$, then $\exists v$ such that $a, b \in X_v$

tw = max size of a bag -1



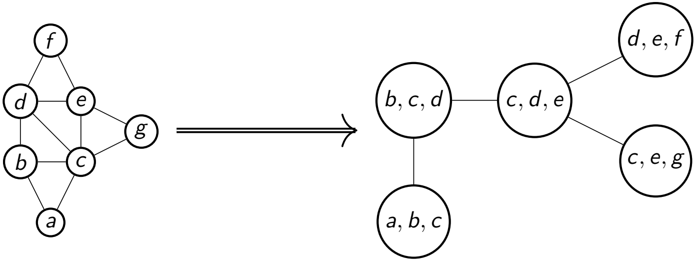
Parameterized complexity: tree decompositions

Tree decomposition of $G(V, E)$ [Halin, 1976] and others

A tree T, each node $v \in$ T has a *bag* $X_v \subseteq V$, such that:

- for $a \in V$, $\{v \; : \; a \in X_v\}$ is a connected subtree of T
- if $ab \in E$, then $\exists v$ such that $a, b \in X_v$

tw = max size of a bag -1



- Computation of a tree decomposition of width 2 tw in time $2^{\mathcal{O}(\text{tw})} n$ [Korhonen, 2021]

Parameterized complexity: nice tree decompositions

Nice tree decomposition of $G(V, E)$ [Kloks, 1994]

T is rooted, leaves and root bags are empty, inner nodes are:

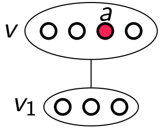
Parameterized complexity: nice tree decompositions

Nice tree decomposition of $G(V, E)$ [Kloks, 1994]

T is rooted, leaves and root bags are empty, inner nodes are:

- *introduce* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \cup a$, $a \in X_v$

introduce



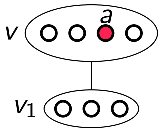
Parameterized complexity: nice tree decompositions

Nice tree decomposition of $G(V, E)$ [Kloks, 1994]

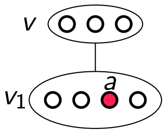
T is rooted, leaves and root bags are empty, inner nodes are:

- *introduce* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \cup a$, $a \in X_v$
- *forget* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \setminus a$, $a \in X_{v_1}$

introduce



forget

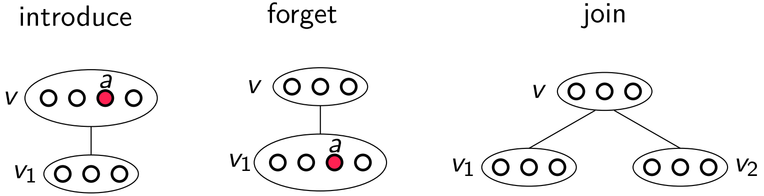


Parameterized complexity: nice tree decompositions

Nice tree decomposition of $G(V, E)$ [Kloks, 1994]

T is rooted, leaves and root bags are empty, inner nodes are:

- *introduce* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \cup a$, $a \in X_v$
- *forget* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \setminus a$, $a \in X_{v_1}$
- *join* node $v \Leftrightarrow$ two children v_1, v_2 s.t. $X_v = X_{v_1} = X_{v_2}$

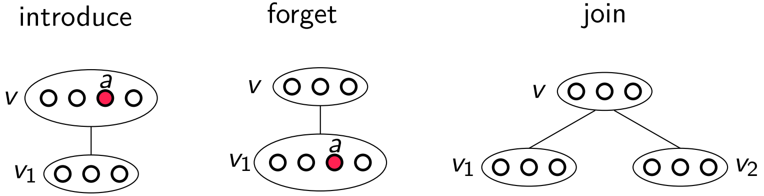


Parameterized complexity: nice tree decompositions

Nice tree decomposition of $G(V, E)$ [Kloks, 1994]

T is rooted, leaves and root bags are empty, inner nodes are:

- *introduce* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \cup a$, $a \in X_v$
- *forget* node $v \Leftrightarrow$ one child v_1 s.t. $X_v = X_{v_1} \setminus a$, $a \in X_{v_1}$
- *join* node $v \Leftrightarrow$ two children v_1, v_2 s.t. $X_v = X_{v_1} = X_{v_2}$



- From a tree decomposition, a nice one with same width and $\mathcal{O}(\text{tw}\, n)$ bags can be computed in time $\mathcal{O}(\text{tw}^2\, n)$ [Kloks, 1994]

Parameterized complexity for TD-PC (1) Preliminaries

Theorem [CDFK, 2024+]

TD-PC is FPT w.r.t. tw and t_{max} (total number of time-steps)

Dynamic programing on a nice tree decomposition

Parameterized complexity for TD-PC (1) Preliminaries

Theorem [CDFK, 2024+]

TD-PC is FPT w.r.t. tw and $t_{\max}$ (total number of time-steps)

Dynamic programing on a nice tree decomposition

Observation

Any arc of $\mathcal{D}$ appears in at most $t_{\max}$ paths of a TD-PC $\Rightarrow$ At most $p = \binom{\text{tw}}{2} \cdot t_{\max}$ temporally disjoint paths contain at least one arc from a given bag

Parameterized complexity for TD-PC (1) Preliminaries

Theorem [CDFK, 2024+]

TD-PC is FPT w.r.t. tw and $t_{\max}$ (total number of time-steps)

Dynamic programing on a nice tree decomposition

Observation

Any arc of $\mathcal{D}$ appears in at most $t_{\max}$ paths of a TD-PC $\Rightarrow$ At most $p = \binom{\text{tw}}{2} \cdot t_{\max}$ temporally disjoint paths contain at least one arc from a given bag

For simplicity, duplicate the arcs such that each has only one time label (so a TD-PC uses arc-disjoint paths)

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

$\Rightarrow$ At most
types for any node

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

- A partition $Q_0, Q_1, \ldots, Q_t$ of the arcs inside X_v (Q_i for $i \neq 0$ is in a temporal path P_i of a TD-PC, Q_0 is the unused arcs)

$\Rightarrow$ At most p^p
types for any node

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

- A partition $Q_0, Q_1, \ldots, Q_t$ of the arcs inside X_v (Q_i for $i \neq 0$ is in a temporal path P_i of a TD-PC, Q_0 is the unused arcs)
- For each Q_i, the vertices V_i of X_v that are in P_i (endpoints of arcs in Q_i and those not incident with arcs in Q_i)

$\Rightarrow$ At most $p^p \times 2^{\mathrm{tw}+1}$
types for any node

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

- A partition $Q_0, Q_1, \dots, Q_t$ of the arcs inside X_v (Q_i for $i \neq 0$ is in a temporal path P_i of a TD-PC, Q_0 is the unused arcs)
- For each Q_i, the vertices V_i of X_v that are in P_i (endpoints of arcs in Q_i and those not incident with arcs in Q_i)
- For each V_i, their order of occupation by P_i

$\Rightarrow$ At most $p^p \times 2^{\mathrm{tw}+1} \times (\mathrm{tw}+1)!$
types for any node

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

- A partition $Q_0, Q_1, \ldots, Q_t$ of the arcs inside X_v (Q_i for $i \neq 0$ is in a temporal path P_i of a TD-PC, Q_0 is the unused arcs)
- For each Q_i, the vertices V_i of X_v that are in P_i (endpoints of arcs in Q_i and those not incident with arcs in Q_i)
- For each V_i, their order of occupation by P_i
- For each Q_i, the vertices in V_i with one or two arcs outside of X_v, the time labels of those arcs, and whether the neighbour appears below or above v in the decomposition

$\Rightarrow$ At most $p^p \times 2^{\text{tw}+1} \times (\text{tw}+1)! \times 2^{\text{tw}+2} \times t_{\max}^2$
types for any node

Parameterized complexity for TD-PC (2) Type

Type: necessary information at each node v

- A partition $Q_0, Q_1, \ldots, Q_t$ of the arcs inside X_v (Q_i for $i \neq 0$ is in a temporal path P_i of a TD-PC, Q_0 is the unused arcs)
- For each Q_i, the vertices V_i of X_v that are in P_i (endpoints of arcs in Q_i and those not incident with arcs in Q_i)
- For each V_i, their order of occupation by P_i
- For each Q_i, the vertices in V_i with one or two arcs outside of X_v, the time labels of those arcs, and whether the neighbour appears below or above v in the decomposition

$\Rightarrow$ At most $p^p \times 2^{\mathrm{tw}+1} \times (\mathrm{tw}+1)! \times 2^{\mathrm{tw}+2} \times t_{\max}^2 \in 2^{\mathcal{O}(p \log p)}$ types for any node

Parameterized complexity for TD-PC (3) Consistency

Consistency of a type

- The ordered vertices V_i, the arcs of Q_i, and the information about the arcs going outside of X_v, induce temporal paths
- The arcs going outside of X_v exist in the digraph and their labels are compatible with the order
- Every vertex of X_v is in a V_i

Now, we compute from the bottom-up, maintaining consistency.

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either *a* is in a path in the type, or *a* is added as a single-vertex path)

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either *a* is in a path in the type, or *a* is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex *a*), discard those where *a* has an arc going above

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either *a* is in a path in the type, or *a* is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex *a*), discard those where *a* has an arc going above
- **Join node:** Check compatibility of the children (partition of arcs, order of vertices, neighbours outside of the bag, are they above or below in the decomposition, ... $\Rightarrow$ all have to agree), don't count twice the paths that intersect the bag

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either a is in a path in the type, or a is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex a), discard those where a has an arc going above
- **Join node:** Check compatibility of the children (partition of arcs, order of vertices, neighbours outside of the bag, are they above or below in the decomposition, ... $\Rightarrow$ all have to agree), don't count twice the paths that intersect the bag

Running time $2^{\mathcal{O}(p \log p)} n$, so FPT w.r.t. $p = f(\text{tw}, t_{\max})$

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either a is in a path in the type, or a is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex a), discard those where a has an arc going above
- **Join node:** Check compatibility of the children (partition of arcs, order of vertices, neighbours outside of the bag, are they above or below in the decomposition, ... $\Rightarrow$ all have to agree), don't count twice the paths that intersect the bag

Running time $2^{\mathcal{O}(p \log p)}n$, so FPT w.r.t. $p = f(\text{tw}, t_{\max})$

And for TPC?

Same principle, but the paths can intersect

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either a is in a path in the type, or a is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex a), discard those where a has an arc going above
- **Join node:** Check compatibility of the children (partition of arcs, order of vertices, neighbours outside of the bag, are they above or below in the decomposition, ... $\Rightarrow$ all have to agree), don't count twice the paths that intersect the bag

Running time $2^{\mathcal{O}(p \log p)} n$, so FPT w.r.t. $p = f(\text{tw}, t_{\max})$

And for TPC?

Same principle, but the paths can intersect $\Rightarrow$ More information in type: how many times in the solution does Q_i appear

Parameterized complexity for TD-PC (4) Computation

Dynamic programing using consistent types of partial solutions

- **Leaf node:** No partial solution since empty
- **Introduce node:** Check compatibility with the child (either a is in a path in the type, or a is added as a single-vertex path)
- **Forget node:** Check compatibility with child (the types are the ones obtained by removing the vertex a), discard those where a has an arc going above
- **Join node:** Check compatibility of the children (partition of arcs, order of vertices, neighbours outside of the bag, are they above or below in the decomposition, ... $\Rightarrow$ all have to agree), don't count twice the paths that intersect the bag

Running time $2^{\mathcal{O}(p \log p)} n$, so FPT w.r.t. $p = f(\text{tw}, t_{\max})$

And for TPC?

Same principle, but the paths can intersect $\Rightarrow$ More information in type: how many times in the solution does Q_i appear $\Rightarrow$ Running time $k^{\mathcal{O}(p \log p)} n$ where $k \in \mathcal{O}(n)$ is the solution size $\Rightarrow$ XP w.r.t. p

Conclusion and future work

Temporal class	**TPC**	**TD-PC**
Oriented paths	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$
Rooted trees	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n^2)$
Oriented trees	$\mathcal{O}(\ell n^2 + n^3)$	NP-hard
DAGs	NP-hard	NP-hard
Digraphs	XP (tw and $t_{\max}$) $n^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))}$	FPT (tw and $t_{\max}$) $2^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))}\, n$

Conclusion and future work

Temporal class	**TPC**	**TD-PC**
Oriented paths	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$
Rooted trees	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n^2)$
Oriented trees	$\mathcal{O}(\ell n^2 + n^3)$	NP-hard
DAGs	NP-hard	NP-hard
Digraphs	XP (tw and $t_{\max}$) $n^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))}$	FPT (tw and $t_{\max}$) $2^{\mathcal{O}(\mathrm{tw}^2\, t_{\max} \log(\mathrm{tw}\, t_{\max}))} n$

Perspectives

- Better FPT, FPT for TPC?
- Approximation? Enumeration?
- Classes of oriented trees where TD-PC is polynomial?
- Other temporal problems that can be reduced to a static problem?

Conclusion and future work

Temporal class	TPC	TD-PC
Oriented paths	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$
Rooted trees	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n^2)$
Oriented trees	$\mathcal{O}(\ell n^2 + n^3)$	NP-hard
DAGs	NP-hard	NP-hard
Digraphs	XP (tw and $t_{\max}$) $n^{\mathcal{O}(\text{tw}^2\, t_{\max} \log(\text{tw}\, t_{\max}))}$	FPT (tw and $t_{\max}$) $2^{\mathcal{O}(\text{tw}^2\, t_{\max} \log(\text{tw}\, t_{\max}))}\, n$

Perspectives

- Better FPT, FPT for TPC?
- Approximation? Enumeration?
- Classes of oriented trees where TD-PC is polynomial?
- Other temporal problems that can be reduced to a static problem?