Efficient generation of simple temporal graphs
(up to “isomorphism”)

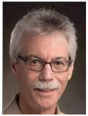
Arnaud Casteigts

LaBRI, Université de Bordeaux

ESTATE-DUCAT Workshop 2022

Related to joint works with:

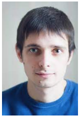
Jason Schoeters (Le Havre)



Joseph Peters (Vancouver)

Michael Raskin (Munich)

Malte Renken (Berlin)



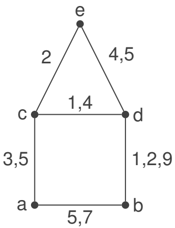
Viktor Zamaraev (Liverpool)

Timothée Corsini (Bordeaux)

Temporal graphs (*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *presence times* to edges (here, discrete)

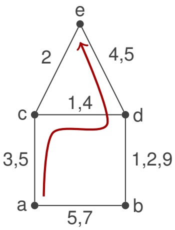
Example:



Temporal graphs (*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *presence times* to edges (here, discrete)

Example:



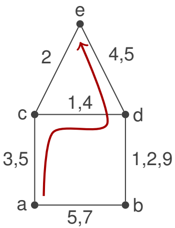
Temporal paths

- Non-strict, ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict, ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (***increasing***)

Temporal graphs (*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \rightarrow 2^{\mathbb{N}}$ assigns *presence times* to edges (here, discrete)

Example:



Temporal paths

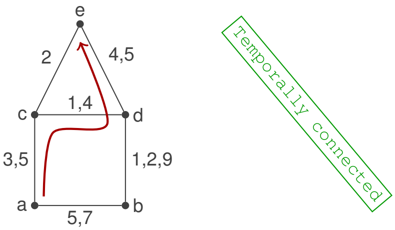
- Non-strict, ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict, ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (***increasing***)

Temporal connectivity: all vertices can reach each other through temporal paths

Temporal graphs (*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *presence times* to edges (here, discrete)

Example:



Temporal paths

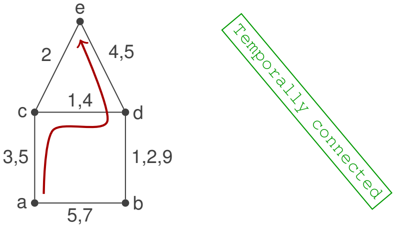
- Non-strict, ex: $\langle (a,c,3), (c,d,4), (d,e,4) \rangle$ (*non-decreasing*)
- Strict, ex: $\langle (a,c,3), (c,d,4), (d,e,5) \rangle$ (***increasing***)

Temporal connectivity: all vertices can reach each other through temporal paths

Temporal graphs (*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *presence times* to edges (here, discrete)

Example:



Temporal paths

- Non-strict, ex: $\langle (a,c,3),(c,d,4),(d,e,4) \rangle$ (*non-decreasing*)
- Strict, ex: $\langle (a,c,3),(c,d,4),(d,e,5) \rangle$ (***increasing***)

Temporal connectivity: all vertices can reach each other through temporal paths

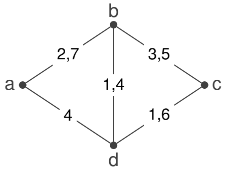
Remark: reachability is non-transitive in general!

Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)

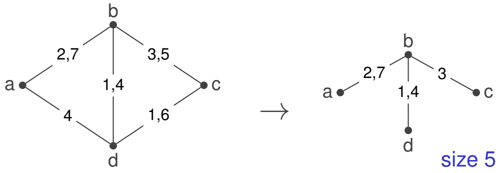


Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)

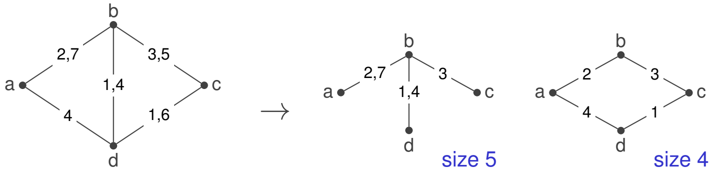


Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)

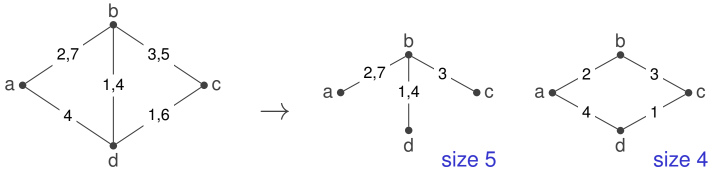


Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)



Can we do better?

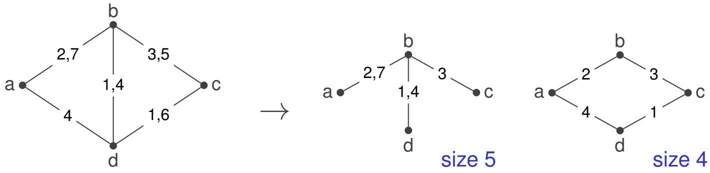
- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

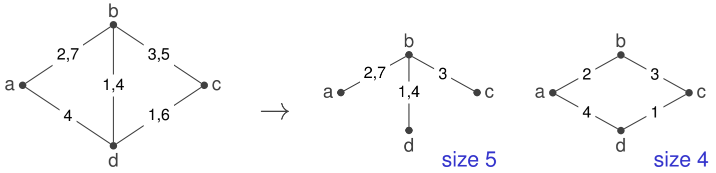
Do spanners of size $2n - 4$ always exist?

Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

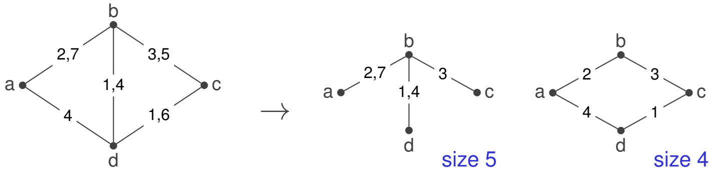
- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)

Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

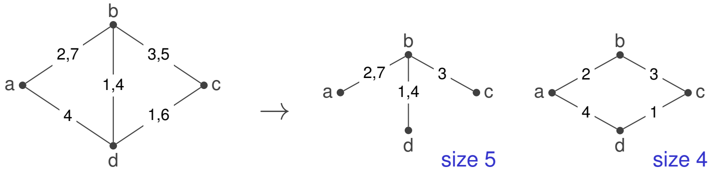
- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)
- In fact, $\exists$ some with $\Omega(n^2)$ labels (Axiotis, Fotakis, 2016)

Temporal spanners (motivation)

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (in number of time labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)
- In fact, $\exists$ some with $\Omega(n^2)$ labels (Axiotis, Fotakis, 2016)

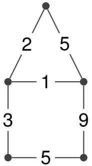
How about complexity?

- Minimum-size spanner is APX-hard (Akrida, Gasieniec, Mertzios, Spirakis, 2017)

An easier model

Simple Temporal Graphs (STGs):

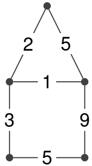
1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



An easier model

<u>Simple</u> Temporal Graphs (STGs):

1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



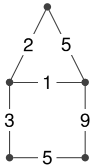
Generality:

- Many negative results apply
- Positive results extend
- No distinction between strict and non-strict temporal paths

An easier model

Simple Temporal Graphs (STGs):

1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



Generality:

- Many negative results apply
- Positive results extend
- No distinction between strict and non-strict temporal paths

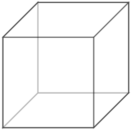
Further motivations:

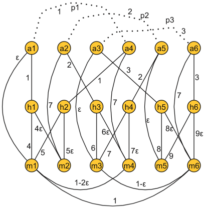
- Population protocols and gossip models (without repetition)
- Edge-ordered graphs (Chvátal, Komlós, 1971)

Back to the bad news... and good news

Recall the bad news:

- $\Omega(n \log n)$
- $\Omega(n^2)$

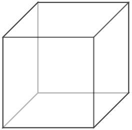


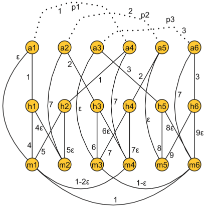


Back to the bad news... and good news

Recall the bad news:

- $\Omega(n \log n)$
- $\Omega(n^2)$





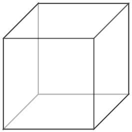
Good news: (C., Raskin, Renken, Zamaraev, 2021):

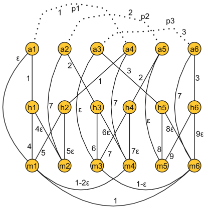
- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, as soon as the graph is temporally connected

Back to the bad news... and good news

Recall the bad news:

- $\Omega(n \log n)$
- $\Omega(n^2)$



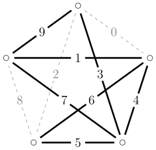


Good news: (C., Raskin, Renken, Zamaraev, 2021):

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, as soon as the graph is temporally connected

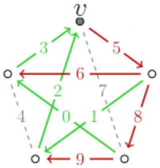
Good news: (C., Peters, Schoeters, 2019):

- Spanners of size $O(n \log n)$ always exist in **complete** temporal graphs



Two techniques for spanners in temporal cliques

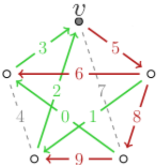
Pivotability



$\rightarrow$ spanners of size $2n - 3$

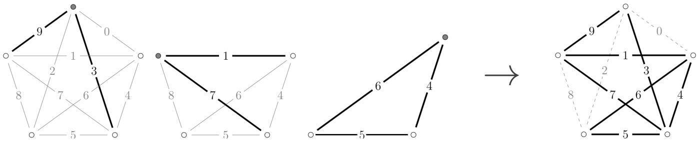
Two techniques for spanners in temporal cliques

Pivotability



→ spanners of size $2n - 3$

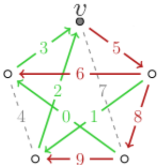
Dismountability



→ spanner of size $2n - 3$.

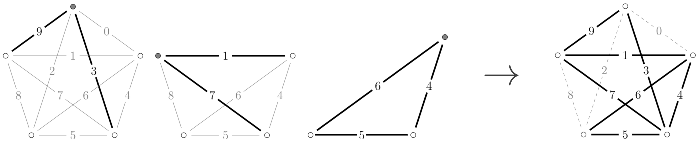
Two techniques for spanners in temporal cliques

Pivotability



$\rightarrow$ spanners of size $2n-3$

Dismountability



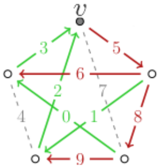
$\rightarrow$ spanner of size $2n-3$.

Unfortunately, only works in most instances

The best we known for general temporal cliques is $O(n \log n)$

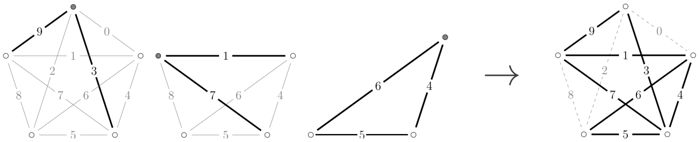
Two techniques for spanners in temporal cliques

Pivotability



$\rightarrow$ spanners of size $2n - 3$

Dismountability



$\rightarrow$ spanner of size $2n - 3$.

Unfortunately, only works in most instances

The best we known for general temporal cliques is $O(n \log n)$

Do spanners of size $2n - 3$ always exist in temporal cliques?

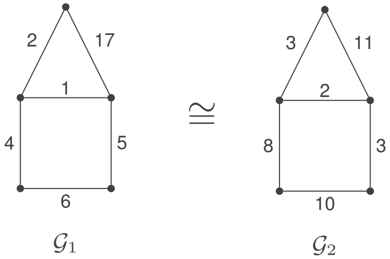
(searching for counter-examples...)

Generation of simple temporal graphs

(all of them, not just cliques)

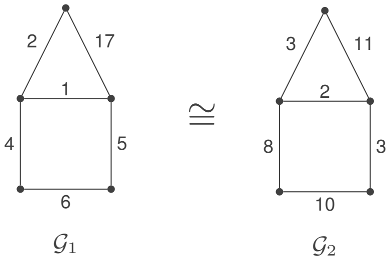
Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)



Equivalence based on reachability (up to time distortion)

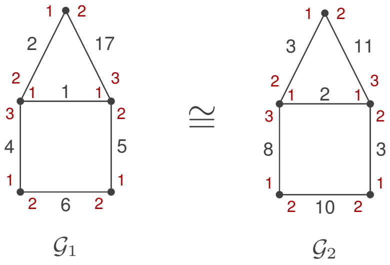
Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)



How to capture this equivalence?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)

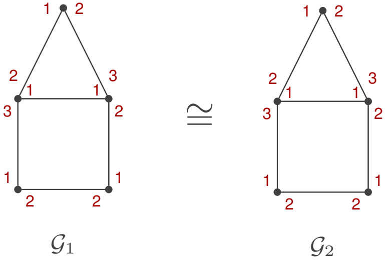


How to capture this equivalence?

- Option 1: Local ordering?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. "Isomorphic")

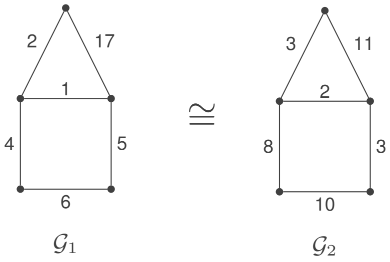


How to capture this equivalence?

- Option 1: Local ordering?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)

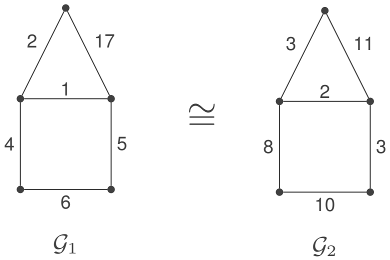


How to capture this equivalence?

- Option 1: Local ordering?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. "Isomorphic")

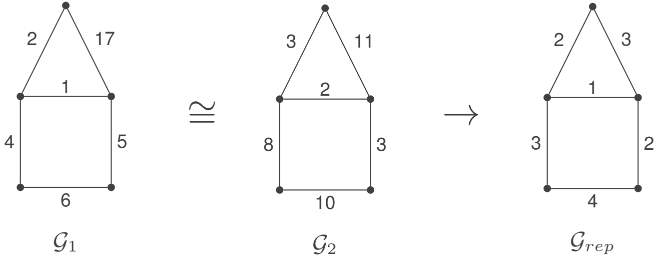


How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)

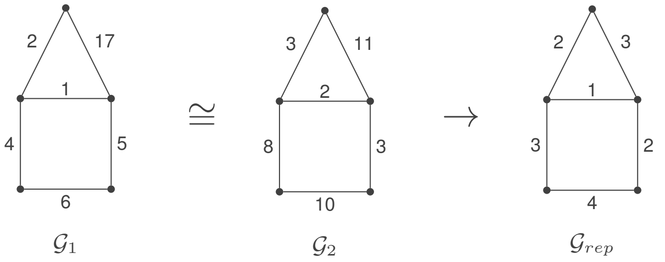


How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability* (i.e. “Isomorphic”)



How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative ✓

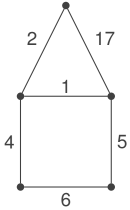
STG representatives have good properties for generation

+ canonization, isomorphism testing, and computation of generators for the automorphism group, are all feasible in *polynomial time*.

STG representatives

Canonization

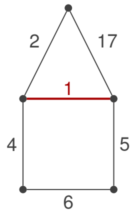
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

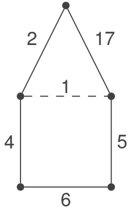
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

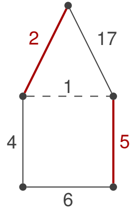
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

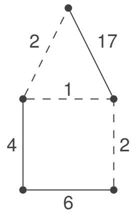
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

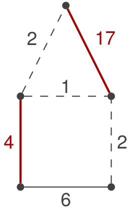
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

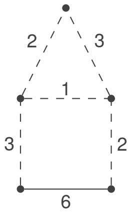
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

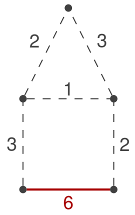
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

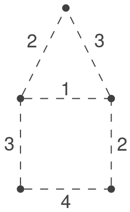
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

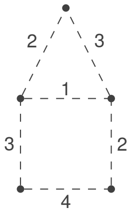
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



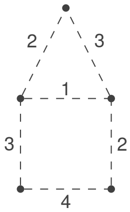
Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

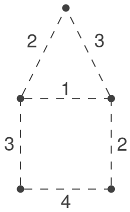
In addition,

Contiguity Lemma: If an edge is labeled $t > 1$, an adjacent edge is labeled $t - 1$.

STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

In addition,

Contiguity Lemma: If an edge is labeled $t > 1$, an adjacent edge is labeled $t - 1$.

(If you know a name for such coloring, let me know.)

How to test for equivalence?

How to test for equivalence?

Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

How to test for equivalence?

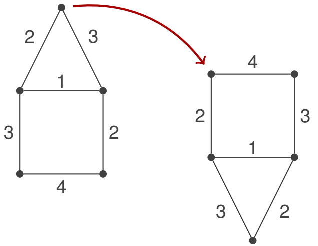
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

How to test for equivalence?

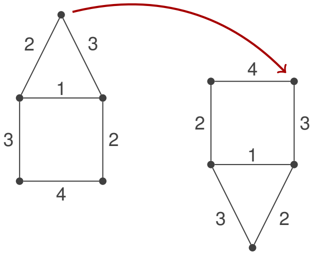
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

How to test for equivalence?

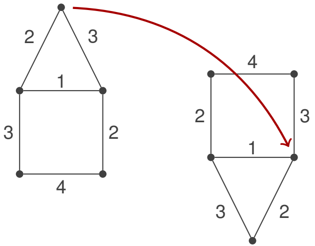
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

How to test for equivalence?

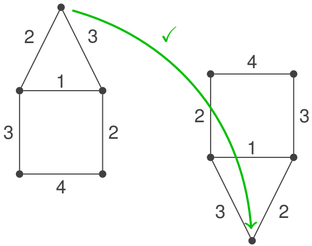
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer YES
4. If not, try the next vertex of G_2 (or answer NO if none remain)

How to test for equivalence?

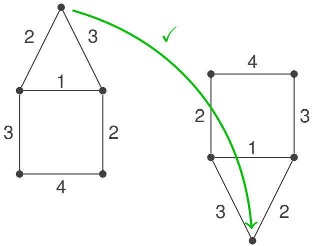
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer YES
4. If not, try the next vertex of G_2 (or answer NO if none remain)

Key observation: when trying to send v_1 to v_2, the mapping among neighbors unfolds recursively without choices (due to the *proper coloring* of the edges)

$\rightarrow$ passes or fails in polynomial time.

How to test for equivalence?

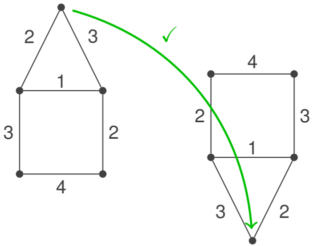
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2
   (or answer `NO` if none remain)

Key observation: when trying to send v_1 to v_2, the mapping among neighbors unfolds recursively without choices (due to the *proper coloring* of the edges)

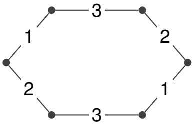
$\rightarrow$ passes or fails in polynomial time.

Remark: Also feasible using Babai & Luks machinery (1983)

Automorphisms of an STG

Case 1: The underlying graph is connected.

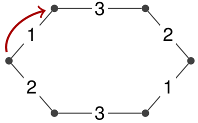
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

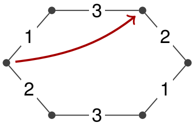
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

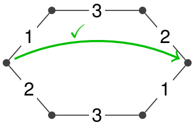
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

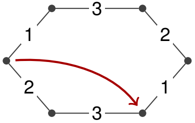
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

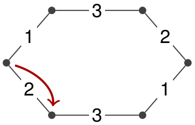
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

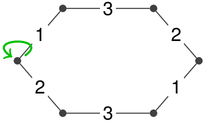
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.

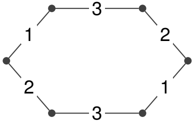


At most n automorphisms!

Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



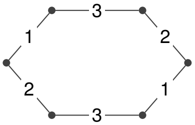
Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

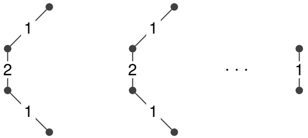
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

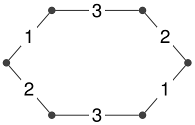


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

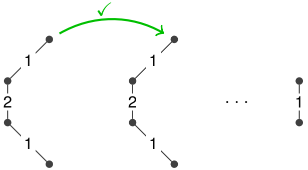
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

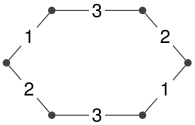


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

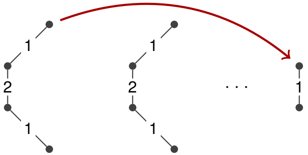
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

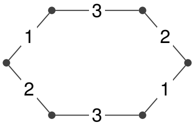


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

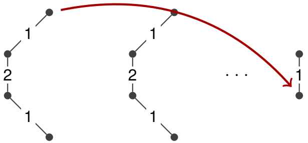
Case 1: The underlying graph is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

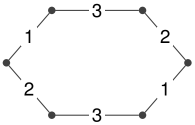


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

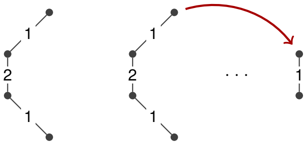
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

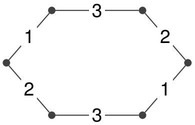


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

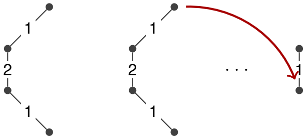
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

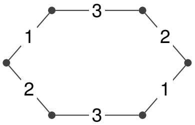


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

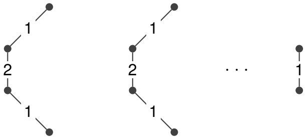
Automorphisms of an STG

Case 1: The underlying graph is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

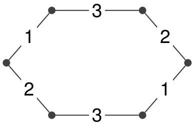


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

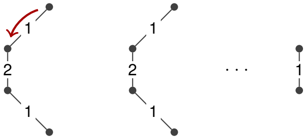
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

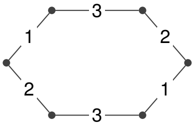


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

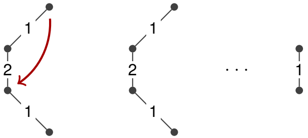
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

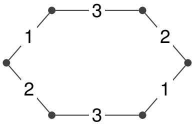


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

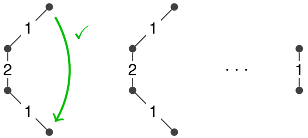
Automorphisms of an STG

Case 1: The underlying graph is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

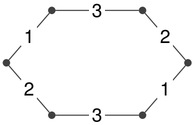


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

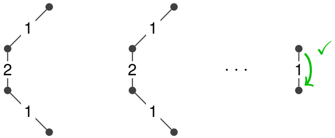
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

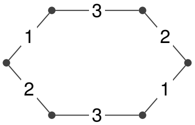


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

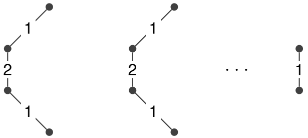
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)



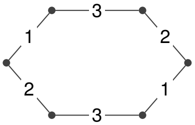
1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Claim: $Aut(\mathcal{G}) = \langle$ isomorphisms + automorphisms $\rangle$

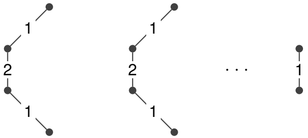
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)



1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Claim: $Aut(\mathcal{G}) = \langle$ isomorphisms + automorphisms $\rangle$

→ *Generators for $Aut(\mathcal{G})$ can be computed in polynomial time!*

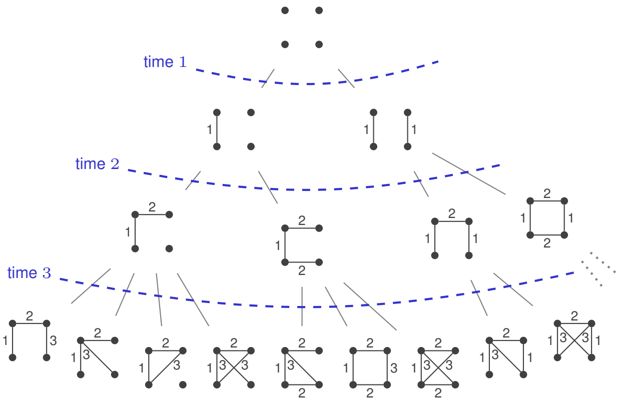
Enumeration up to "isomorphism"

(motivated by the conjecture on spanners)

Generation tree

Principle: One level = one time unit

$\rightarrow$ children of a graph $=$ all the possible ways to add the *next time*



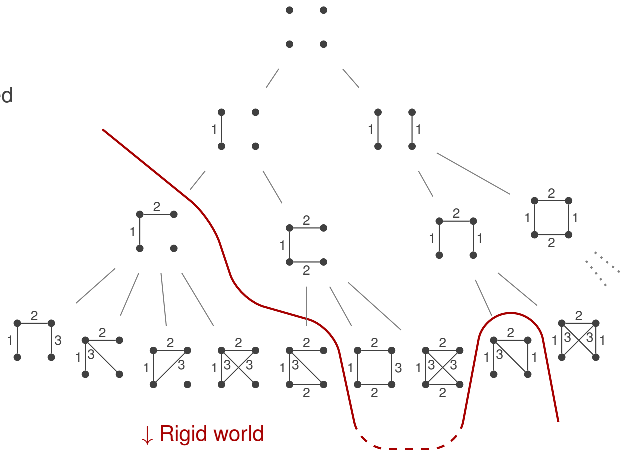
Generation tree

Principle: One level = one time unit

$\rightarrow$ children of a graph $=$ all the possible ways to add the *next time*

Key properties

1. Rigidity is inherited



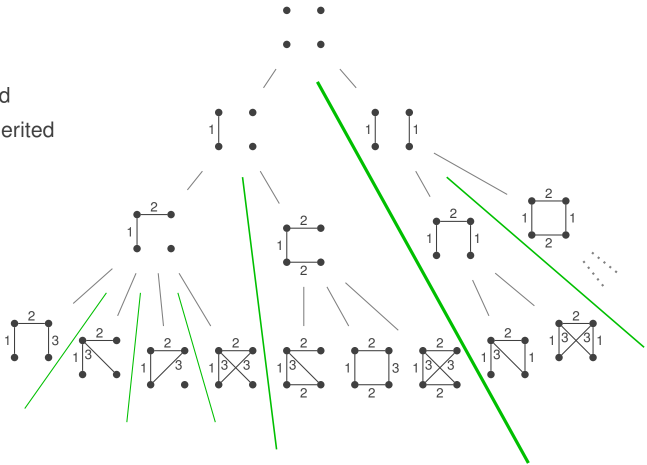
Generation tree

Principle: One level = one time unit

$\rightarrow$ children of a graph = all the possible ways to add the *next time*

Key properties

1. Rigidity is inherited
2. Dissimilarity is inherited



$\downarrow$ Isomorphism types separated (forever)

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

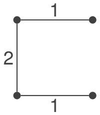
Output: All STG representatives that extend $\mathcal{G}$ with time $t + 1$.

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t + 1$.

First, how to decide if a *non-edge* is eligible to receive a $(t + 1)$ time label? (E.g. here, time 3)

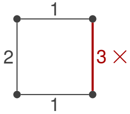


Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t + 1$.

First, how to decide if a *non-edge* is eligible to receive a $(t + 1)$ time label? (E.g. here, time 3)



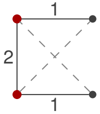
Coloring lemma: (t+1) must be adjacent to (t)

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



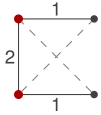
Coloring lemma: (t+1) must be adjacent to (t)

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

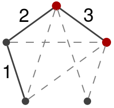
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



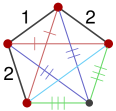
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries

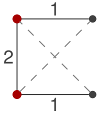


Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

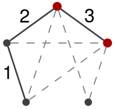
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



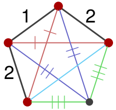
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries



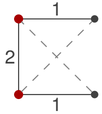
$\rightarrow$ Enumerate all matchings of eligible *non-edges*. Each one defines a successor.

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

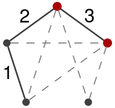
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



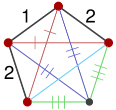
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries



$\rightarrow$ Enumerate all matchings of eligible *non-edges*. Each one defines a successor.

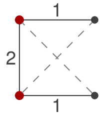
$\equiv$ Independent sets in the *line graph* of eligible *non-edges* (standard algorithm)

Generating successors in the tree?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

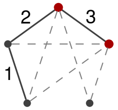
First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



Coloring lemma: (t+1) must be adjacent to (t)

Two cases

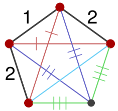
$\mathcal{G}$ is rigid



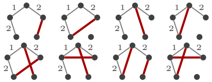
→ Enumerate all matchings of eligible *non-edges*. Each one defines a successor.

≡ Independent sets in the *line graph* of eligible *non-edges* (standard algorithm)

$\mathcal{G}$ has symmetries



→ Enumerate matchings of eligible *non-edges* whose *multisets of orbits* are distinct



Done using the generators for $Aut(\mathcal{G})$

Using the generator

https://github.com/acasteigts/STGen

Implemented in Julia

(other versions in Python, Java, and Rust)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Using the generator

https://github.com/acasteigts/STGen

Implemented in Julia

(other versions in Python, Java, and Rust)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Using the generator

https://github.com/acasteigts/STGen

Implemented in Julia
(other versions in Python, Java, and Rust)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Back to the spanner question

Do simple temporal cliques admit spanners of size $2n-3$?

Using the generator

https://github.com/acasteigts/STGen

Implemented in `Julia`
(other versions in `Python`, `Java`, and `Rust`)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Back to the spanner question

Do simple temporal cliques admit spanners of size $2n-3$?

$\rightarrow$ True for $n \leq 7$ (and for all non-rigid graphs at $n = 8$). **Otherwise still open! :-)**

Some numbers

# Vertices	# STGs	# Temporally connected STGs	# Simple Temporal cliques
1	1	1	1
2	2	1	1
3	4	1	1
4	62	32	20
5	15378	10207	4524
6	89769096	70557834	23218501
7	13828417028594	?	3129434545680
8	?	?	?

Some numbers

# Vertices	# STGs	# Temporally connected STGs	# Simple Temporal cliques
1	1	1	1
2	2	1	1
3	4	1	1
4	62	32	20
5	15378	10207	4524
6	89769096	70557834	23218501
7	13828417028594	?	3129434545680
8	?	?	?

Thanks!