Temporal Locality and Temporal Parameters

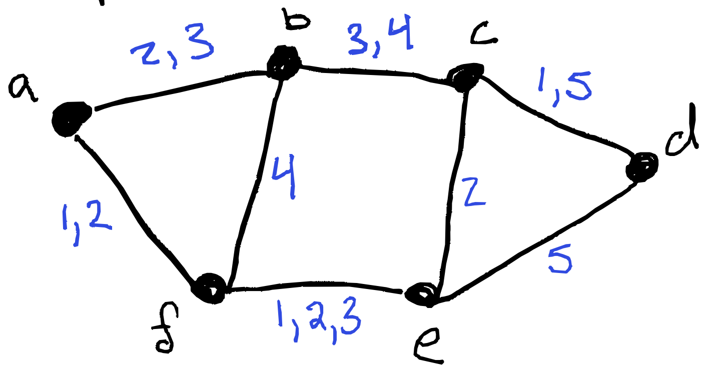
Jess, Sam Hand, Laura Larios-Jones, Kitty Meeks

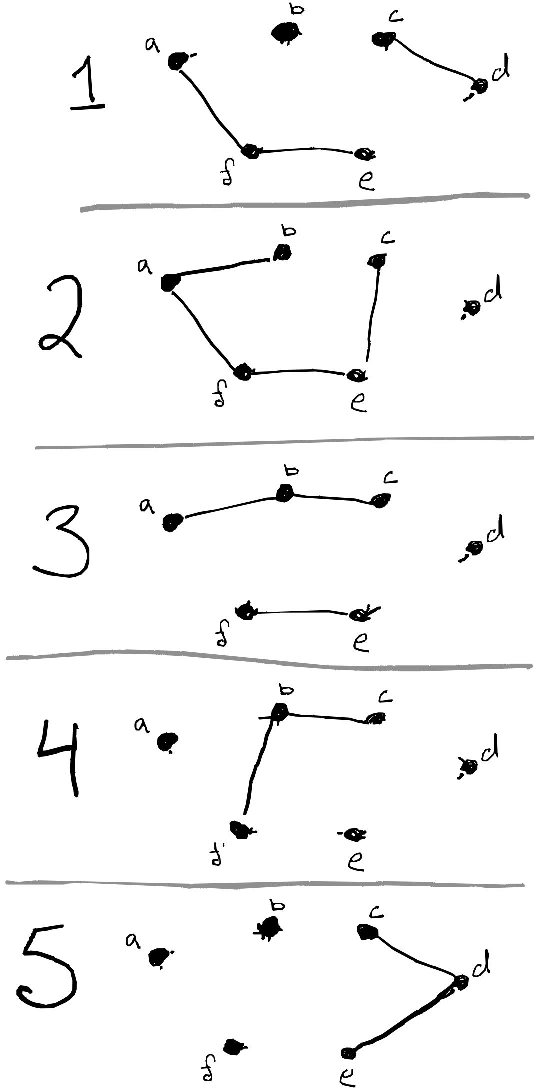
The Plan:

- Temporal graphs + parameterised algorithms
- A motivating example
- "Convenient" locality notions for problems
- Solving these + VIM + TIM

Does time really need to be a line?

Temporal Graphs





For us:

Graphs with edges active at specified timesteps.

Important ideas:

- underlying graph ($G_\downarrow$)
- lifetime (λ)
- active interval.

Why temporal graphs?

- Fun
- Profit

Problems on Temporal Graphs

- Many problems can be made temporal. (but often in multiple sensible ways!)

Example:
Does there exist a <u>temporal clique</u> of size k?

⇓

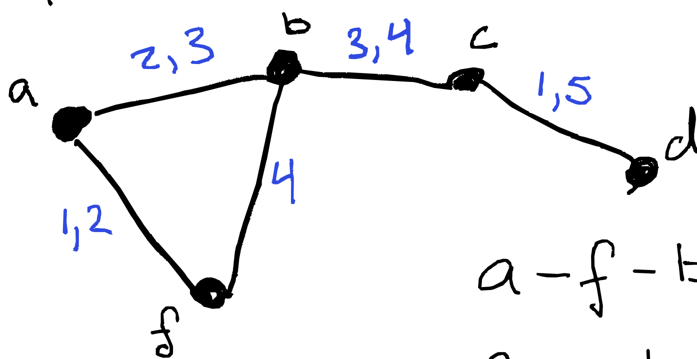
??

- What counts for adjacency:
  - ever adjacent?
  - all adjacent at same time step?
  - all adjacent all the time?
  - adjacent in some time window, or at some frequency?

Depending on def. same set of vertices could be both a clique <u>and</u> independent!!

Temporal Graph Problems

- Temporal Hamiltonian Path



a − f − b − c − d NO

f − a − b − c − d YES

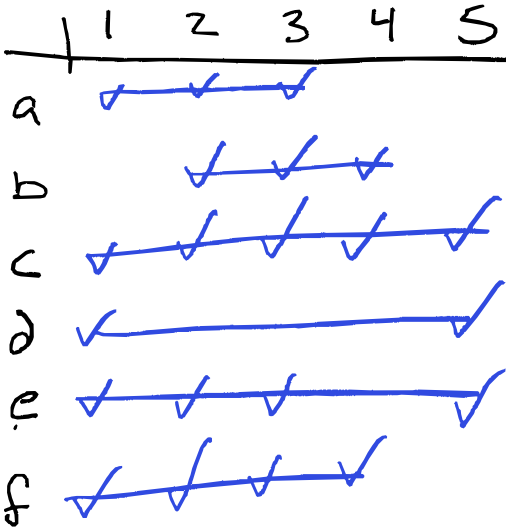
- Modification for reachability:
  remove k edge-appearances to limit
  max reachable set size

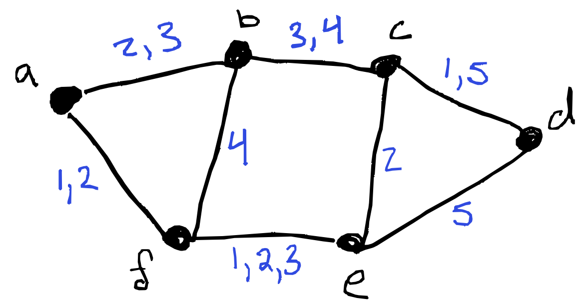
Temporal Graph Parameters

(you may have heard some of these)

~~Vertex-interval-membership-width~~

VIM-width

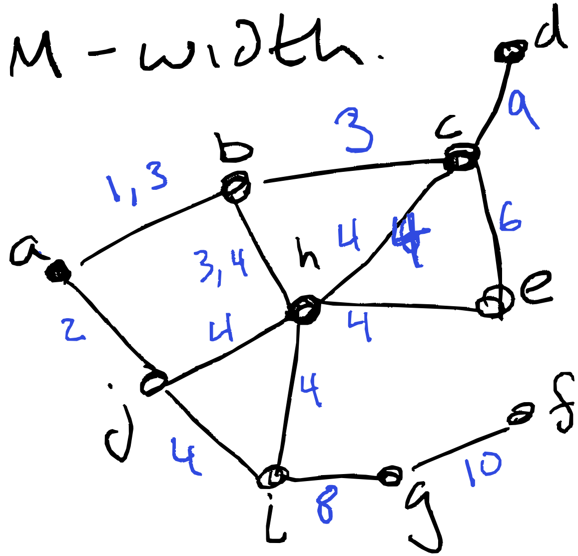


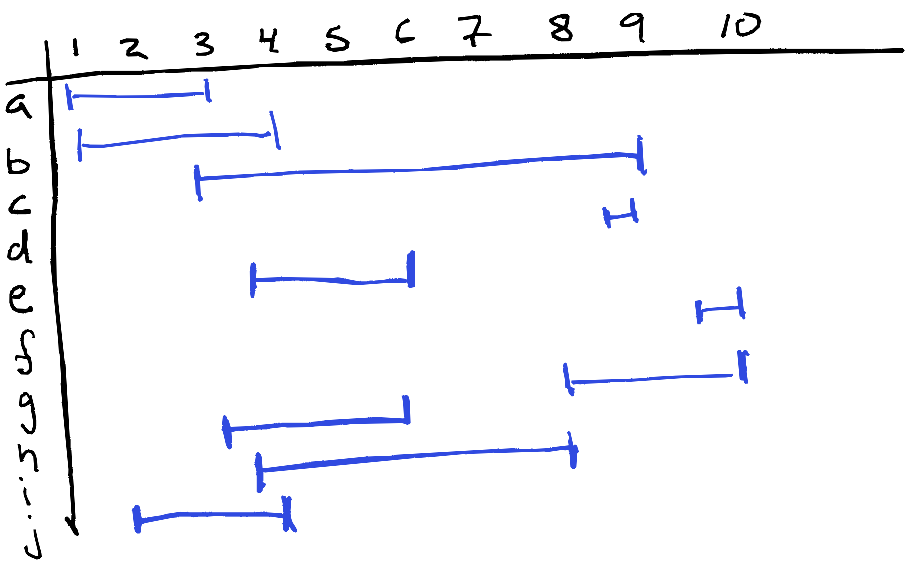


VIM-w = max number in active interval (6 here ;))

active intervals

VIM-width.





Why do we care about this?

Because for some hard problems we can devise algorithms that are

$$O(n^c f(w))$$

width

How do VIM Dynamic Programs Work?

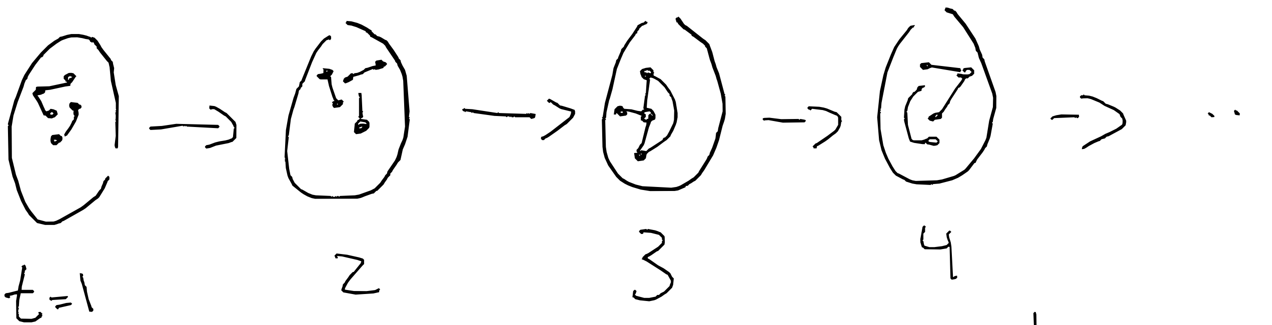
1. Make a series of bags, one for each time.

2. Start at the beginning, consider all possibilities at each bag.
   → and be able to tell if a possibility is supported by the time before it.

3. Get to the end.

An example: Hamitonian Path.

Hamiltonian Path DP by VIM



at a bag, a state records what verts are visited, "where" path currently is

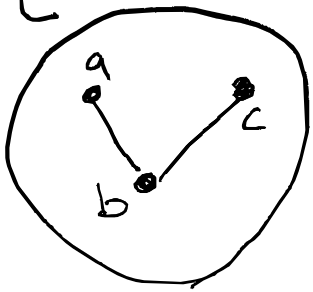
→ Key ingredients:

- We can generate all states
- We can check if a state is "supported" at prev. time step
- our state captures everything we need

Insight - a vertex can only change label when active

Labels: {visited, current, unvisited}

Time t



States:

$a \to c$
$b \to V$
$c \to V$

$a \to c$
$b \to V$
$c \to u$

$a \to c$
$b \to u$
$c \to u$

$a \to c$
$b \to u$
$c \to V$

$a \to V$
$b \to c$
$c \to V$

. . . $\underbrace{O(|\text{Labels}|^{\omega})}_{\text{of these}}$

size of bag

Support:

e.g

$a \to u$
$b \to c$
$c \to V$

$t-1$

supports

$a \to c$
$b \to V$
$c \to V$

t

Much as I love these DPs, they often look similar.

This inspires a meta-algorithm on a well-behaved problem type

What ingredients did we need?

- states that capture enough about problem (and few of them)
  - → related: we can ignore vertices outside active interval
- ability to check bag-to-bag efficiently
- ability to check states

Let's get formal(ish)

We need three definitions:

- (k, X)-State
- (k, X)-Temporally Uniform Problem
- (k, X)-Locally Temporally Uniform Problem.

Punchline:

If a problem is (k, X)-Locally Temporally Uniform,
then we can solve it in:

$$O\big(\overset{\text{lifetime}}{\lambda}\ \underset{\text{runtime of checking algs}}{p(n)}\ \underset{\text{size of biggest thing in } k}{b}^{2k}\ |X|^{2\overset{\text{VIM-width}}{w}}\big)$$

natural

set of labels

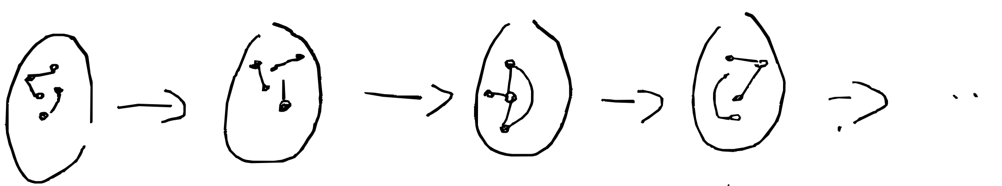
Def: (K, X)-state on vertices V of G

is a pair (ℓ, c)

labelling of vertices in V with labels in X

vector of K integers. (magnitude poly in $|G|$)

e.g. the visited, unvisited, current labels for Hamiltonian



$t=1$ 2 3 4

Def: (K,X)-Temporally Uniform Problem if:

- $\exists$ transition algorithm

  state $\xrightarrow[\text{graph}]{}$ state?

- $\exists$ accepting algorithm

  state?

- $\exists$ a set of starting states for an instance

s.t. instane x is a yes-instance iff

$\exists$ a sequence of states that:

- starts in the set of starts
- transitions one-to-the-next
- all states are accepted.

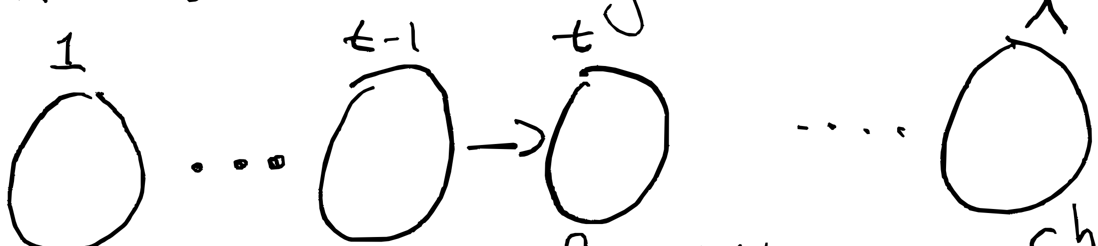
Def: (K, X)-Locally-Temporally Uniform Problem

If previous definition plus:

- vertices can only change state when incident at an edge.

(via several quite-technical requirements)

What does the algorithm look like?

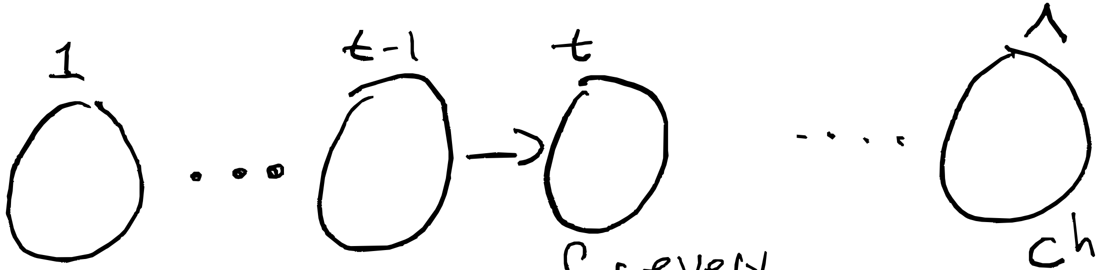


start with start states

given all states

for every state add if it's a transition from a t-1 state

check if end state is acceptable.



start with start states

given all states

for every state add if it's a transition from a t-1 state

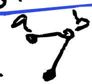
check if end state is acceptable

$$O\Big(n\, p(n)\, \underbrace{b^{2k}\, |X|^{2w}}_{\text{"for every state .. given all } t-1 \text{ states"}}\Big)$$

Example: Temporal Hamiltonian Path.

states: (mapping of vertices to {vis, unvis, curr} , number visited vertices)

starting states:

$G=$ 

$t=0$

$\begin{pmatrix} a \to c \\ b \to u \\ c \to u \end{pmatrix}, 1$) ; ($\begin{pmatrix} a \to u \\ b \to c \\ c \to u \end{pmatrix}, 1$)

acceptence:

at Λ, number visited = number vertices

transition:

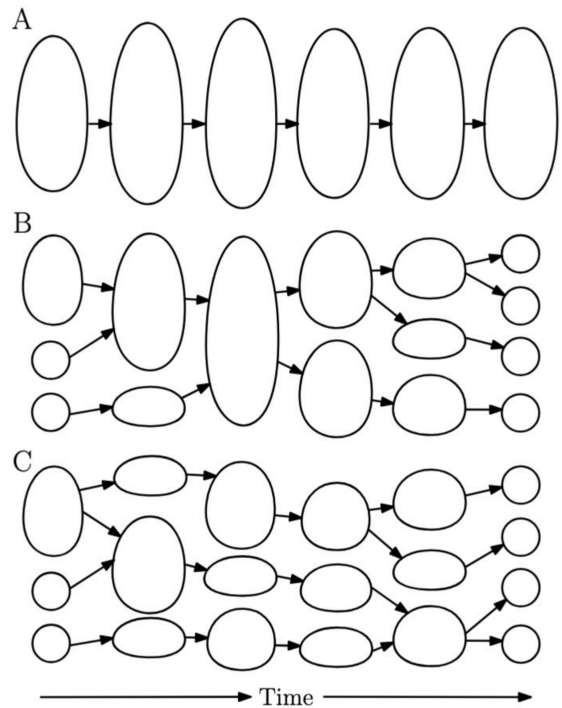
If states are the same or

current changes along an edge,
old current moves to visited,
number visited vertices incremented.

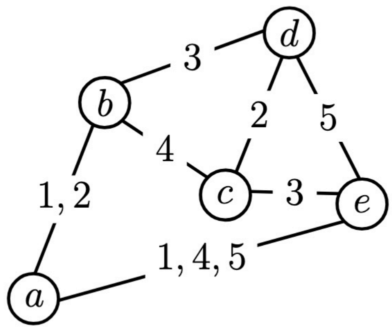
But there's more!

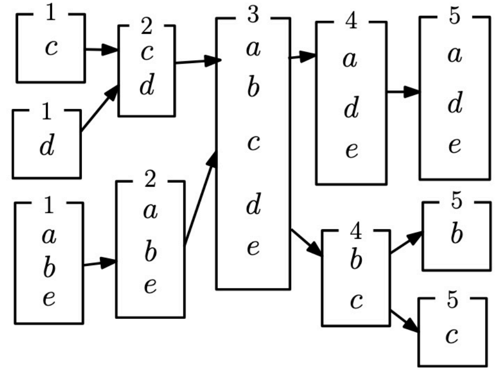
tree-interval-membership width

TIM width

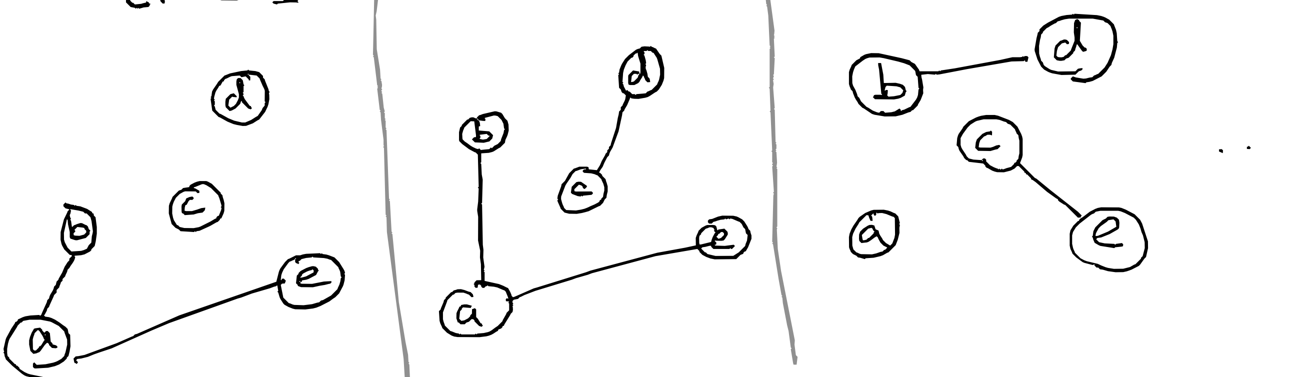


TIM Decomposition example:





time 1 time 2 time 3



Why TIM?

- we can define analogous Locality problem requirements
- and from there a meta-algorithm

More complex, but

$$VIM \geq TIM$$

Summary:

We picked out a set of characteristics that make a problem DP-able by VIM decomp

+ similarly for the new TIM decomp

Thanks