Sliding Window Temporal Graph Coloring

George B. Mertzios[1] Hendrik Molter[2] Viktor Zamaraev[1]

[1]Department of Computer Science, Durham University, Durham, UK

[2]Algorithmics and Computational Complexity, TU Berlin, Germany

Appeared at AAAI 2019

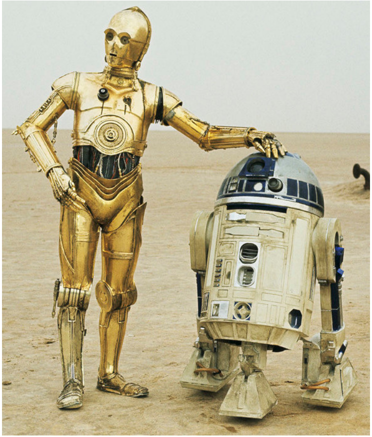
NeST Workshop, 12 June 2019
Department of Computer Science
Liverpool University

Main question:

- What is the natural problem that extends graph coloring to the case where the graph changes over time?

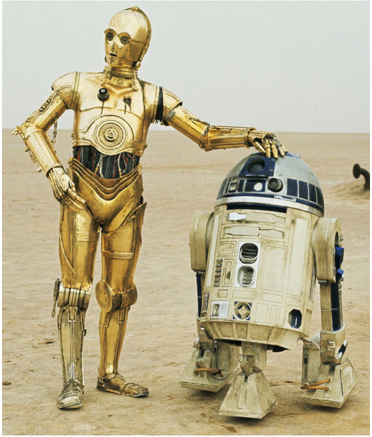
A motivating scenario:

- Mobile agents broadcast information



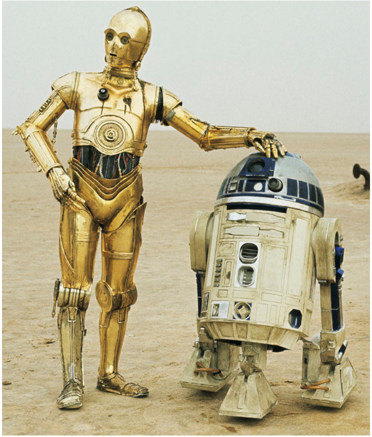
A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information



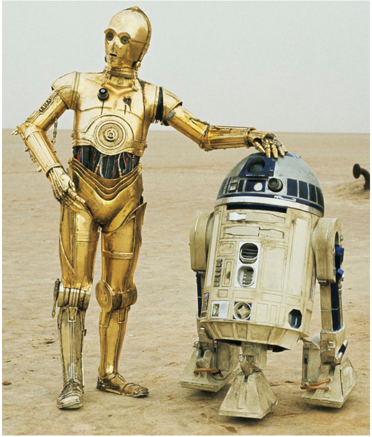
A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels



A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)



A motivating scenario:

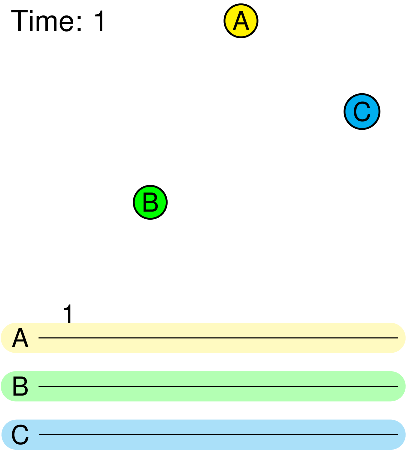
- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"

A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

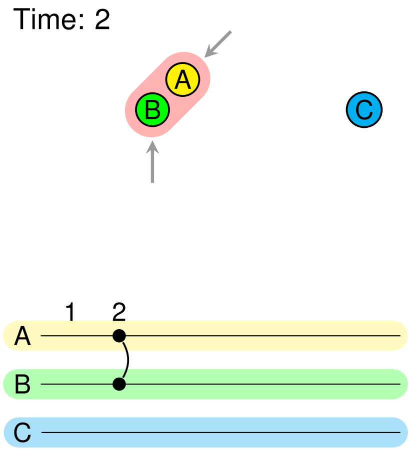
"Dynamic Channel Assignment Problem"



A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

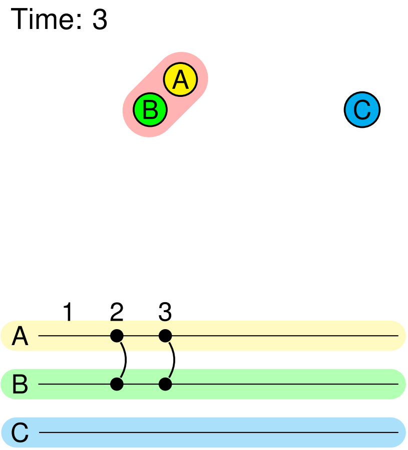
"Dynamic Channel Assignment Problem"



A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

“Dynamic Channel Assignment Problem”

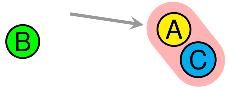


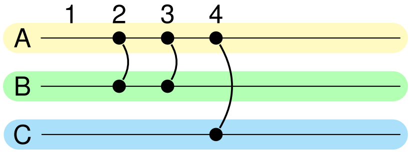
A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"

Time: 4

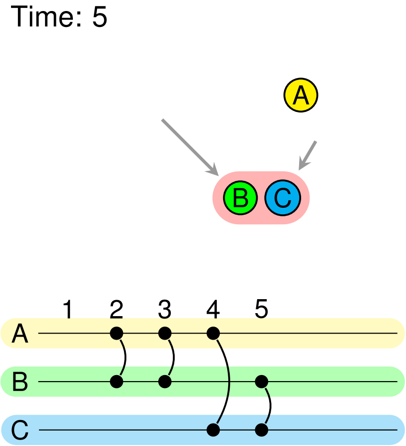




A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

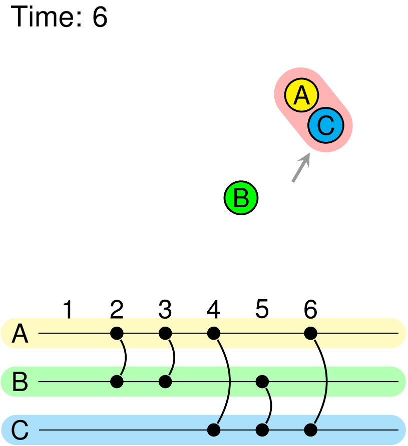
"Dynamic Channel Assignment Problem"



A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"



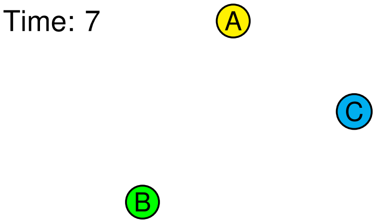
Introduction

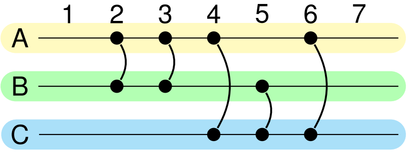
Motivation

A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"





A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

“Dynamic Channel Assignment Problem”

Modeling:

- Vertices in a temporal graph

Introduction

Motivation

A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"

Modeling:

- Vertices in a temporal graph
- Time-edges

Introduction

Motivation

A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"

Modeling:

- Vertices in a temporal graph
- Time-edges
- Vertices need to be **differently colored** in order to exchange information

A motivating scenario:

- Mobile agents broadcast information
- When agents meet they can exchange information
- Information can only be exchanged if agents broadcast on **different** channels
- Agents should be able to exchange information within reasonable time windows around their meetings (every Δ time)

"Dynamic Channel Assignment Problem"

Modeling:

- Vertices in a temporal graph
- Time-edges
- Vertices need to be **differently colored** in order to exchange information
- Each time-edge should be "properly colored" at least once in each Δ-window in which it exists

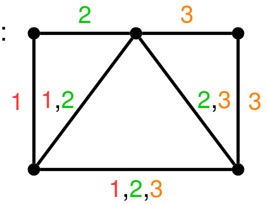
Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.

Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.

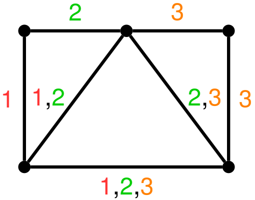
(G, λ):



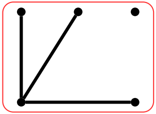
Temporal Graph

A **temporal graph** (G,λ) with lifetime T is a graph $G=(V,E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.

(G,λ):

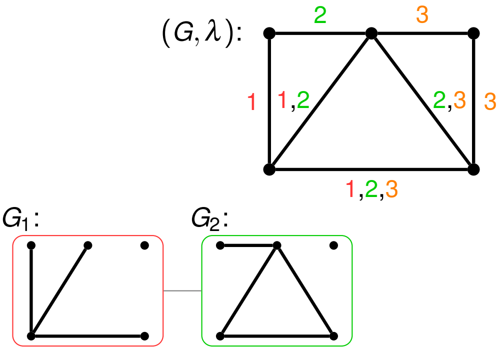


G_1:



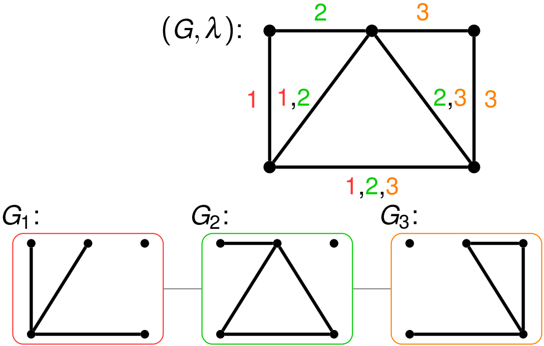
Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\dots,T\}}$ that assigns time labels to edges.



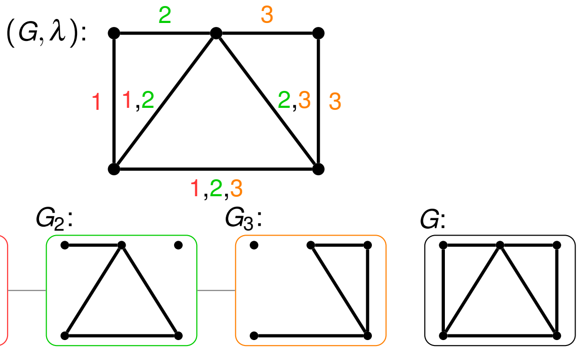
Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \rightarrow 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.



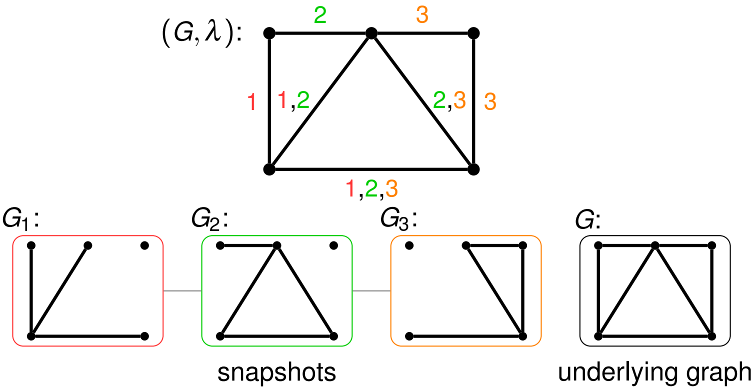
Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.



Temporal Graph

A **temporal graph** (G, λ) with lifetime T is a graph $G = (V, E)$ with a labeling function $\lambda : E \to 2^{\{1,2,\ldots,T\}}$ that assigns time labels to edges.



Proper Sliding Δ-Window Temporal Coloring

A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that:

Proper Sliding Δ-Window Temporal Coloring

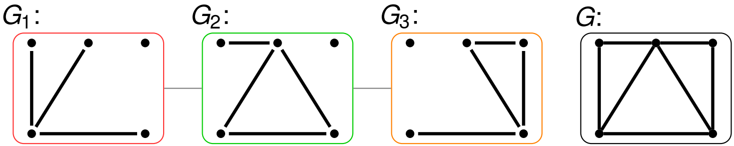
A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that:

for every Δ-time window W_t and for every edge $e \in E$, if $W_t \cap \lambda(e) \neq \emptyset$ then e is properly colored in at least one time slot $t \in W_t \cap \lambda(e)$.

Proper Sliding Δ-Window Temporal Coloring

A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \dots, \phi_T)$ such that:

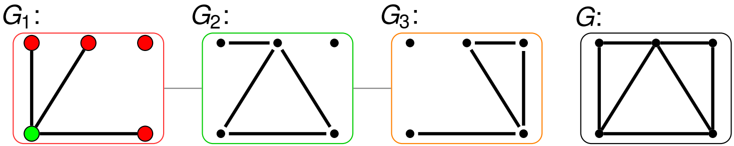
for every Δ-time window W_t and for every edge $e \in E$, if $W_t \cap \lambda(e) \neq \emptyset$ then e is properly colored in at least one time slot $t \in W_t \cap \lambda(e)$.



Proper Sliding Δ-Window Temporal Coloring

A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that:

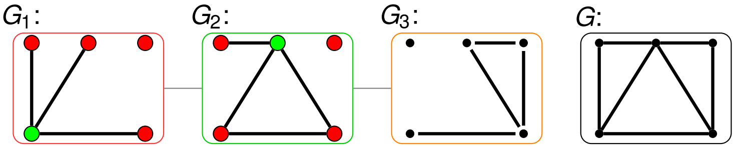
for every Δ-time window W_t and for every edge $e \in E$, if $W_t \cap \lambda(e) \neq \emptyset$ then e is properly colored in at least one time slot $t \in W_t \cap \lambda(e)$.



Proper Sliding Δ-Window Temporal Coloring

A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that:

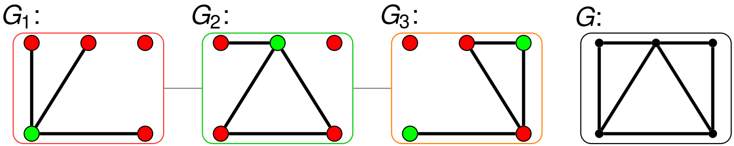
for every Δ-time window W_t and for every edge $e \in E$, if $W_t \cap \lambda(e) \neq \emptyset$ then e is properly colored in at least one time slot $t \in W_t \cap \lambda(e)$.



Proper Sliding Δ-Window Temporal Coloring

A proper sliding Δ-window temporal coloring of (G, λ) is a coloring vector $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that:

for every Δ-time window W_t and for every edge $e \in E$, if $W_t \cap \lambda(e) \neq \emptyset$ then e is properly colored in at least one time slot $t \in W_t \cap \lambda(e)$.



Sliding Window Temporal Coloring (SWTC)

Input: A temporal graph (G, λ), and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring ϕ of (G, λ) using at most k colors?

Sliding Window Temporal Coloring (SWTC)

Input: A temporal graph (G, λ), and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring ϕ of (G, λ) using at most k colors?

Main Hardness Results:

SWTC is NP-hard, even if

- $k = 2$, $\Delta = 2$, $T = 3$,
  G is 3-colorable, $O(1)$ max. deg.,
  every connected component in every snapshot has $O(1)$ size.

Sliding Window Temporal Coloring (SWTC)

Input: A temporal graph (G, λ), and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring ϕ of (G, λ) using at most k colors?

Main Hardness Results:

SWTC is NP-hard, even if

- $k = 2$, $\Delta = 2$, $T = 3$, G is 3-colorable, $O(1)$ max. deg., every connected component in every snapshot has $O(1)$ size.
- $k = 2$, $\Delta = 2$, vertex cover number of G is $O(1)$.

Main Algorithmic Results:

- Exp. time algorithm (asympt. optimal in n under ETH).

Sliding Window Temporal Coloring (SWTC)

Input: A temporal graph (G, λ), and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring ϕ of (G, λ) using at most k colors?

Main Hardness Results:

SWTC is NP-hard, even if

- $k = 2$, $\Delta = 2$, $T = 3$, G is 3-colorable, $O(1)$ max. deg., every connected component in every snapshot has $O(1)$ size.
- $k = 2$, $\Delta = 2$, vertex cover number of G is $O(1)$.

Main Algorithmic Results:

- Exp. time algorithm (asympt. optimal in n under ETH).
- Extension for small number of agents (FPT algorithm for n).

Sliding Window Temporal Coloring (SWTC)

Input: A temporal graph (G, λ), and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring ϕ of (G, λ) using at most k colors?

Main Hardness Results:

SWTC is NP-hard, even if

- $k = 2$, $\Delta = 2$, $T = 3$, G is 3-colorable, $O(1)$ max. deg., every connected component in every snapshot has $O(1)$ size.
- $k = 2$, $\Delta = 2$, vertex cover number of G is $O(1)$.

Main Algorithmic Results:

- Exp. time algorithm (asympt. optimal in n under ETH).
- Extension for small number of agents (FPT algorithm for n).
- FPT-approx. algorithm for parameter "vertex cover number of G" (additive error of one).

Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm I

Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

Observation

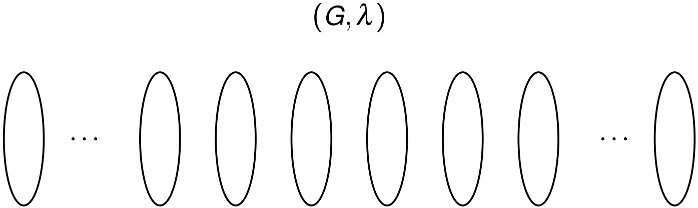
Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.

Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

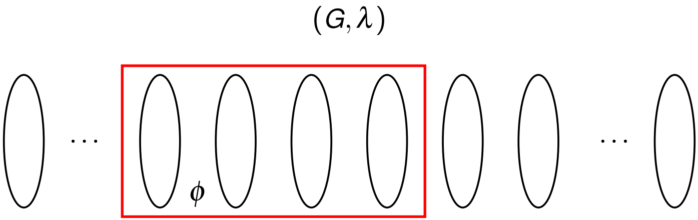
If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.



Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.



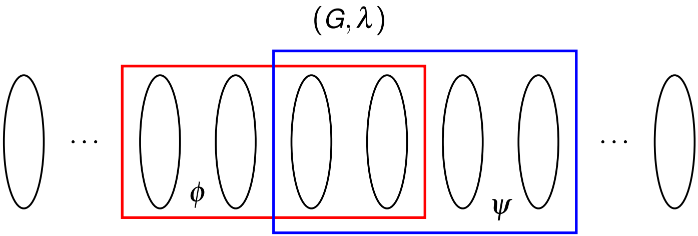
Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm I

Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.



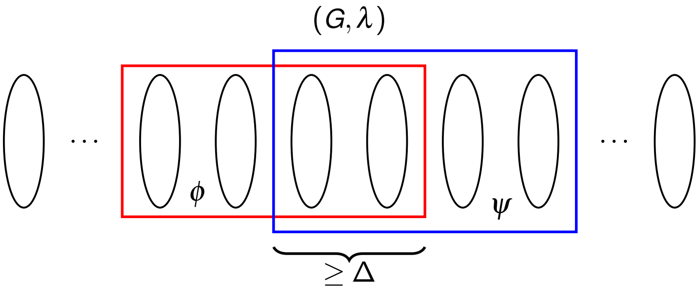
Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm I

Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.



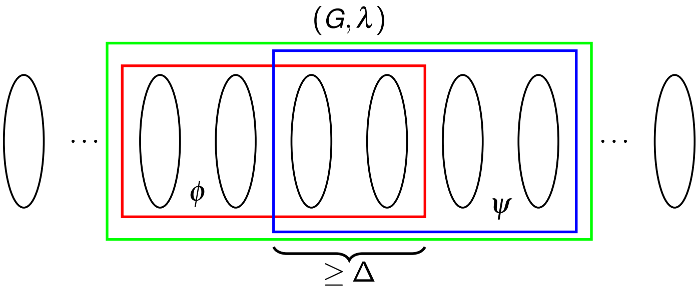
Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm I

Observation

Let ϕ and ψ be two proper sliding Δ-window temporal colorings for two intervals of the snapshots of (G, λ).

If they overlap on at least Δ time steps and agree how to color the overlap, they can be combined.

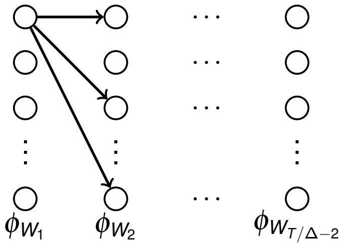


Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm II

1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.

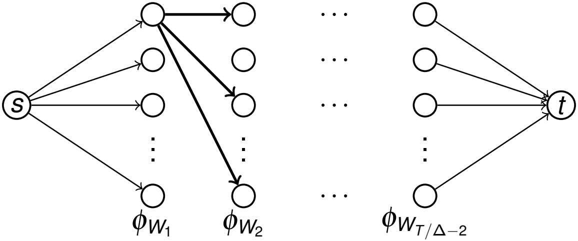
1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.
2. Create a DAG with ϕ_{W_i} as vertices and connect ϕ_{W_i} to $\phi_{W_{i+1}}$ if the two colorings agree on the overlapping part.



Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm II

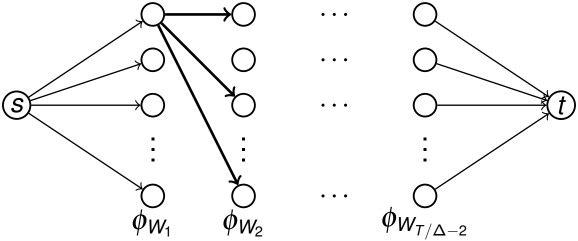
1. For 2Δ-windows $W_i = [i\Delta+1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.
2. Create a DAG with ϕ_{W_i} as vertices and connect ϕ_{W_i} to $\phi_{W_{i+1}}$ if the two colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all ϕ_{W_1} and we create a sink vertex t and connect $\phi_{W_{T/\Delta-2}}$ to it.



Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm II

1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.
2. Create a DAG with ϕ_{W_i} as vertices and connect ϕ_{W_i} to $\phi_{W_{i+1}}$ if the two colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all ϕ_{W_1} and we create a sink vertex t and connect $\phi_{W_{T/\Delta-2}}$ to it.
4. If there is a path from s to t, answer YES, otherwise NO.



Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm II

1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.
2. Create a DAG with ϕ_{W_i} as vertices and connect ϕ_{W_i} to $\phi_{W_{i+1}}$ if the two colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all ϕ_{W_1} and we create a sink vertex t and connect $\phi_{W_{T/\Delta-2}}$ to it.

Theorem

SWTC can be solved in $O(k^{4\Delta \cdot n} \cdot T)$ time.

Sliding Window Temporal Graph Coloring

Main Exponential Time Algorithm II

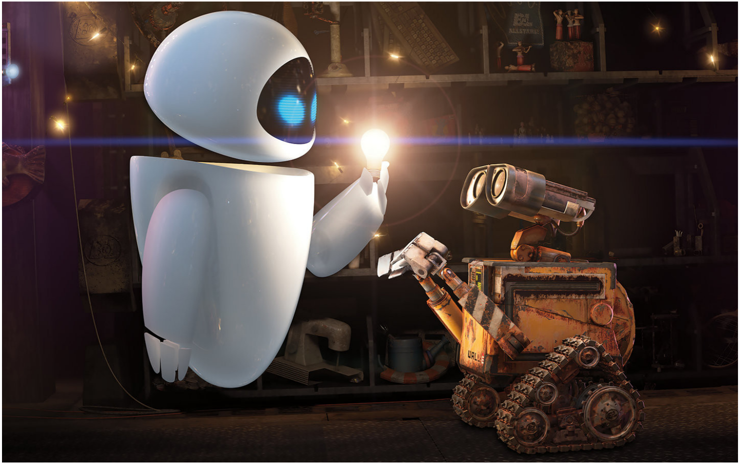
1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed (but arbitrary) way.
2. Create a DAG with ϕ_{W_i} as vertices and connect ϕ_{W_i} to $\phi_{W_{i+1}}$ if the two colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all ϕ_{W_1} and we create a sink vertex t and connect $\phi_{W_{T/\Delta-2}}$ to it.

Theorem

SWTC can be solved in $O(k^{4\Delta \cdot n} \cdot T)$ time.

ETH Lower Bound

SWTC does not admit a $2^{o(n) \cdot f(T+k)}$-time algorithm for any computable function f unless ETH fails.



How to exploit few vertices?

Sliding Window Temporal Graph Coloring

Fixed-Parameter Tractability II

Observation

If a snapshot appears n^2 times in a Δ-window, all its edges can be colored properly. *[just properly color every edge of it once]*

Sliding Window Temporal Graph Coloring

Fixed-Parameter Tractability II

Observation

If a snapshot appears n^2 times in a Δ-window, all its edges can be colored properly. *[just properly color every edge of it once]*

- Idea: Replace additional copies with edgeless snapshots.

Observation

If a snapshot appears n^2 times in a Δ-window, all its edges can be colored properly. *[just properly color every edge of it once]*

- Idea: Replace additional copies with edgeless snapshots.
- Problem: Replacing a snapshot should not reduce the number of its copies in other Δ-windows (which may contain less than n^2 copies).

Observation

If a snapshot appears n^2 times in a Δ-window, all its edges can be colored properly. *[just properly color every edge of it once]*

- Idea: Replace additional copies with edgeless snapshots.
- Problem: Replacing a snapshot should not reduce the number of its copies in other Δ-windows (which may contain less than n^2 copies).

Reduction Rule

If a snapshot appears more than $2n^2$ times in a Δ-window, then replace of its "middle" copies with an edgeless snapshot.

Sliding Window Temporal Graph Coloring

Fixed-Parameter Tractability II

Observation

If a snapshot appears n^2 times in a Δ-window, all its edges can be colored properly. *[just properly color every edge of it once]*

- Idea: Replace additional copies with edgeless snapshots.
- Problem: Replacing a snapshot should not reduce the number of its copies in other Δ-windows (which may contain less than n^2 copies).

Reduction Rule

If a snapshot appears more than $2n^2$ times in a Δ-window, then replace of its "middle" copies with an edgeless snapshot.

Lemma

If the reduction rule is not applicable, each Δ-window contains at most $2n^2 \cdot 2^{n^2}$ (non-trivial) snapshots.

Sliding Window Temporal Graph Coloring

Fixed-Parameter Tractability III

Recall our first exponential-time algorithm:

Theorem

Sliding Window Temporal Coloring can be solved in $O(k^{4\Delta \cdot n} \cdot T)$ time.

Therefore, since every Δ-window has at most $2n^2 \cdot 2^{n^2}$ snapshots (and since $k \leq n$):

Theorem

SWTC is linear-time fixed-parameter tractable (FPT) with respect to n (i.e. in $O(f(n) \cdot T)$ time).

Recall our first exponential-time algorithm:

Theorem

Sliding Window Temporal Coloring can be solved in $O(k^{4\Delta \cdot n} \cdot T)$ time.

Therefore, since every Δ-window has at most $2n^2 \cdot 2^{n^2}$ snapshots (and since $k \leq n$):

Theorem

SWTC is linear-time fixed-parameter tractable (FPT) with respect to n (i.e. in $O(f(n) \cdot T)$ time).

One of our hardness results:

Theorem

SWTC is NP-hard, even if the vertex cover number of the underlying graph G is at most $2k + 13$ (where $k =$ number of colors).

Thus we cannot hope for an (exact) FPT algorithm with respect to the parameter "vertex cover number of the underlying graph".

Sliding Window Temporal Graph Coloring

Fixed-Parameter Tractability III

However:

Theorem

SWTC admits a linear-time FPT-approximation algorithm for parameter "vertex cover number of G" with an additive error one. (Objective: Minimize number of colors.)

However:

Theorem

SWTC admits a linear-time FPT-approximation algorithm for parameter "vertex cover number of G" with an additive error one. (Objective: Minimize number of colors.)

Idea:

- Compute in linear FPT-time a minimum vertex cover of G (the rest is independent set in every slot!)
- Use our exponential algorithm to optimally solve SWTC in the temporal graph induced by the vertex cover vertices
- This is a lower bound on the number of colors needed for (G, λ)
- Color all other vertices in all slots with a fresh color

Reduction from **Monotone Exactly 1-in-3 SAT**.

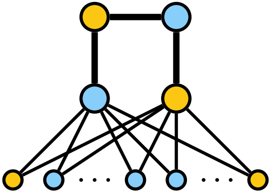
Reduction from **Monotone Exactly 1-in-3 SAT**.

Main Idea: Encode variables with vertices, clauses with snapshots.

Reduction from **Monotone Exactly 1-in-3 SAT**.

Main Idea: Encode variables with vertices, clauses with snapshots.

Variable Gadget:



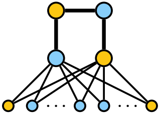
2 colors, sliding window size $\Delta = 2$.

Sliding Window Temporal Graph Coloring

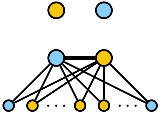
Constant Underlying VC Reduction – Main Ideas II

Reduction from **Monotone Exactly 1-in-3 SAT**.

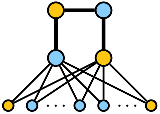
Main Idea: Encode variables with vertices, clauses with snapshots.



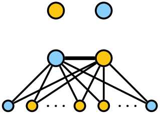
Type 1 snapshot.



Type 2 snapshot.



Type 3 snapshot.



Type 4 snapshot.

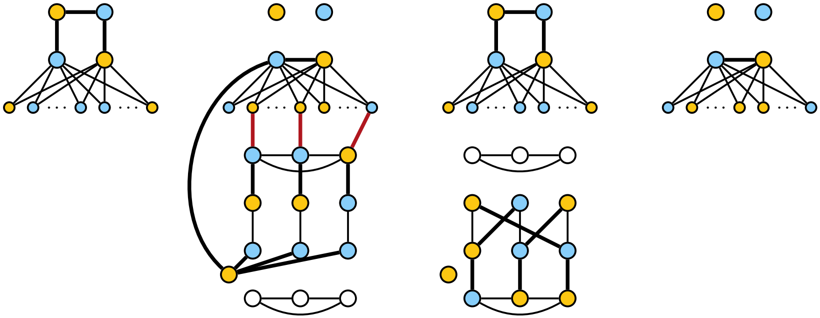
2 colors, sliding window size $\Delta = 2$.

Sliding Window Temporal Graph Coloring

Constant Underlying VC Reduction – Main Ideas II

Reduction from **Monotone Exactly 1-in-3 SAT**.

Main Idea: Encode variables with vertices, clauses with snapshots.



Type 1 snapshot. Type 2 snapshot. Type 3 snapshot. Type 4 snapshot.

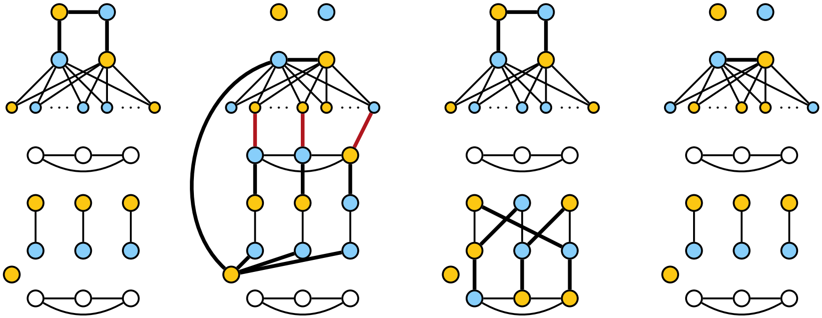
2 colors, sliding window size $\Delta = 2$.

Sliding Window Temporal Graph Coloring

Constant Underlying VC Reduction – Main Ideas II

Reduction from **Monotone Exactly 1-in-3 SAT**.

Main Idea: Encode variables with vertices, clauses with snapshots.



Type 1 snapshot. Type 2 snapshot. Type 3 snapshot. Type 4 snapshot.

2 colors, sliding window size $\Delta = 2$.

Outlook

and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.

Outlook

and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.
- No poly kernel wrt. n.

Outlook

and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.
- No poly kernel wrt. n.

Future Work:

- Restrict input graphs to only change slowly over time.

Outlook
and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.
- No poly kernel wrt. n.

Future Work:

- Restrict input graphs to only change slowly over time.
- Bound number of vertices that can change color each time step.

Outlook
and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.
- No poly kernel wrt. n.

Future Work:

- Restrict input graphs to only change slowly over time.
- Bound number of vertices that can change color each time step.
- Bound number of times a vertex can change color.

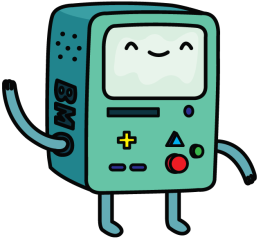
Outlook
and Future Work

Further Results:

- NP-hard for constant k, Δ, and T even if each snapshot is a cluster graph.
- No poly kernel wrt. n.

Future Work:

- Restrict input graphs to only change slowly over time.
- Bound number of vertices that can change color each time step.
- Bound number of times a vertex can change color.



https://arxiv.org/pdf/1811.04753.pdf

Link to arXiv.

Thank you!