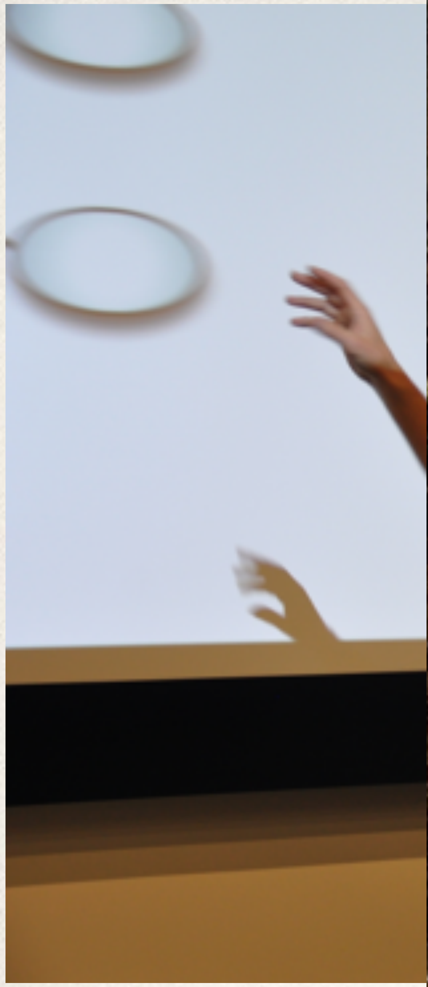
Tree-cut Width: Computation and Algorithmic Applications

Eun Jung Kim, CNRS - Paris Dauphine University

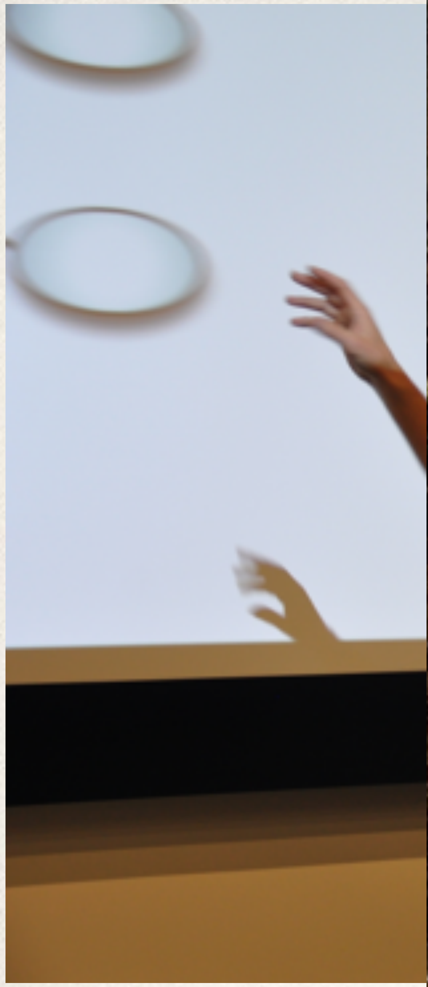
AGTAC, Koper, Slovenia
17 June 2015

Tree-cut width proposed by Paul Wollan, 2013



Tree-cut wi

Algorithmic application of tree-cut width
joint-work with Robert Ganian and Stefan Szeider.



Constructing a tree-cut decomposition
joint-work with Sang-il Oum, Christophe Paul, Ignasi Sau and Dimitrios Thilikos.

Tree-cut wi

Algorithm
joint-work wit

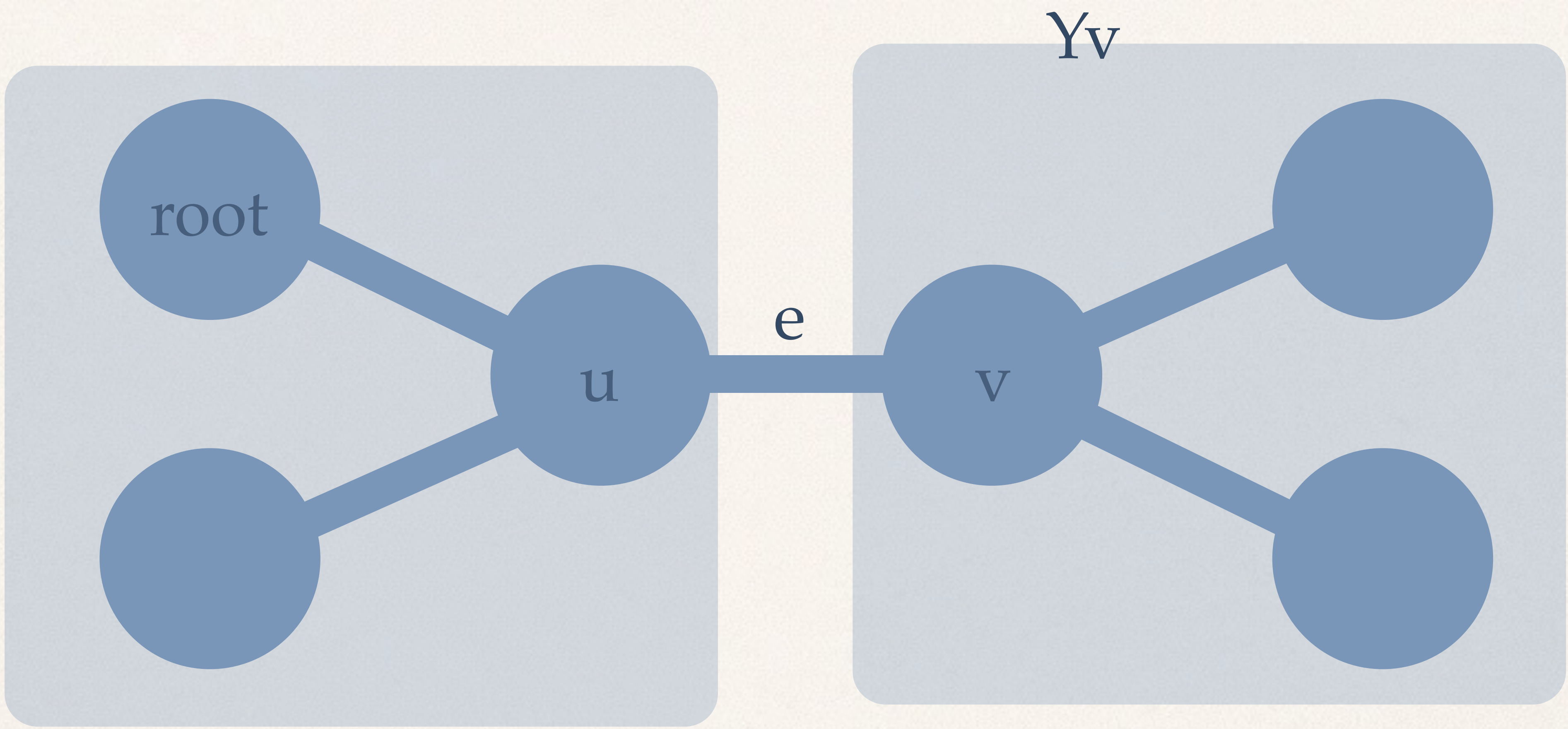
Tree-cut decomposition

[Marx&Wollan 2014, Wollan 2015]

(T, χ={Xt, t $\in$ V(T)}) is a tree-cut decomposition of G if

- T is a tree
- χ forms a <u>near-</u>partition of V(G)

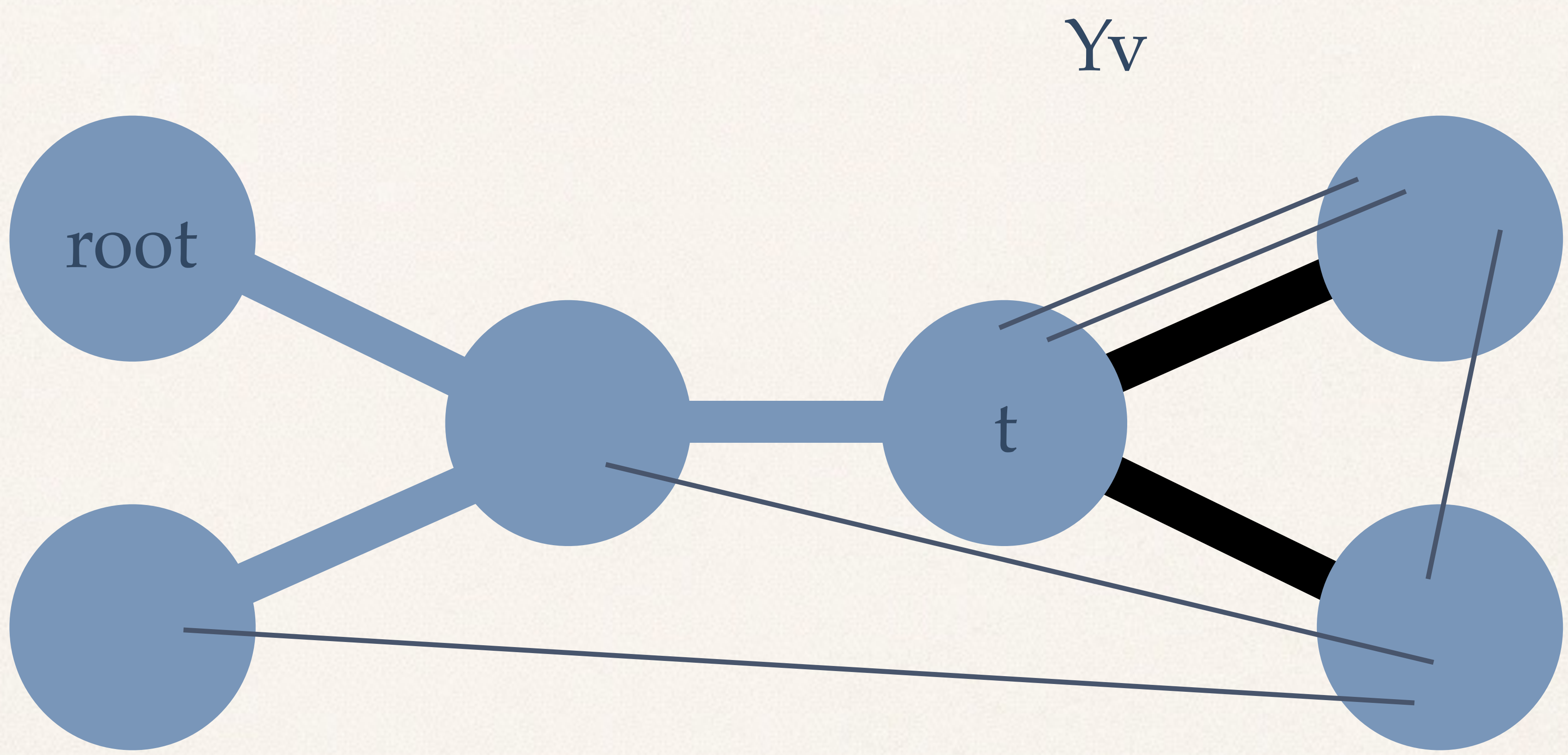
Tree-cut width: (1) cut



cut(e) = the set of edges with one point in Yv and another in V(G)-Yv

Tree-cut width: (2) torso

3-edge-connected case

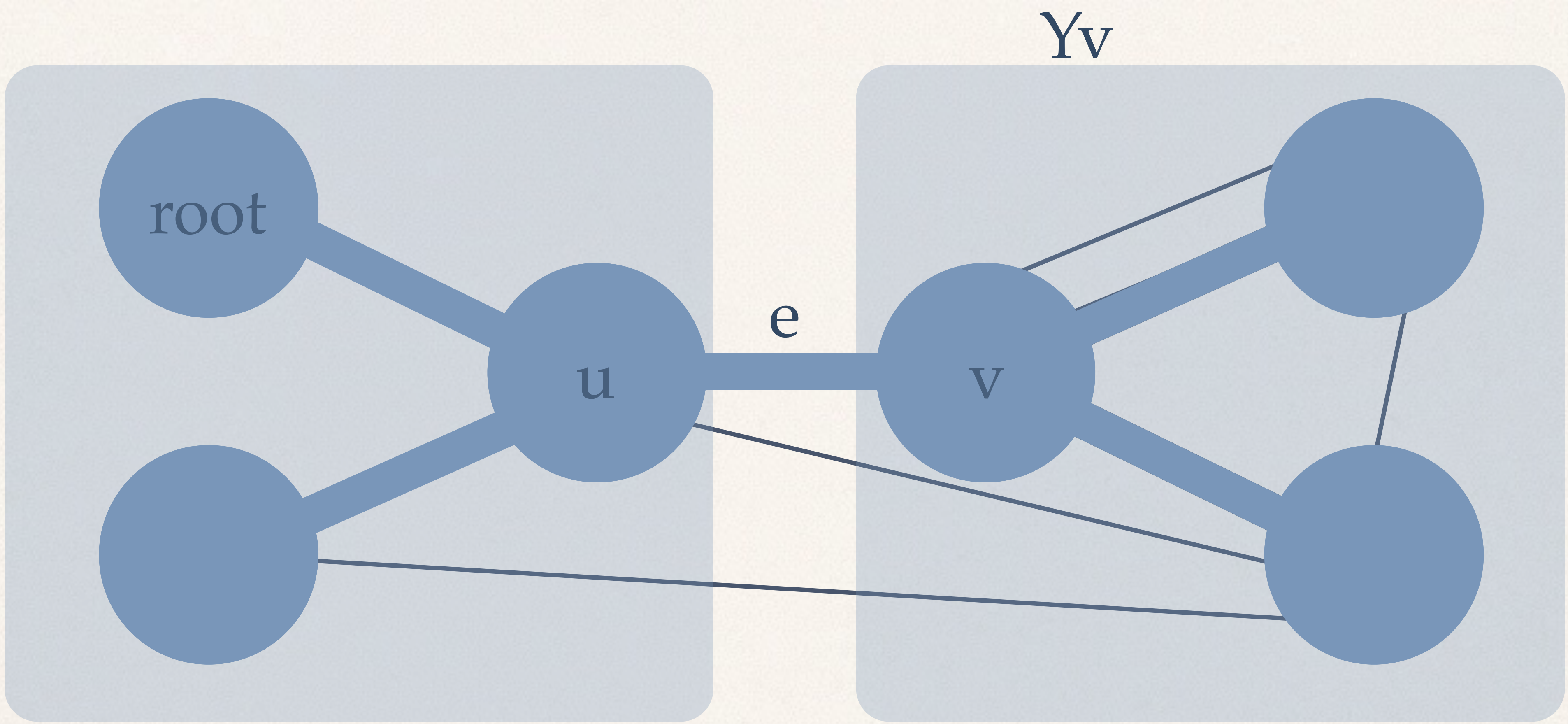


Rt = all neighboring tree nodes of t

|torso(t)| = |Xt| + |Rt|

Tree-cut width: (3) width

3-edge-connected case



cut(e) = the set of edges with one point in Yv and another in V(G)-Yv

Rt = all neighboring tree nodes of t

|torso(t)| = |Xt|+|Rt|

Tree-cut width: (3) width

3-edge-connected case

Yv

$$\text{width}(T,\chi) = \max \{ |\text{cut}(e)|,\ |\text{torso}(t)| \}$$

$$\text{tcw}(G) = \min \text{width}(T,\chi)$$

cut(e) = the set of edges with one point in Yv and another in V(G)-Yv

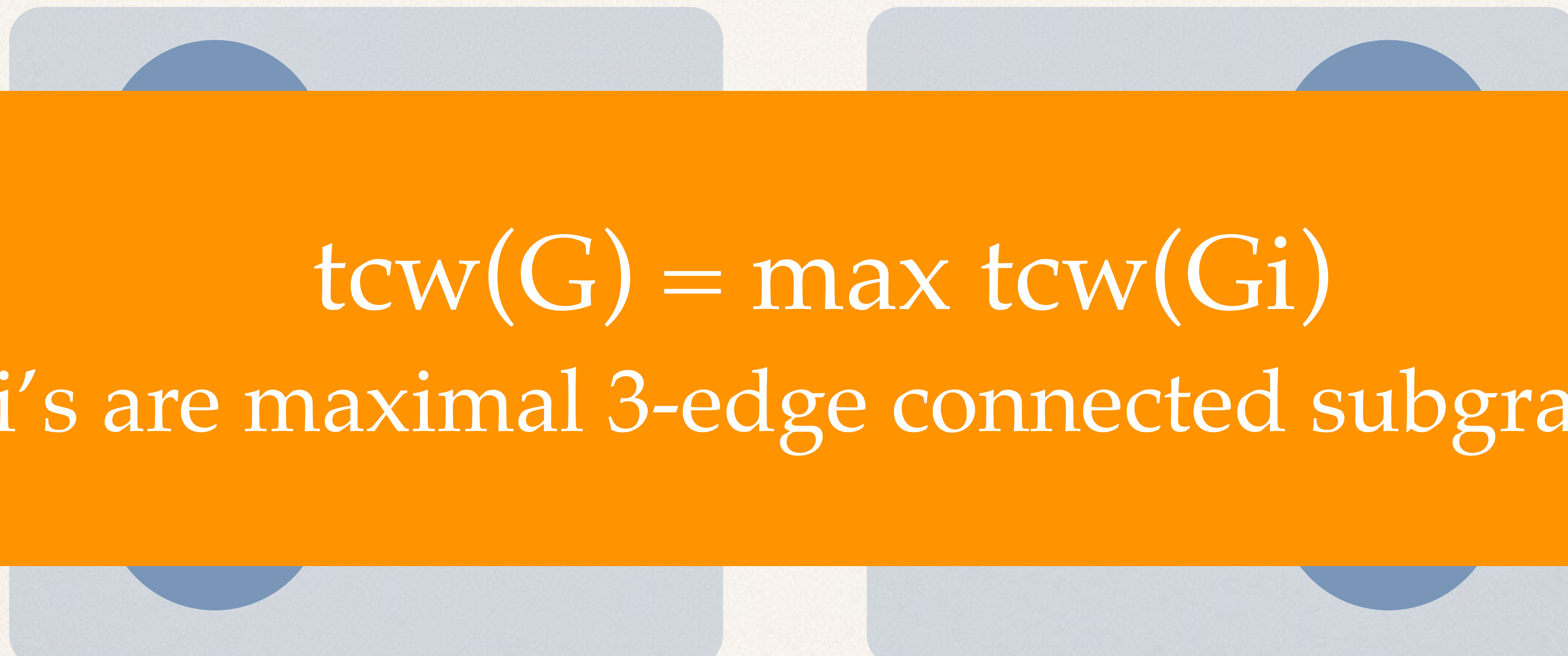
Rt = all neighboring tree nodes of t

torso(t) = |Xt| + |Rt|

Tree-cut width: (3) width

general case

Yv



$$tcw(G) = \max tcw(G_i)$$

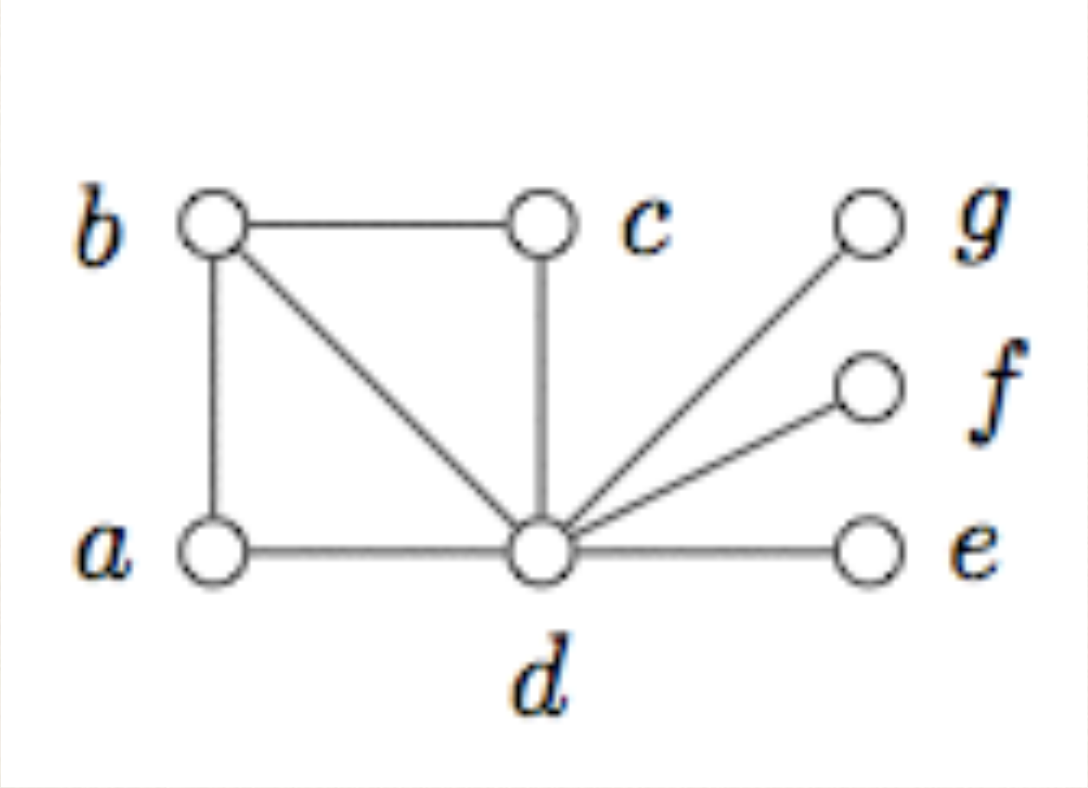
G_i's are maximal 3-edge connected subgraphs

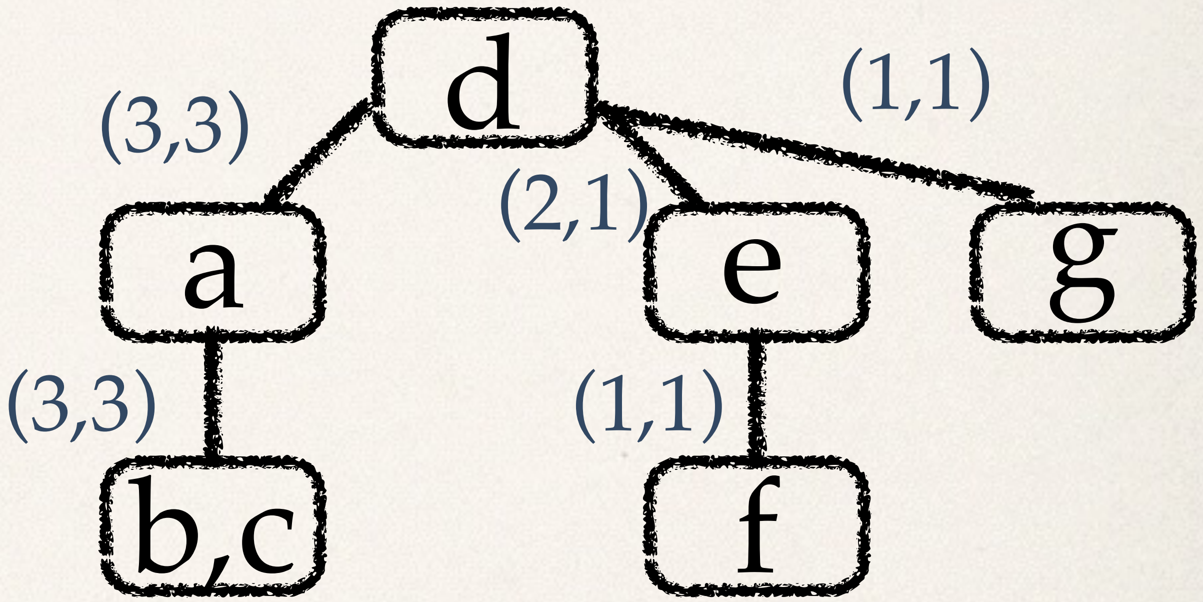
cut(e) = the set of edges with one point in Y_v and another in $V(G)-Y_v$

R_t = all neighboring tree nodes of t

$torso(t) = |X_t| + |R_t|$

Tree-cut width: (4) example

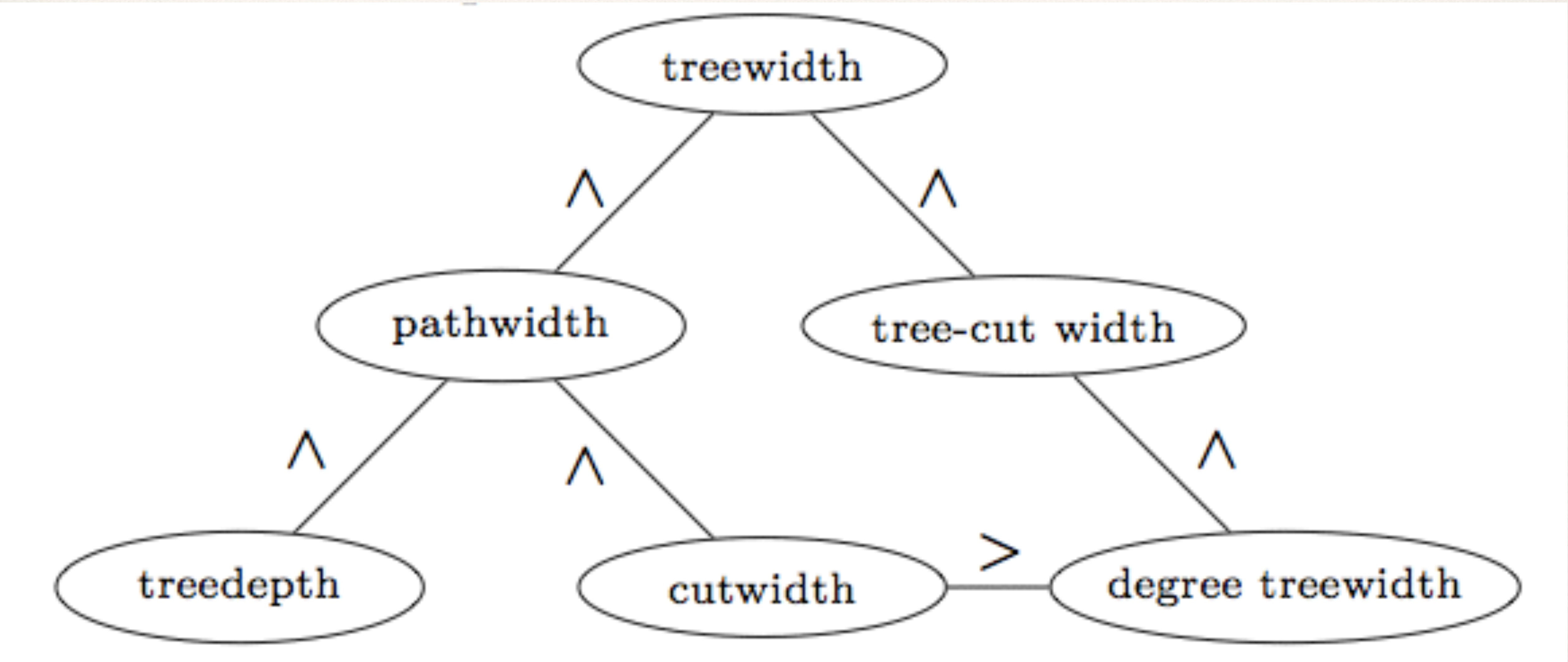




cut(t) = cut(e) where e=(t,p(t))

width = 3

Relations with other width measures



Tree-cut width for algorithms?

- Tree decomposition turned out to be a successful tool for algorithms design
- How about tree-cut decomposition?
  - tw = O(tcw^2): having small tcw is stronger than small tw
  - Intractable problems on graph with small tw may have hope on graph with small tcw

Algorithmic applications
with Robert Ganian and Stefan Szeider

	Parameter		
Problem	*treewidth*	*tree-cut width*	*max-degree and treewidth*
Capacitated Vertex Cover	W[1]-hard[7]	FPT(Thm 9)	FPT
Capacitated Dominating Set	W[1]-hard[7]	FPT(Thm 23)	FPT
Imbalance	Open[28]	FPT(Thm 16)	FPT[28]
List Coloring	W[1]-hard[11]	W[1]-hard(Thm 24)	FPT(Obs 4)
Precoloring Extension	W[1]-hard[11]	W[1]-hard(Thm 24)	FPT(Obs 4)
Boolean CSP	W[1]-hard[35]	W[1]-hard(Thm 25)	FPT[35]

FPT w.r.t. parameter k means there is a f(k)poly(n)-algorithm.
W[1]-hard means f(k)poly(n)-algorithm is unlikely.

Computing a tree-cut decomposition

- QUEST: design an algorithm which answers the question exactly
  - Given a graph G: produce a tree-cut decomposition of width at most k or declare that tcw > k.
- …and which runs as quickly as possible

- Deciding if tcw $\leq$ k is NP-complete: from min bisection
- Exact computation: non-uniform, non-constructive
  - Graphs of tcw $\leq$ k are closed under immersion [Wollan 2015]
  - Graphs are w.q.o. under immersion [N.Robertson, P.D.Seymour 2010]
  - W.Q.O. of immersion implies a finite characterization by forbidden immersions. [N.Robertson, P.D.Seymour 2010]
  - Immersion testing can be done in f(k)poly(n)
    [M. Grohe, K.-i. Kawarabayashi, D. Marx, and P. Wollan 2011]
- Approximation
  - 2-approximation in time 2^O(k^2 · logk) · n^2
    [by E.J.Kim, S.Oum, C.Paul, D.Thilikos, I.Sau 2015]

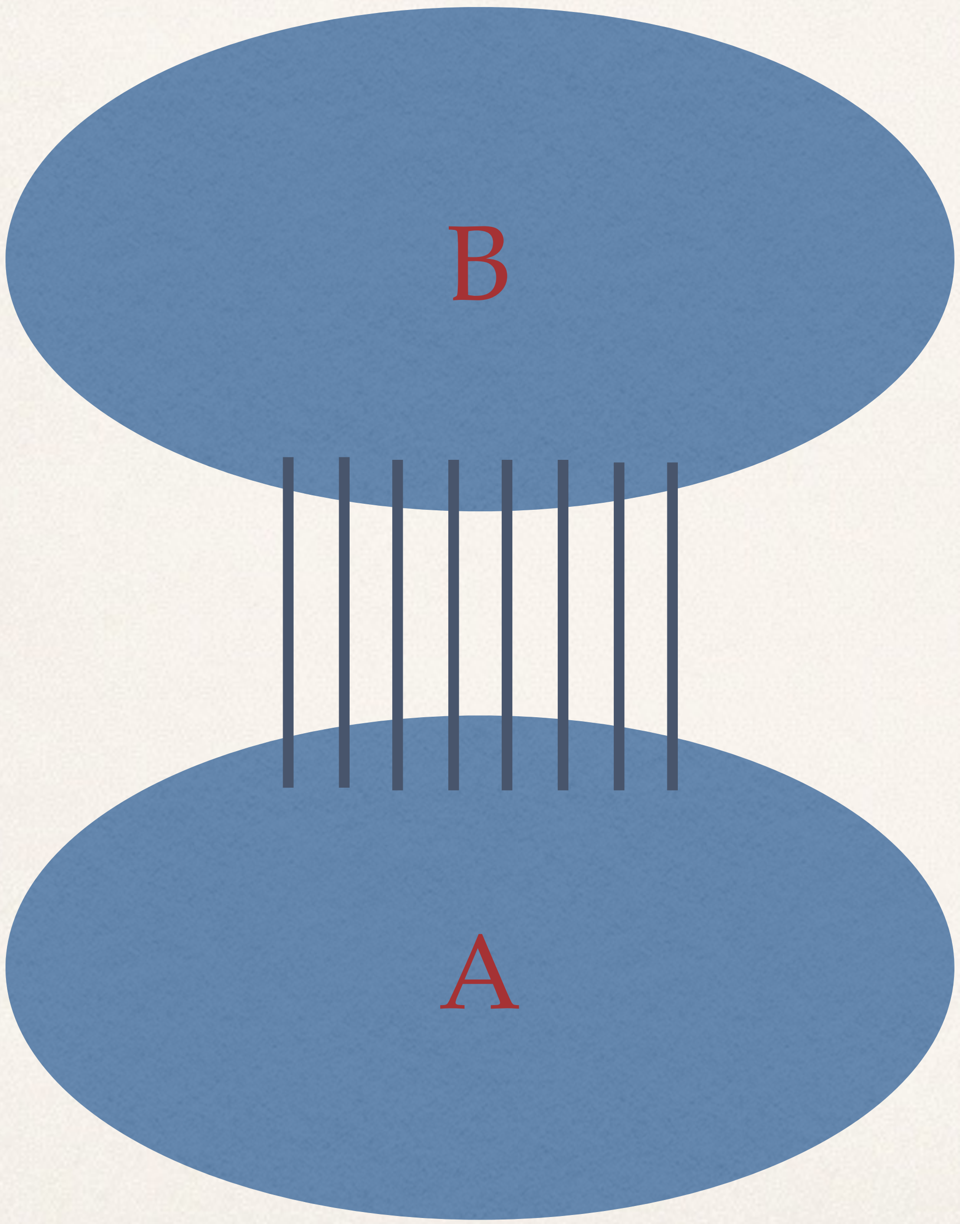
Computing a tree-cut decomposition

approximately

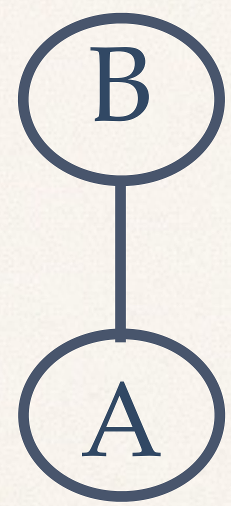
- QUEST: design an algorithm which answers the question ~~exactly~~
  - Given a graph G: produce a tree-cut decomposition of width at most ~~k~~ or declare that tcw > k.

    2k
- …and which runs as quickly as possible.

Sketch of our algorithm



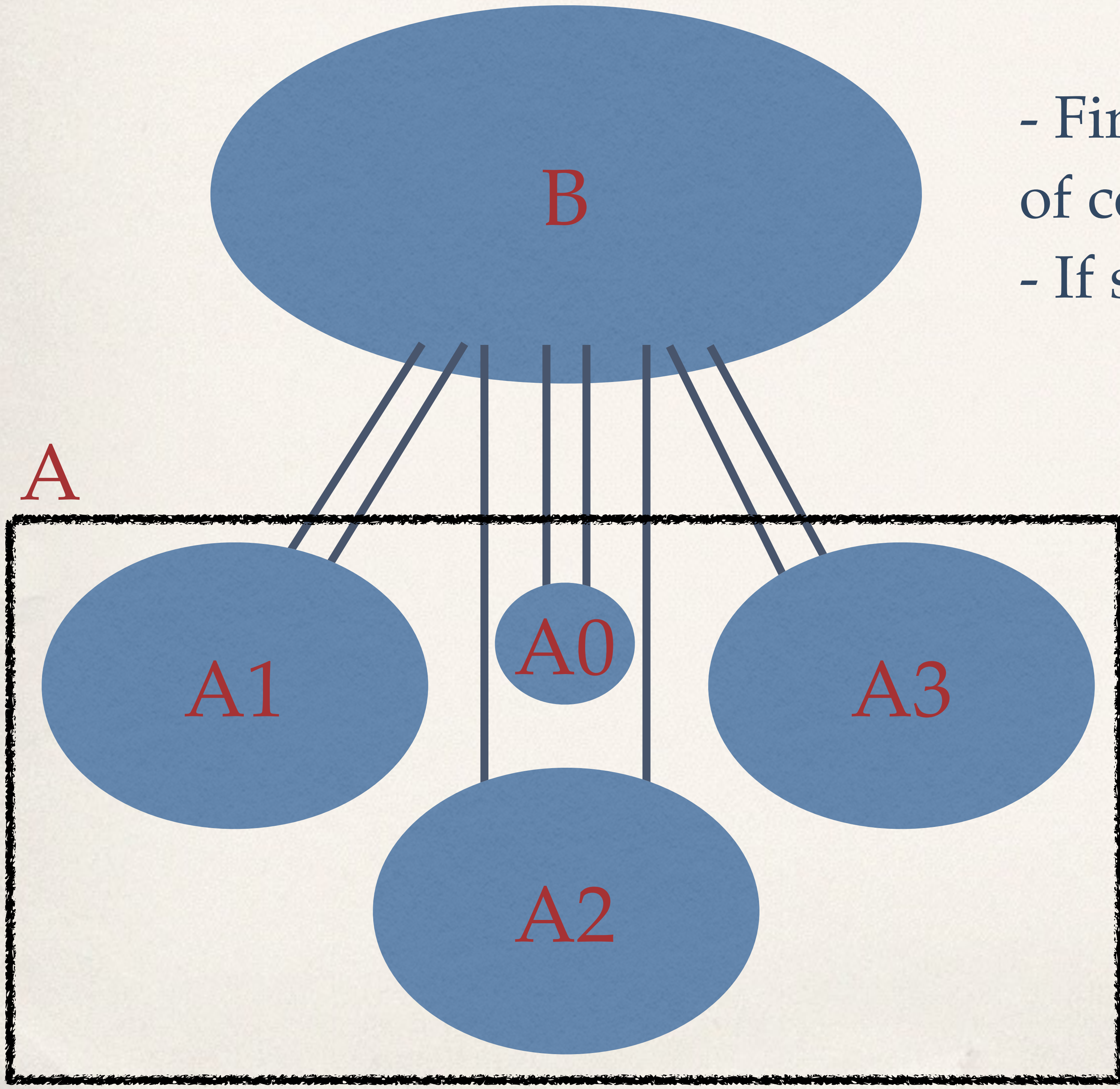
- Find a random cut (A,B) of size $\leq 2k$
- This corresponds to a decomposition



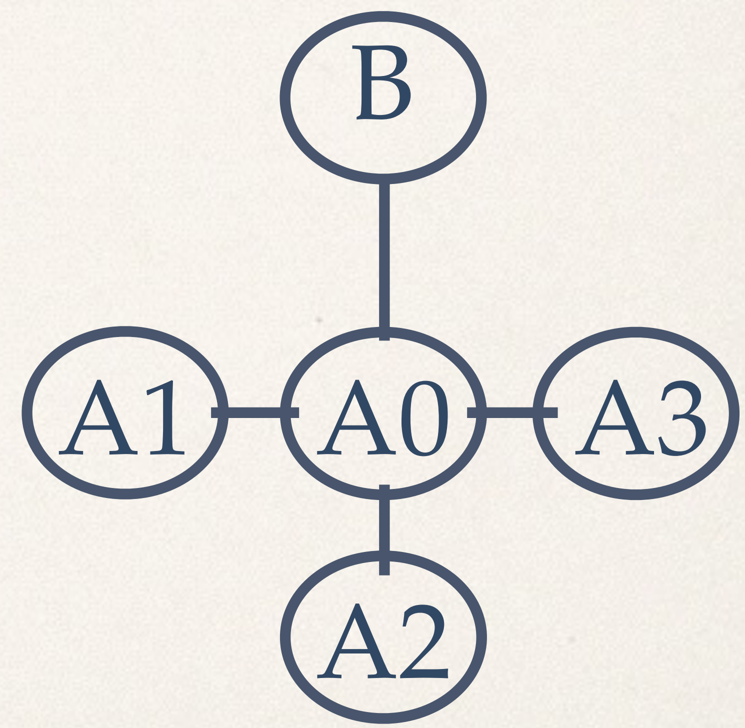
$(T, \chi=\{X_t, t \in V(T)\})$

- Currently, too large bags.
- Idea: "**Grow**" the tree, "**Reduce**" the bag sizes.

Sketch of our algorithm

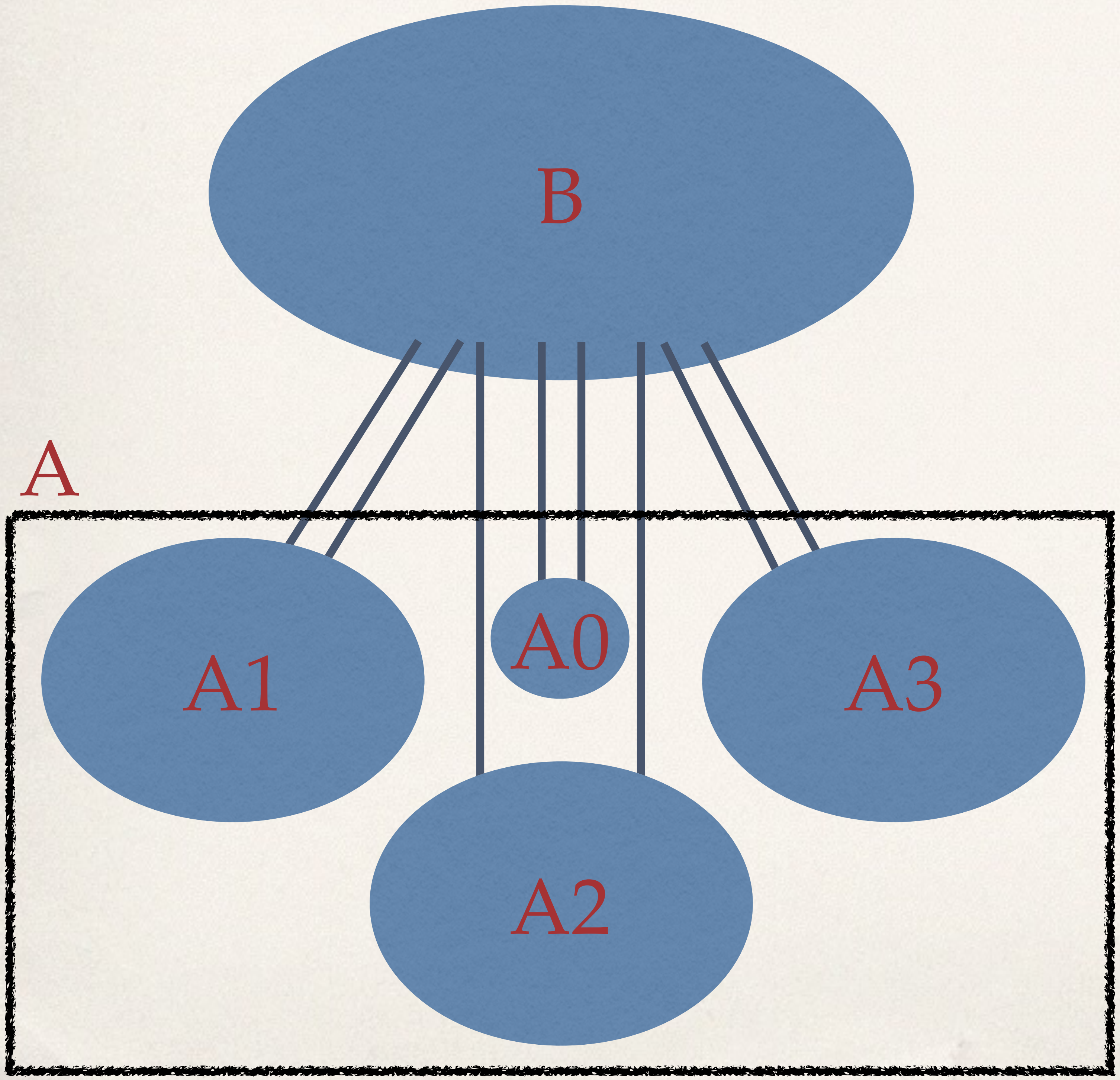


- Find a partition of A meeting a set of conditions (*)
- If such a partition exists - refine A



$(T, \chi=\{X_t, t \in V(T)\})$

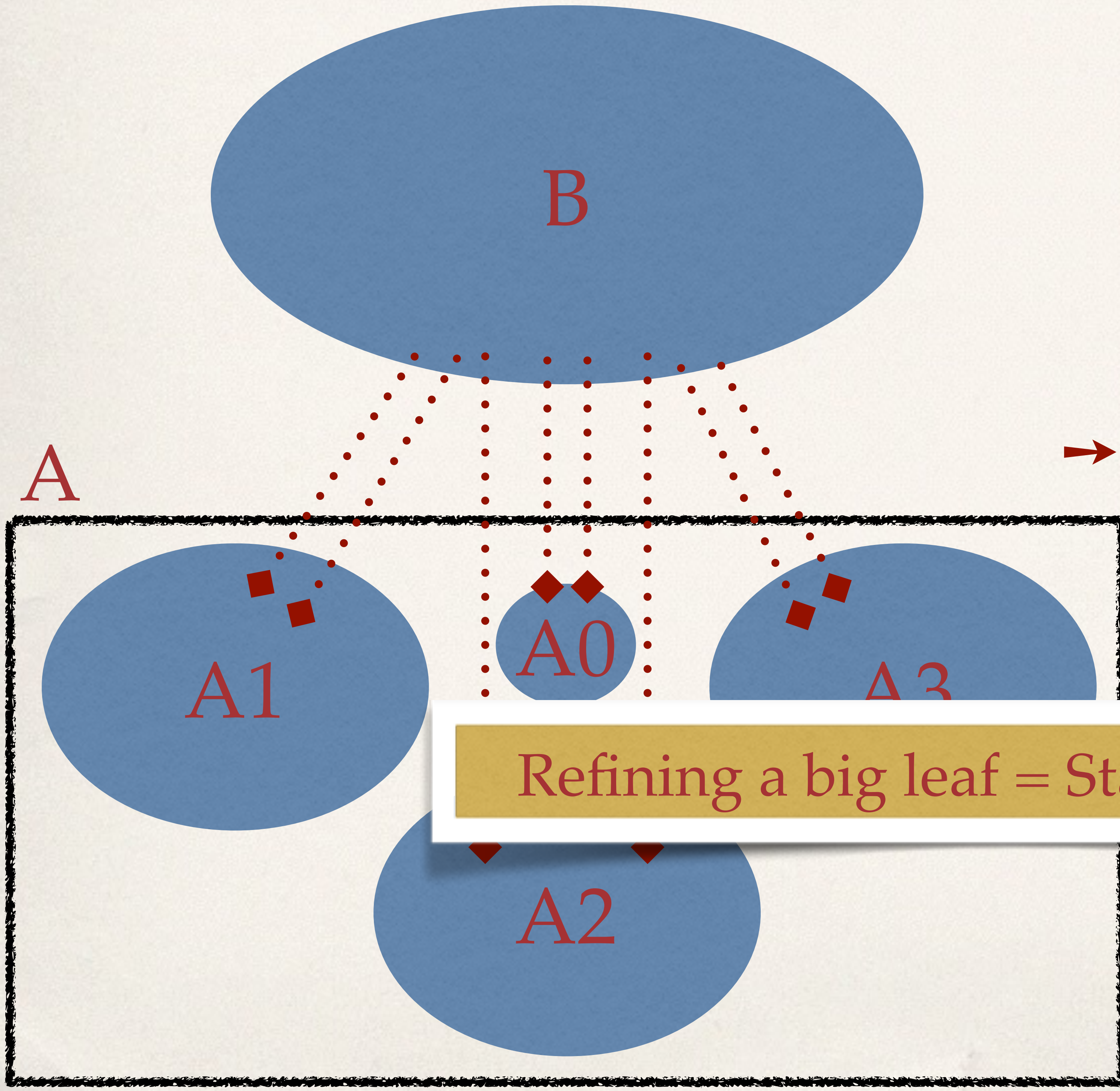
Sketch of our algorithm



Find a partition of A such that

- cut $(A_i, A \setminus A_i) \leq k$, $i \in \{1,2,3\}$
- cut $(A_i, B) \leq k$
- $|A_0|$ + number of parts $\leq k$

Sketch of our algorithm



Find a partition of A such that

- cut (Ai,A ∖ Ai) ≤ k, i∈ {1,2,3}
- ~~cut (Ai,B) ≤ k~~
- number of parts ≤ k

➙ each part Ai has ≤ k“terminals”

Refining a big leaf = Star-Cut Problem

Algorithm for Star-Cut

- Fact
  - tw ≤ 3tc
  - 5-approx ... ender et al. 2013]

- Algorithm
  1. Run Bo ... > k
  2. Dynam ... most 15k^2
     - for each of 15k^2 vertices, guess 'i' s.t. v belongs to Ai
     - keep track of <u>#cut (Ai,A∖Ai)</u> and <u>#terminals in Ai</u>
     - runtime: k^(bagsize) • n

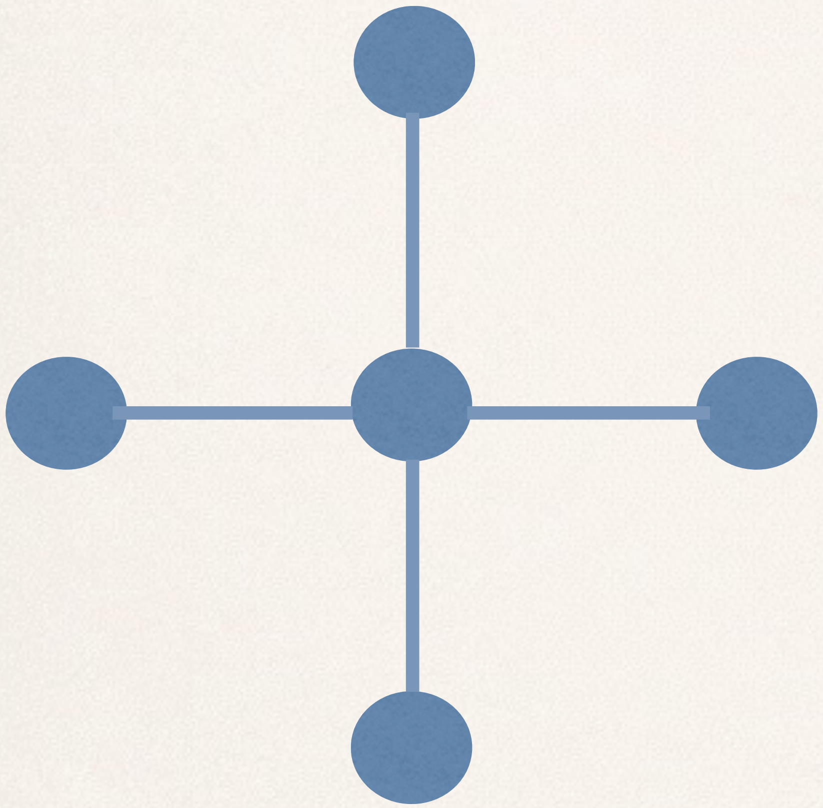
Iteratively solve Star-cut to refine the initial tree-cut decomposition. The entire routine runs in k^O(k^2) • n • n

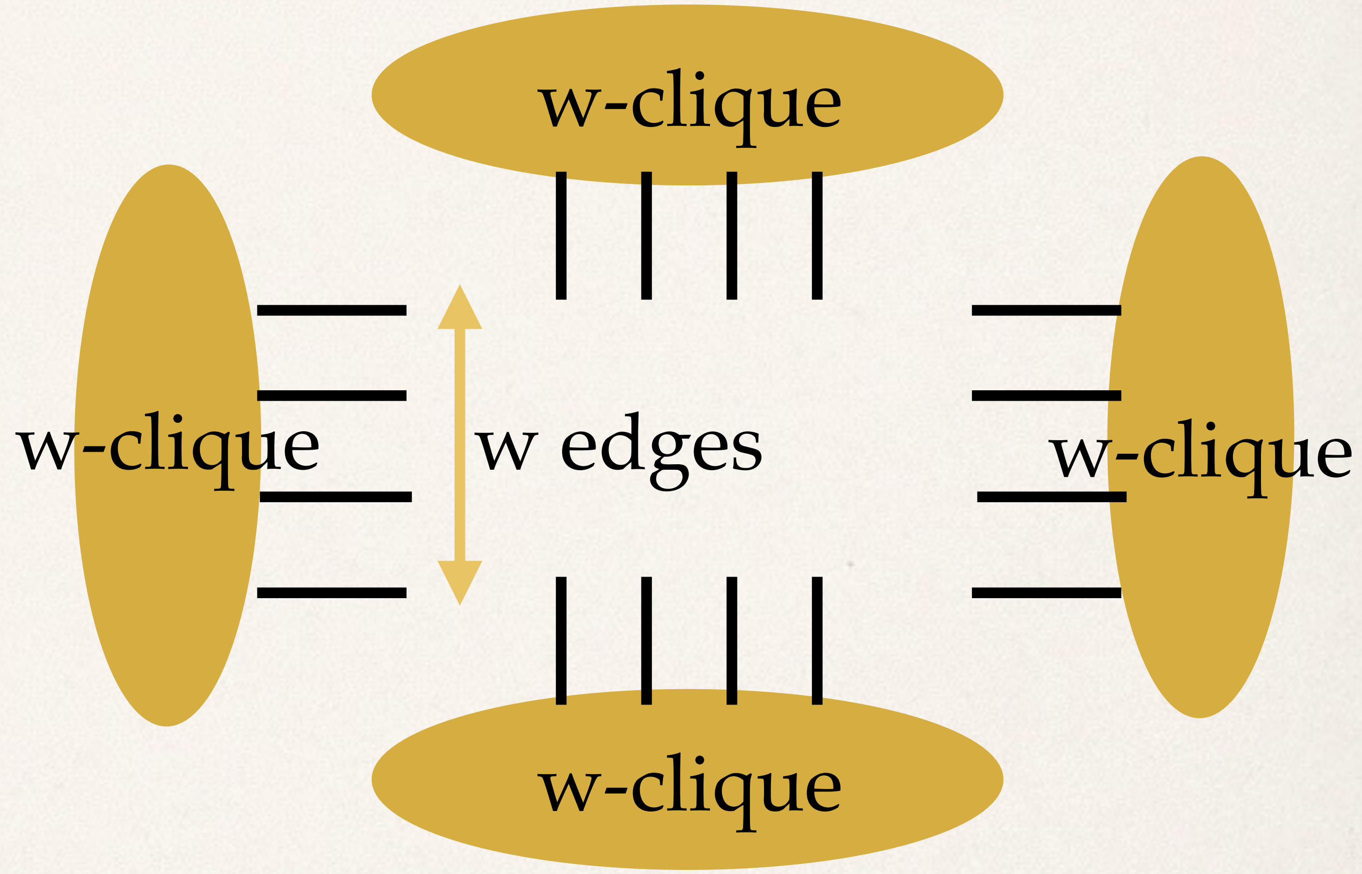
Tree-cut width vs treewidth

- Can the above algorithm be improved? DP can be improved?
- tw = O(tcw^2): in fact the binding function is tight.
- There is an infinite family of graphs whose tree-cut width is w, and treewidth is Ω(tcw^2).

Graphs with tw=Ω(tcw^2)

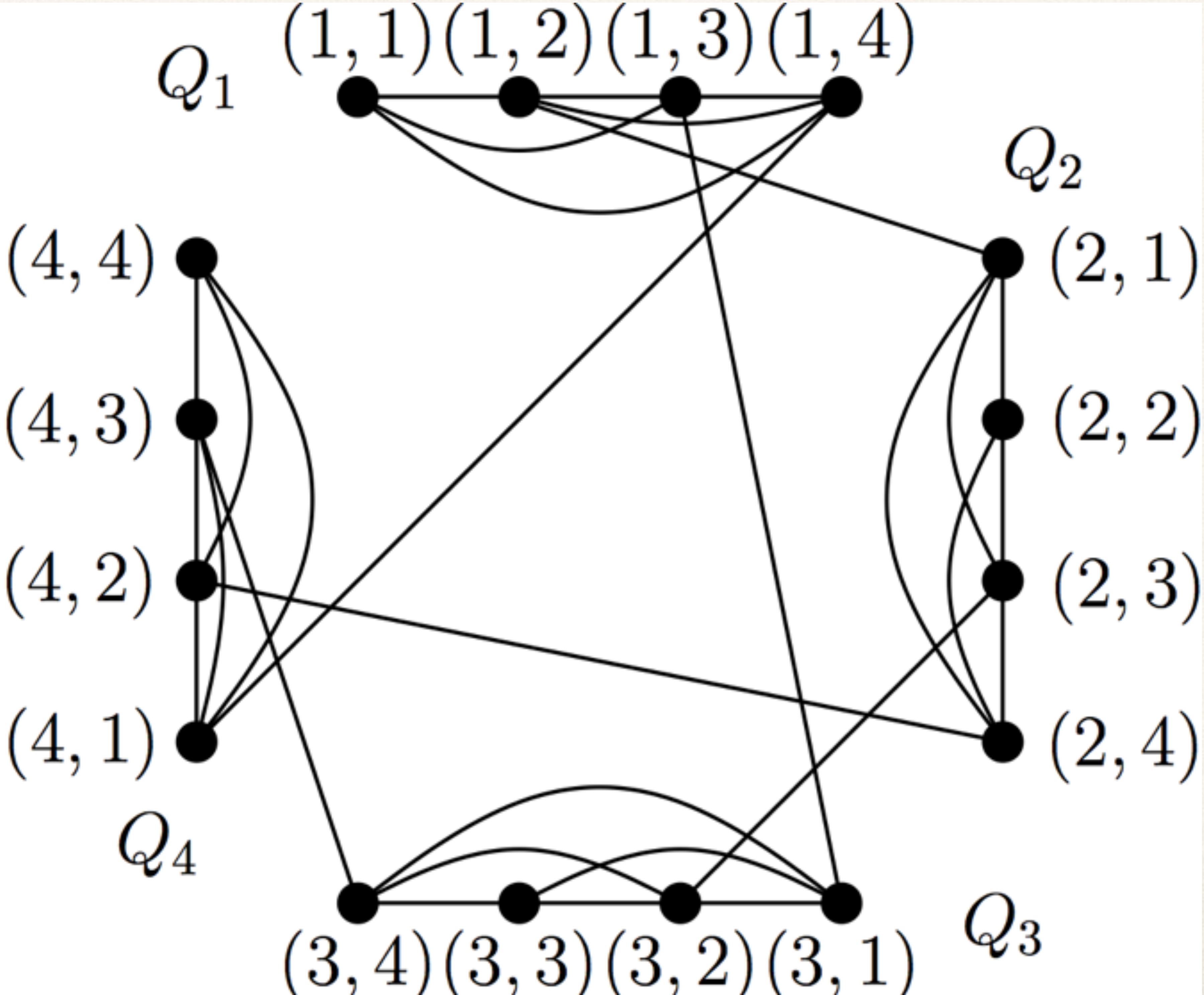
We want to build a graph with tree-cut width w+1



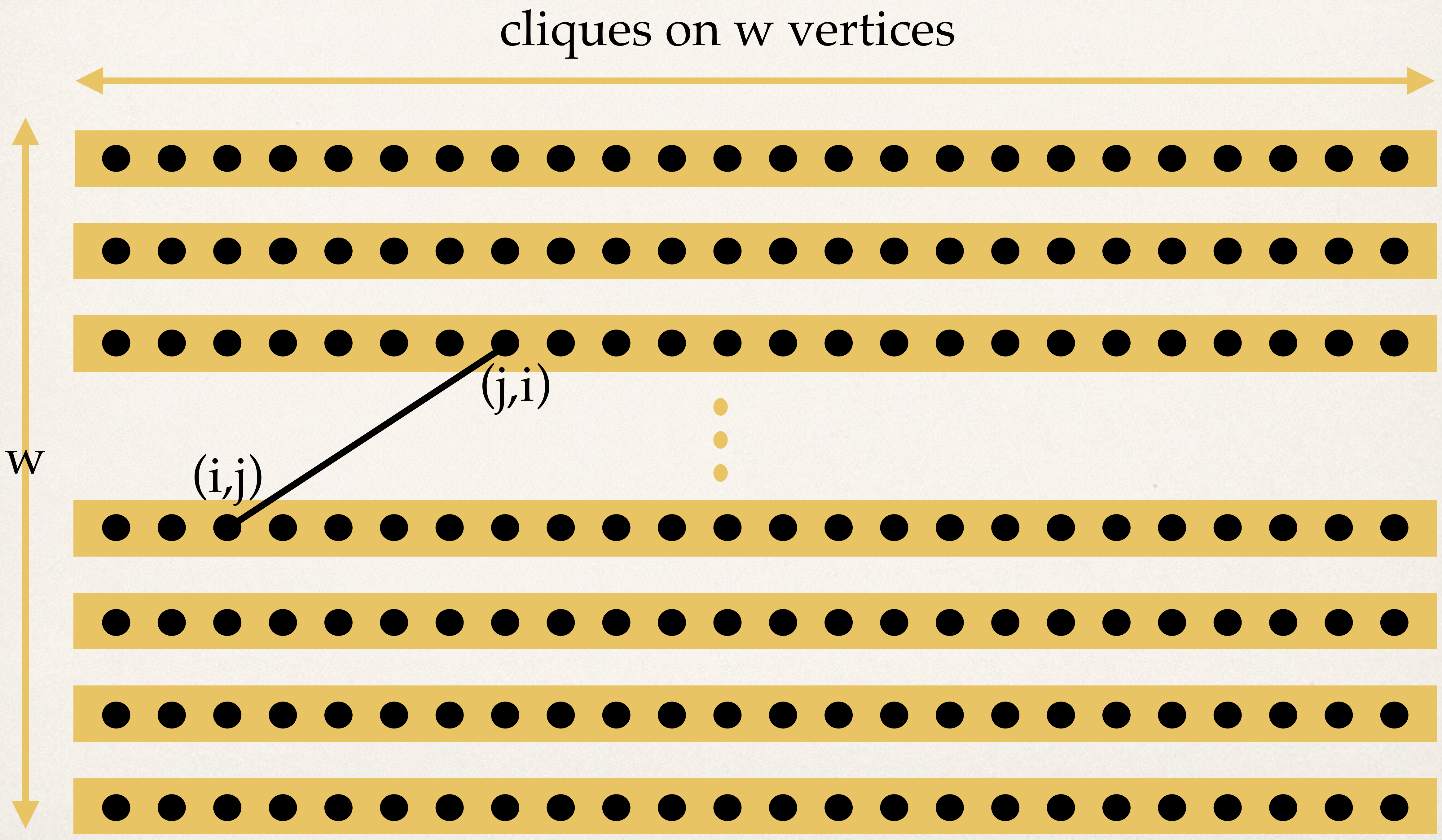


…which looks as simple as possible, while its treewidth is as large as possible.

Graphs with tw=Ω(tcw^2)



Graphs with tw=Ω(tcw^2)



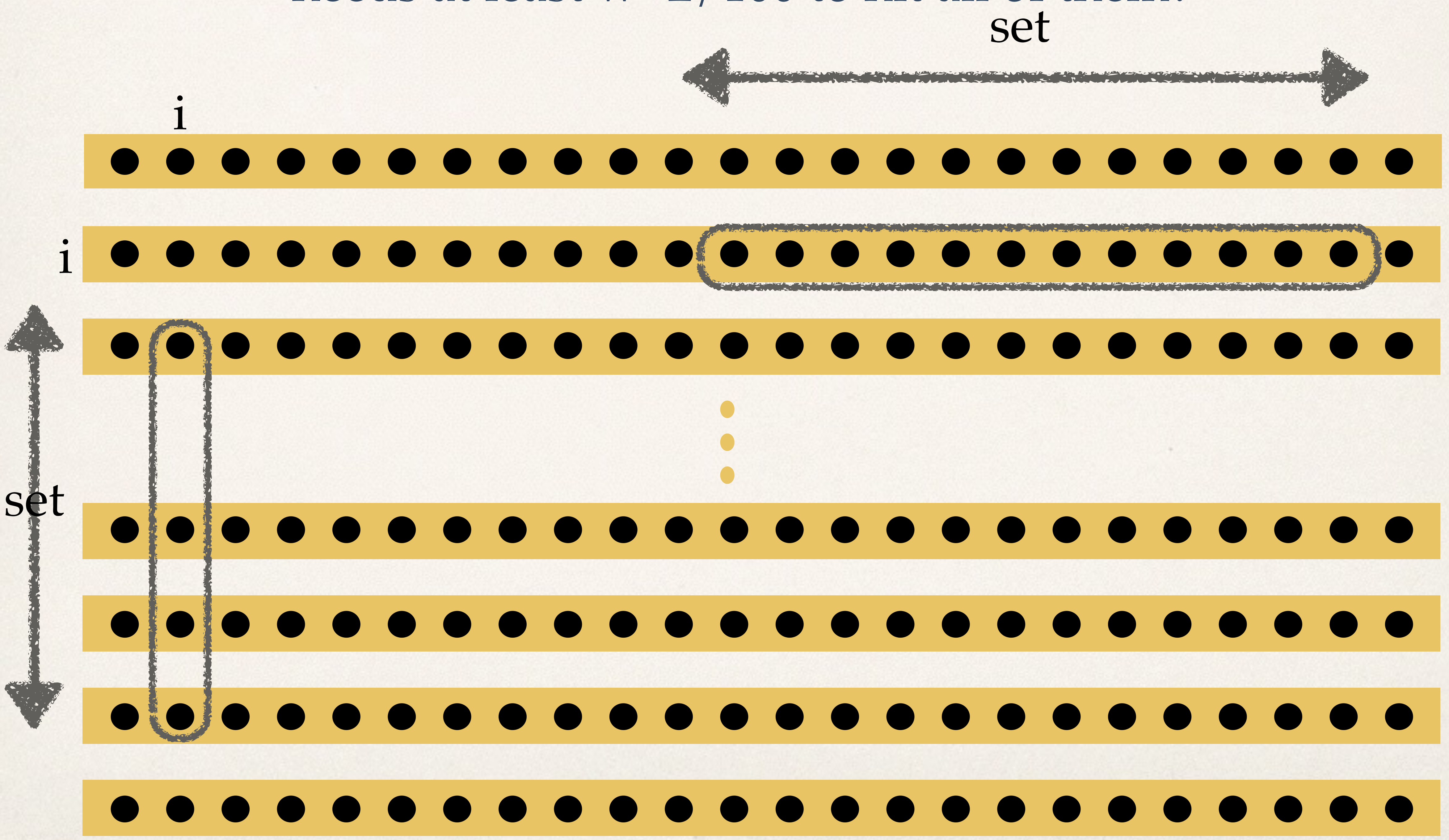
Proving lower bound for tw

- Bramble $\mathcal{B}$ of G: a collection of connected subgraph of G, mutually "touching" each other, i.e. intersecting or adjacent.
- Order of Bramble $\mathcal{B}$: minimum size of a hitting set
- THM [Seymour and Thomas 93]: tw ≥ order of <u>any</u> bramble - 1
- Goal: construct a bramble whose order is w^2/100

Our bramble $\mathcal{B}$: $\forall$i $\in$[w], $\forall$set $\subseteq$ [w]\i of size w/2,

$\mathcal{B}$ contains the induced graph on {(i,j)(j,i): j$\in$ set}

- each, connected? ✔
- mutually touching? ✔
- needs at least w^2/100 to hit all of them?

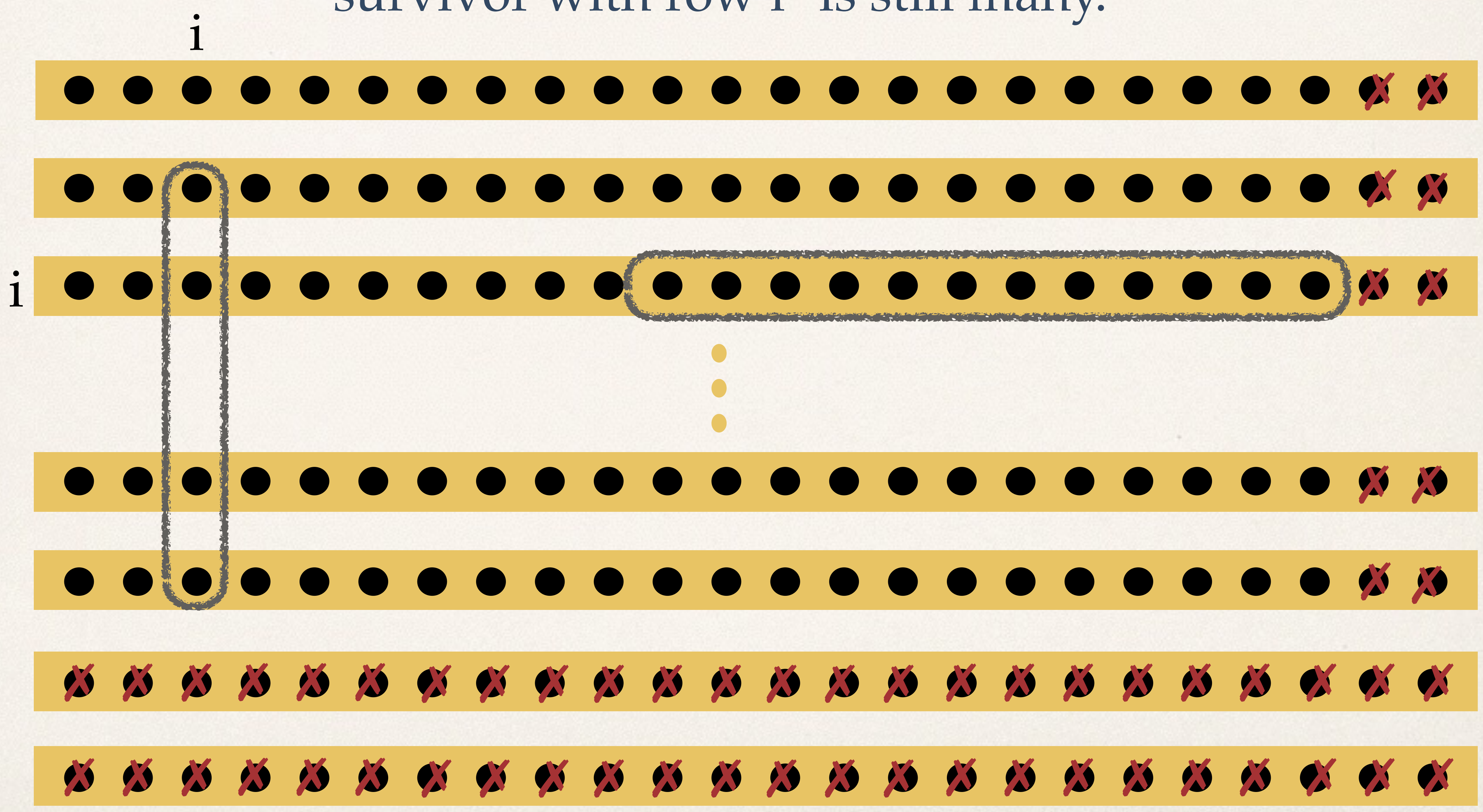


Let ✗ be a hitting set < w^2/100

What if ✗ is randomly distributed…

In real life:

- you can find many rows "i" where still many vertices survive.
- among such "i", you can find one column i* whose common survivor with row i* is still many.



Further Questions

- For problems hard on graphs with small tw:
  are there problems showing different computational behavior on small pw and small tcw? e.g. CDC/CVC and boolean CSP
- Our algorithms run in time k^poly(k)
  Better running time? Or optimal?
  further conditions on graphs to accelerate the runtime?
- 2-approximation runs in w^O(w^2).
  Faster algorithm? exact computation?
- In the end, is tree-cut width an interesting graph

Thanks!