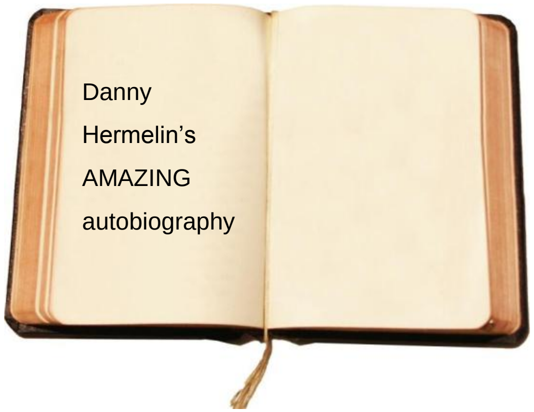
Treewidth Governs the Complexity of Target Set Selection

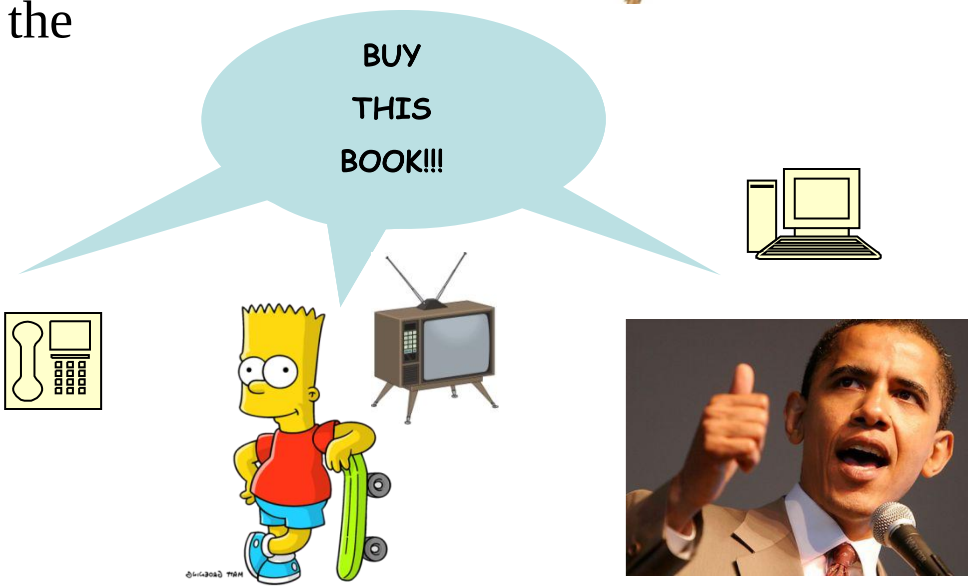
Danny Hermelin
Max Planck Institute for Informatics

Joint work with: O. Ben-Zwi, D. Lokshtanov, I. Newman

Target Set Selection

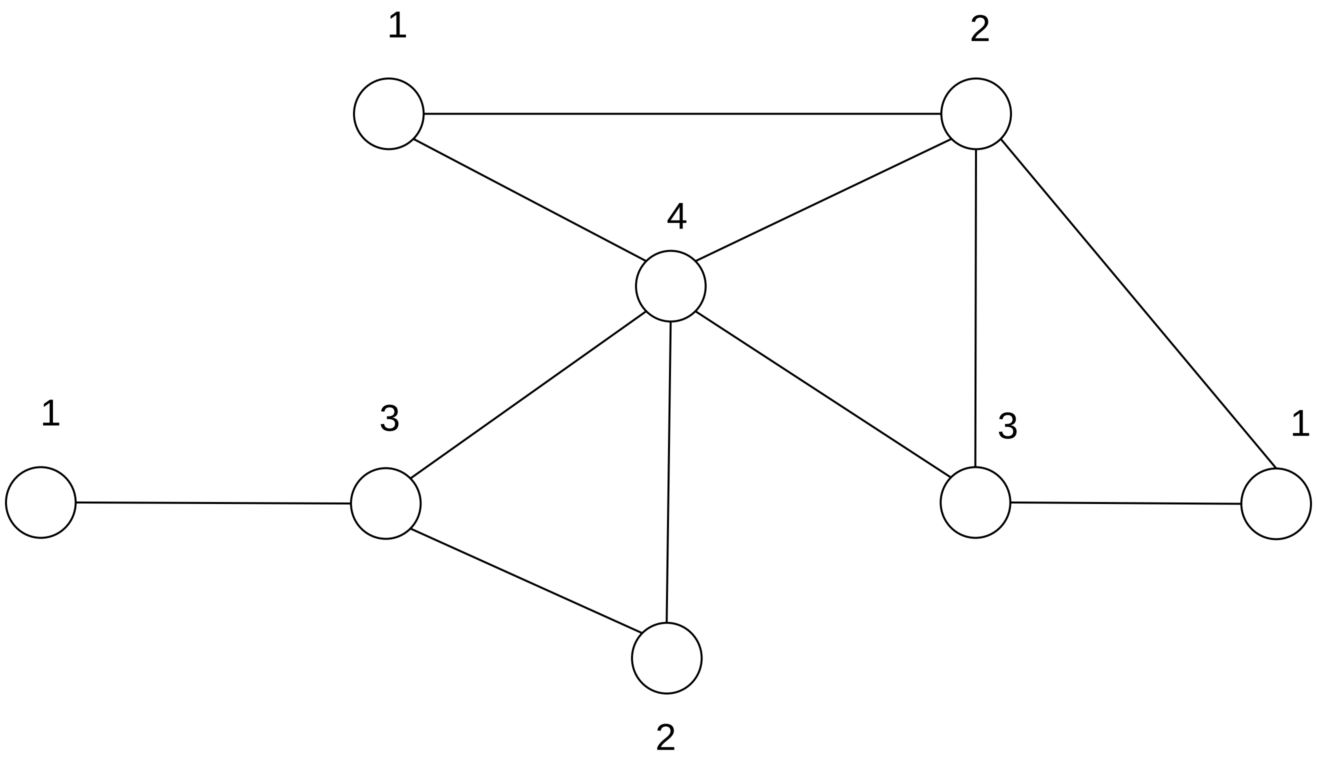
- Suppose you have a new product you want to sell
- Find influential individuals
- Persuade them to spread the word





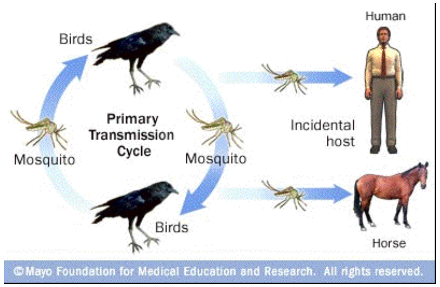
Target Set Selection

- Model as an activation process in a graph (social network):
    - vertices have integer thresholds



Target Set Selection

- Marketing on a social network
- Analyzing diffusion processes of ideas or innovations on social networks
- Predicting virus spread over large populations



Target Set Selection

Formal Definition

- **Instance**: Integers k,l and a graph G with thresholds $t : V(G) \rightarrow \mathrm{N}$
- **Goal:** Find $S \subseteq V(\mathrm{G})$, $|S| \leq k$, that activates at least l vertices in G

Previous Work:

- Domingos and Richardson [KDD 2001]
  - Introduced diffusion process into CS
- Kempe, Kleinberg and Tardos [KDD 2003, ICALP 2005]
  - Inapproximability result of $n^{(1-\varepsilon)}$ for maximizing l, given k
  - Approximation when thresholds are uniformly distributed
- Chen [SODA 2008]
  - Polylogarithmic inapproximability result for minimizing k, given l
  - Poly-time algorithm for trees (worst case thresholds)

Target Set Selection

Formal Definition

- **Instance**: Integers k,l and a graph G with thresholds $t : V(G) \rightarrow$ N.
- **Goal:** Find $S \subseteq V$(G) , $|S| \leq k$, that activates at least l vertices in G.

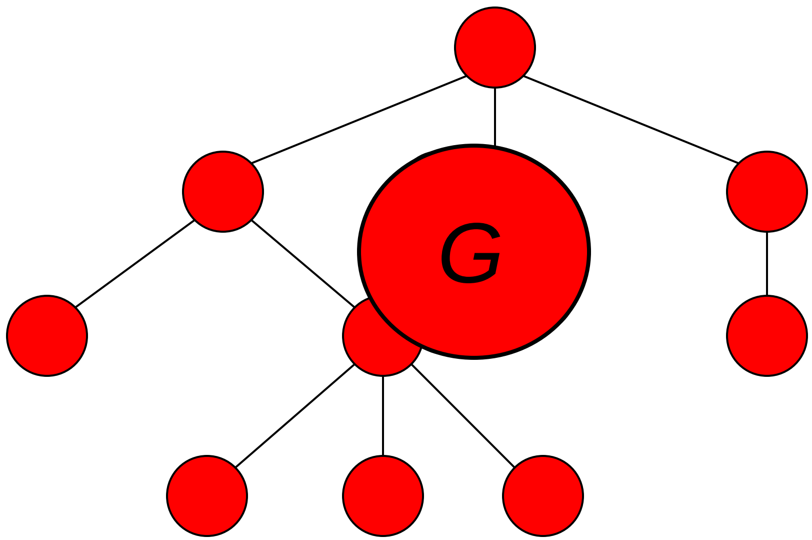
Our Results:

- For a graph with n vertices and treewidth w:
  - TSS can be solved in $n^{O(w)}$
  - TSS cannot be solved in $n^{o(\sqrt{w})}$

treewidth governs the complexity of TSS

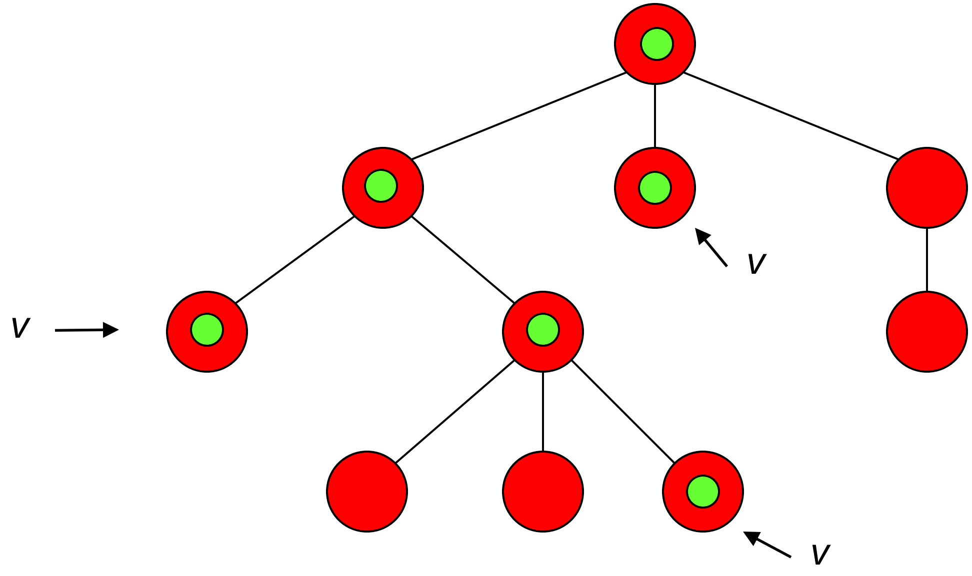
The Treewidth Parameter

- A tree decomposition of *G* is a tree whose nodes are subgraphs of *G* s.t.:
    1. The union of all subgraphs is *G*.



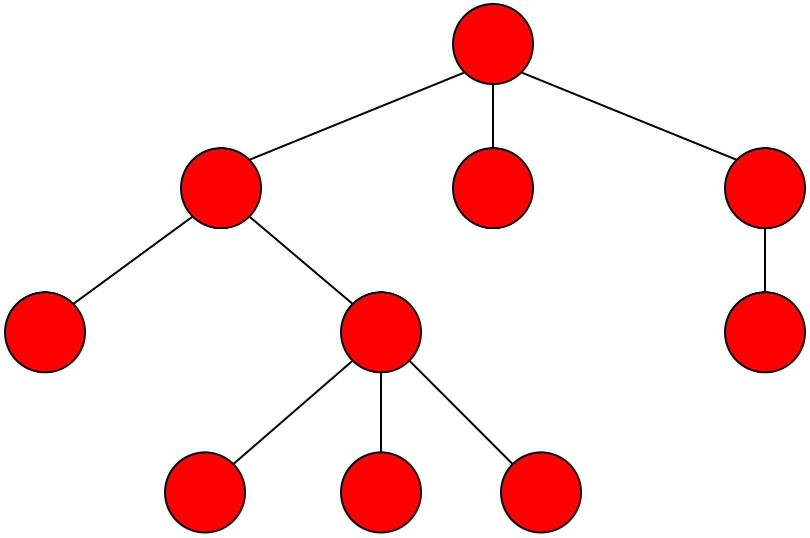
The Treewidth Parameter

- A tree decomposition of G is a tree whose nodes are subgraphs of G s.t.:
  1. The union of all subgraphs is G.
  2. For all $v \in V(G)$, the collection of nodes containing v is connected.



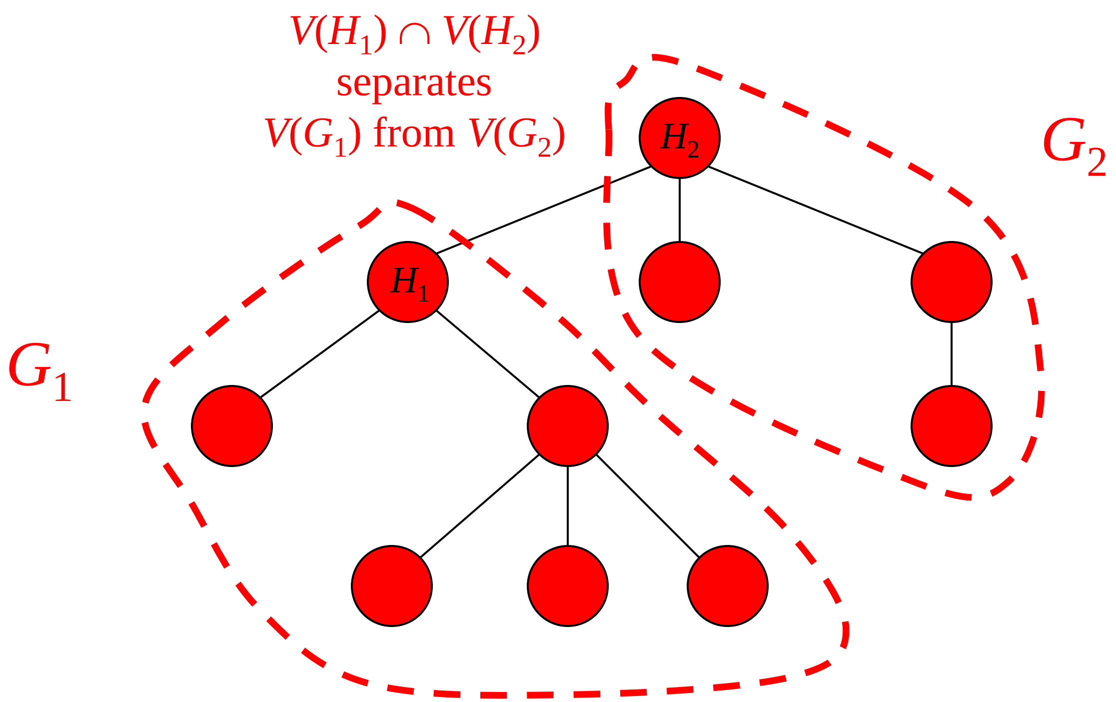
The Treewidth Parameter

- width of decomposition := max $|V(H)|$ over all subgraphs (tree nodes) H
- treewidth of G := min width tree-decomposition of G



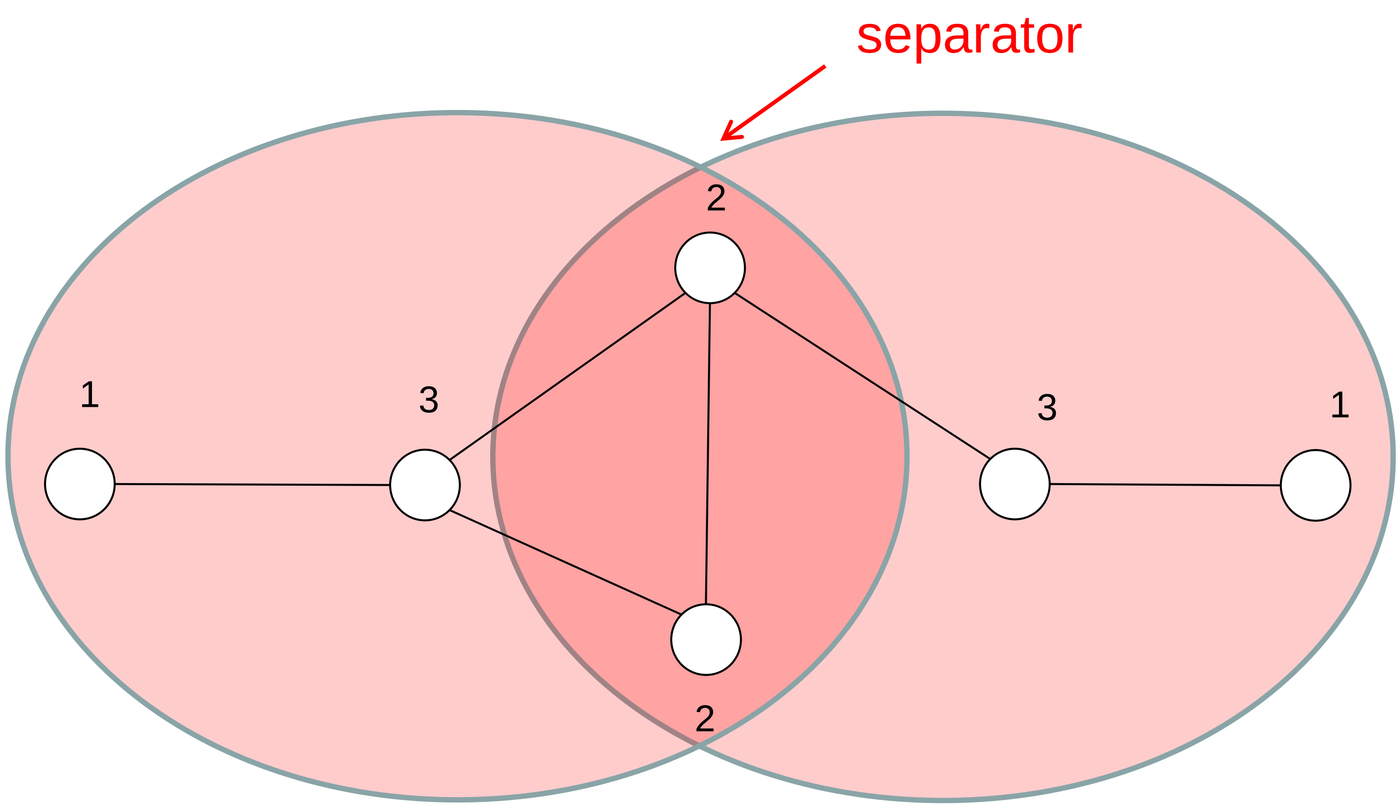
The Treewidth Parameter

- Separation property of tree-decompositions
- Dynamic-programming in bottom-up fashion



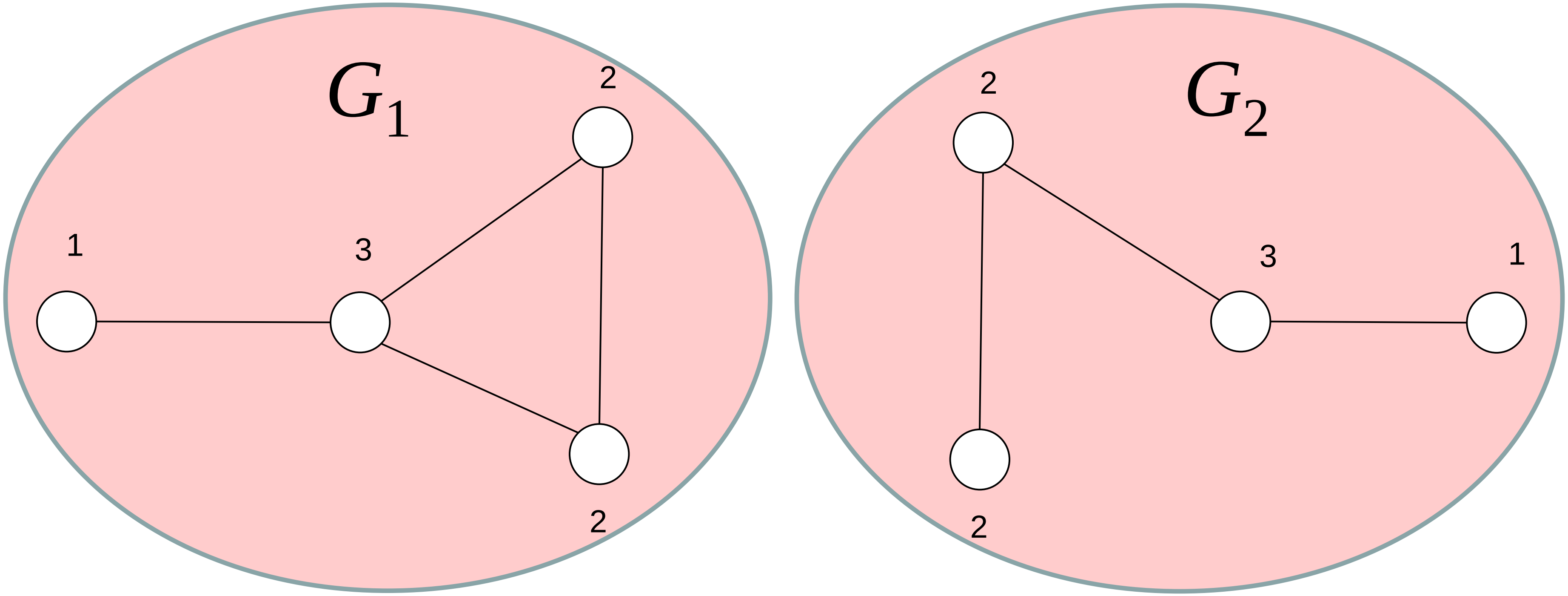
Algorithm for Bounded Treewidth

- Use dynamic-programming in bottom-up fashion



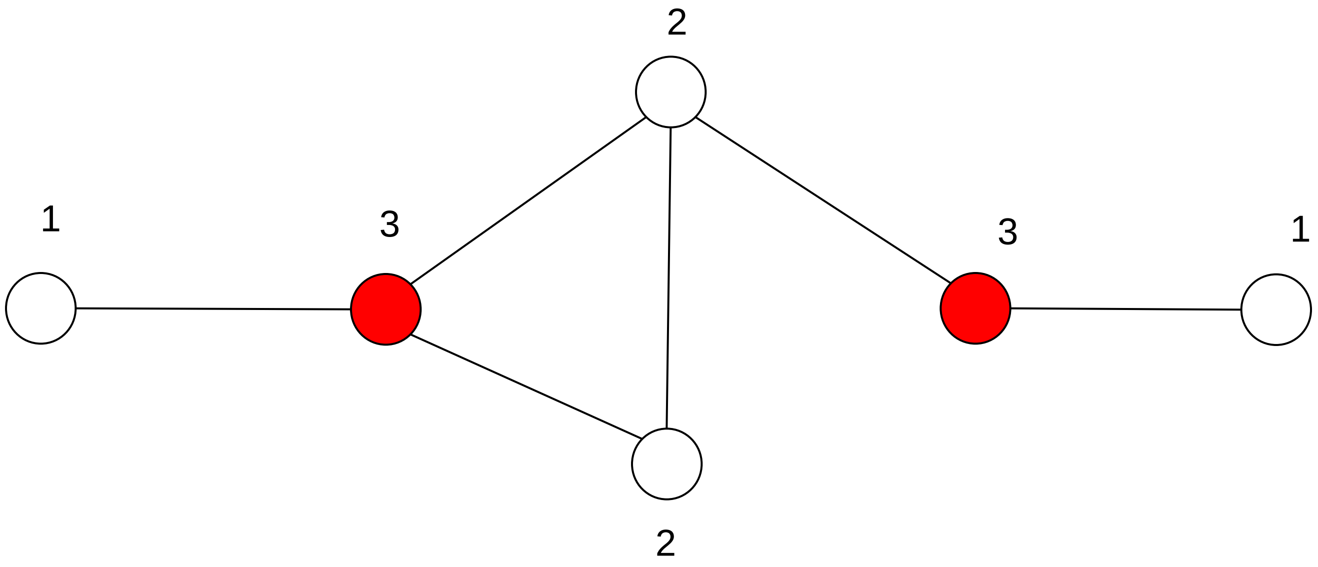
Algorithm for Bounded Treewidth

- Use dynamic-programming in bottom-up fashion
  - Combine solutions from G_1 and G_2 into solutions for G



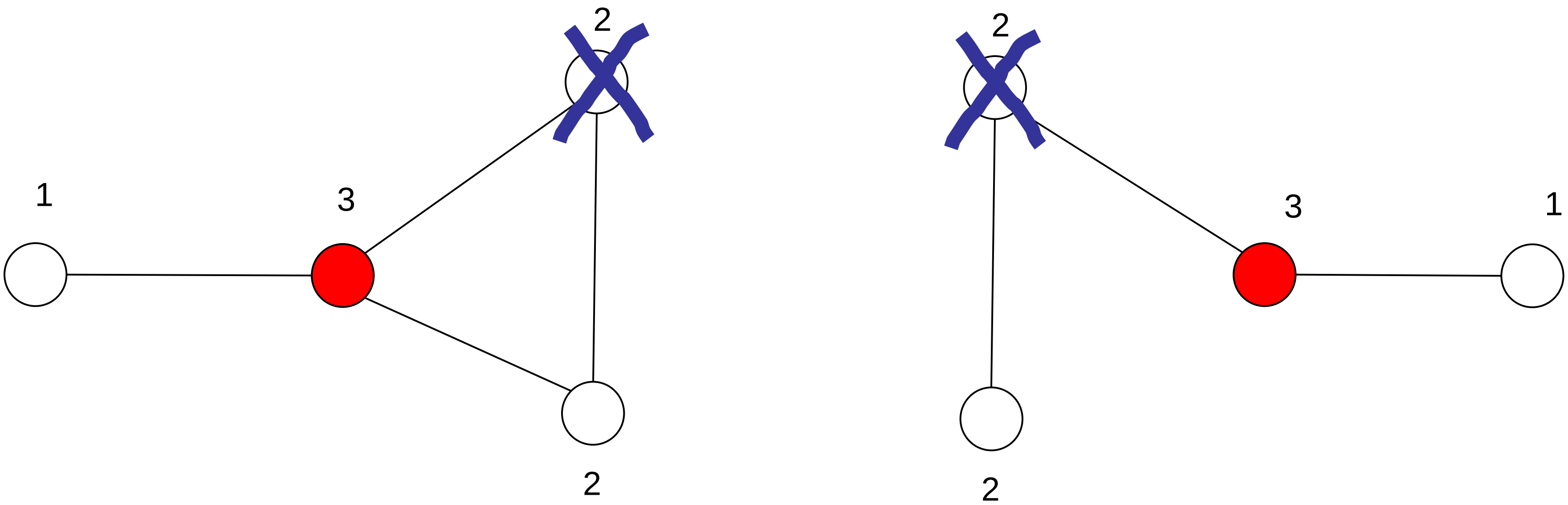
Algorithm for Bounded Treewidth

- Problem 1
  - Solution in G is not a solution in $G_1 + G_2$



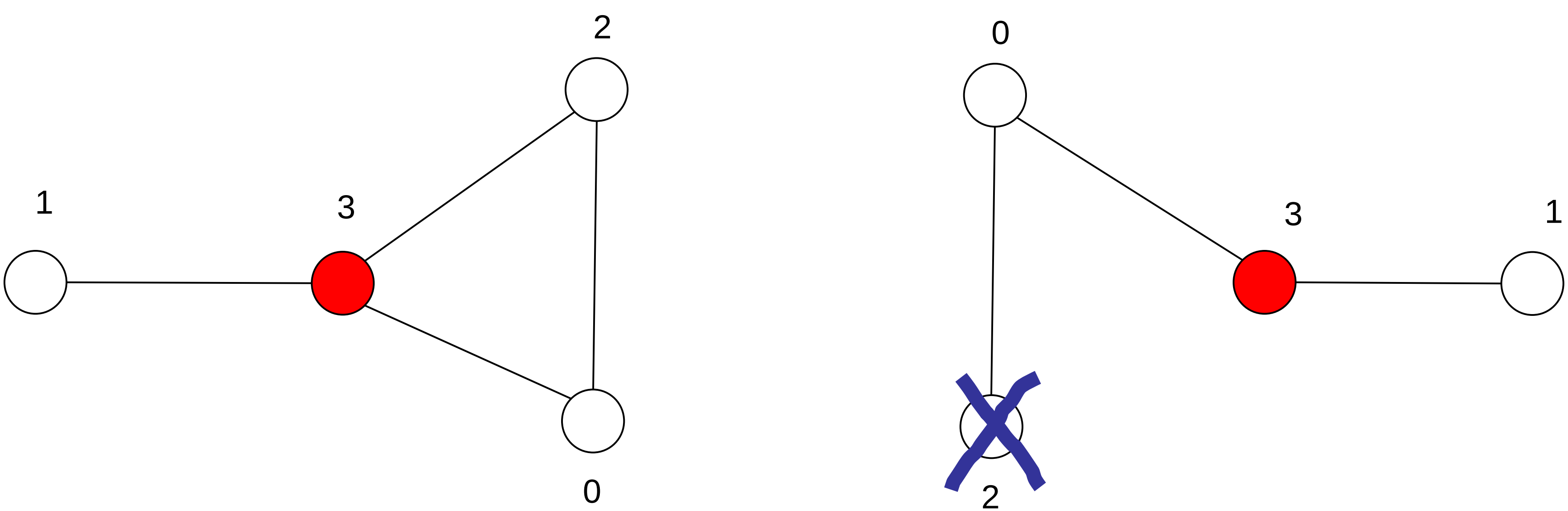
Algorithm for Bounded Treewidth

- Problem 1
  - Solution in G is not a solution in $G_1 + G_2$



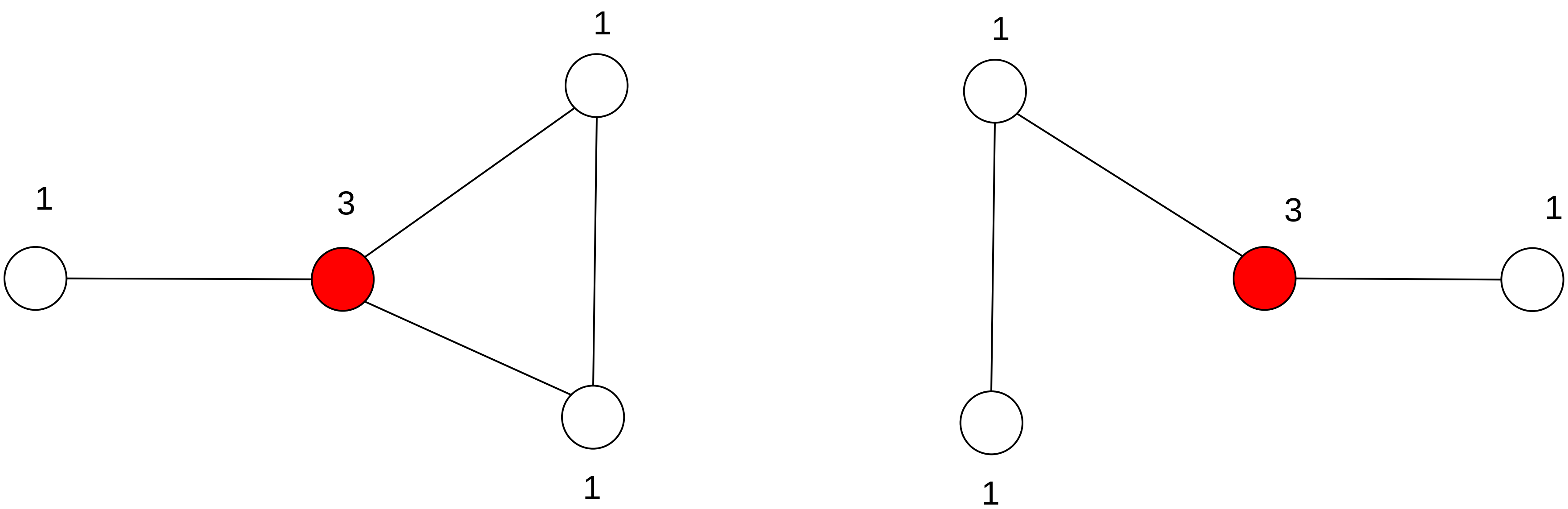
Algorithm for Bounded Treewidth

- Solution:
  - Try all possible thresholds at the separator
  - Merge if they sum-up to (at-least) original thresholds



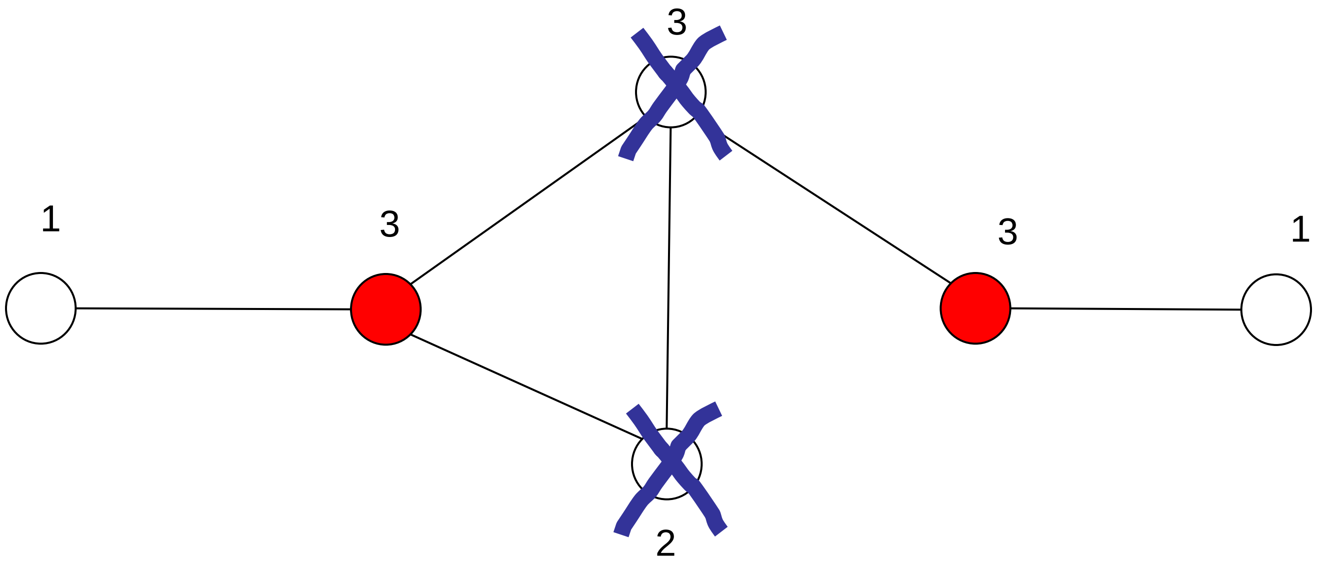
Algorithm for Bounded Treewidth

- Solution:
  - Try all possible thresholds at the separator
  - Merge if they sum-up to (at-least) original thresholds



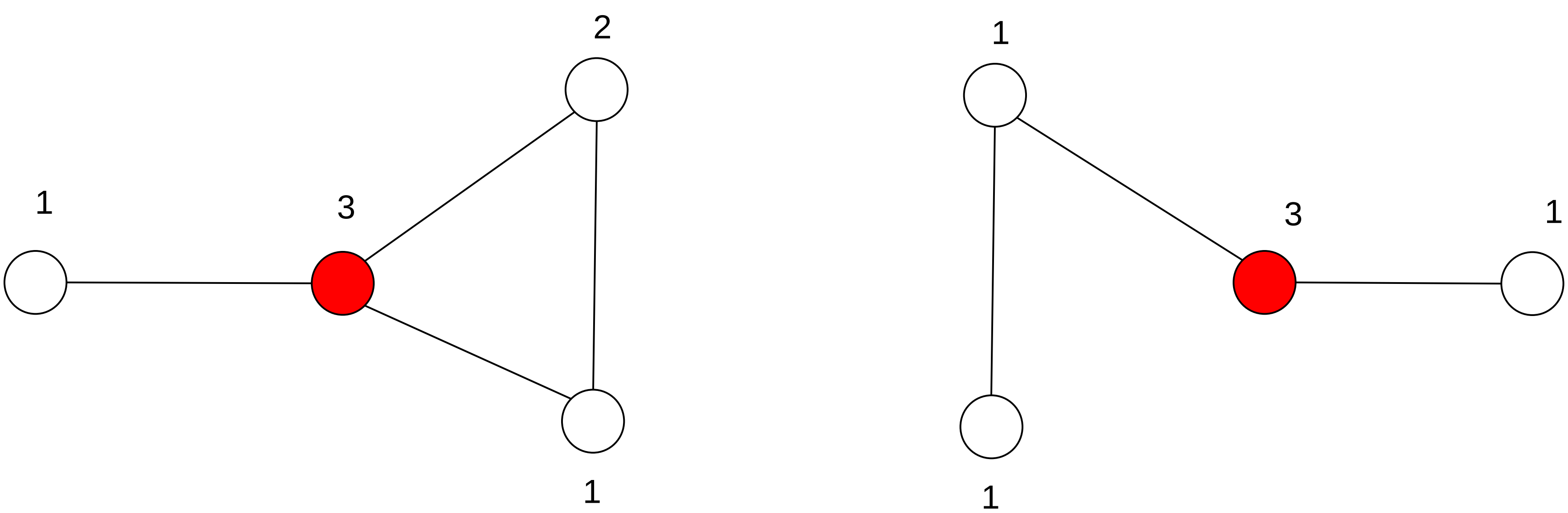
Algorithm for Bounded Treewidth

- Problem 2
  - Solution in $G_1 + G_2$ is not a solution in G



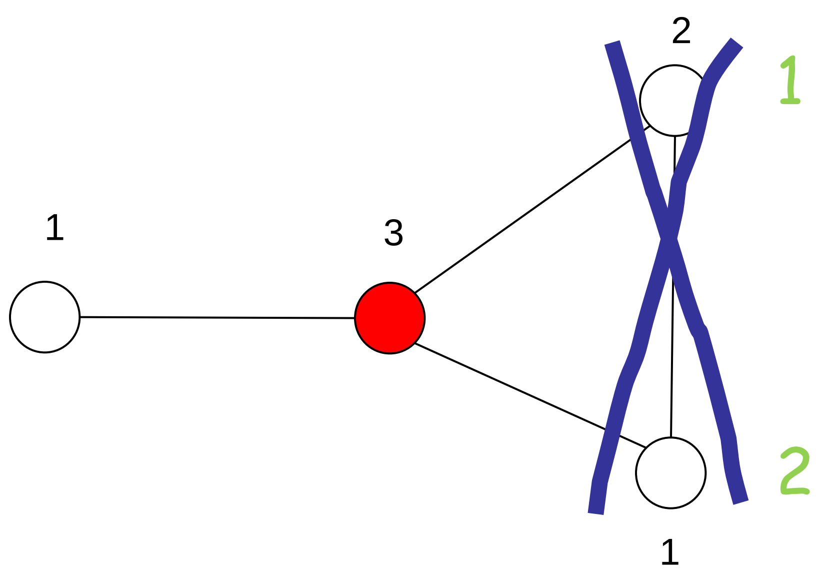
Algorithm for Bounded Treewidth

- Problem 2
  - Solution in $G_1 + G_2$ is not a solution in G
    - Vertices in G_1 and G_2 get activated in different order - deadlock in G



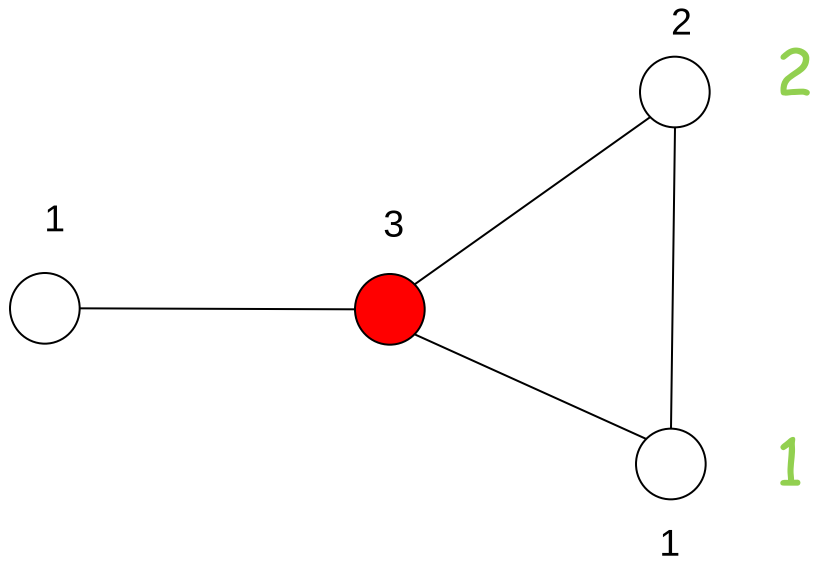
Algorithm for Bounded Treewidth

- Solution:
  - Try all possible activation orderings at the separator.
  - Merge solutions which have the same order.



Algorithm for Bounded Treewidth

- Solution:
  - Try all possible activation orderings at the separator.
  - Merge solutions which have the same order.



Algorithm for Bounded Treewidth

- Algorithm outline:
  1. Try all possible thresholds at the separator $= n^{O(w)}$
  2. Try all possible activation orderings at the separator $= w^{O(w)}$
  3. Merge solutions s.t.:
     - Thresholds sum up (at least) to original
     - Same ordering
  4. Total computation on each node $= n^{O(w)}$
  5. Total time complexity : $O(n)$ nodes * $n^{O(w)} = n^{O(w)}$

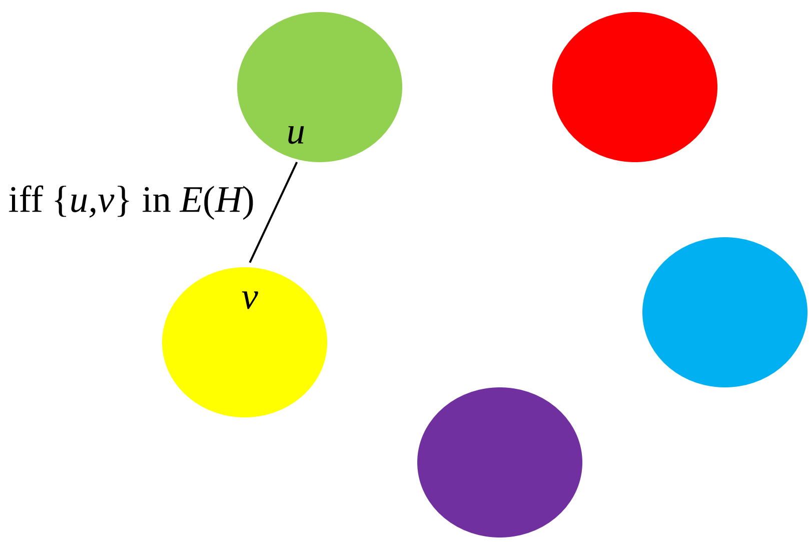
Lower Bounds

- Theorem [Chen et al. CCC '04]: k-Clique cannot be solved in $n^{o(k)}$ unless all problems in **SNP** can be solved in sub-exponential time.
- Reduce k-Clique to TSS in poly-time s.t.:
  - An instance (H,k) of k-Clique will reduced to an instance of (G,s) of TSS s.t. $tw(G) = O(k^2)$.
- Combined with theorem above we get:
  - TSS cannot be solved in $n^{o(\sqrt{\ w})}$ unless all problems in **SNP** can be solved in sub-exponential time.

- Intermediate problem: Multicolored Clique.

Lower Bounds

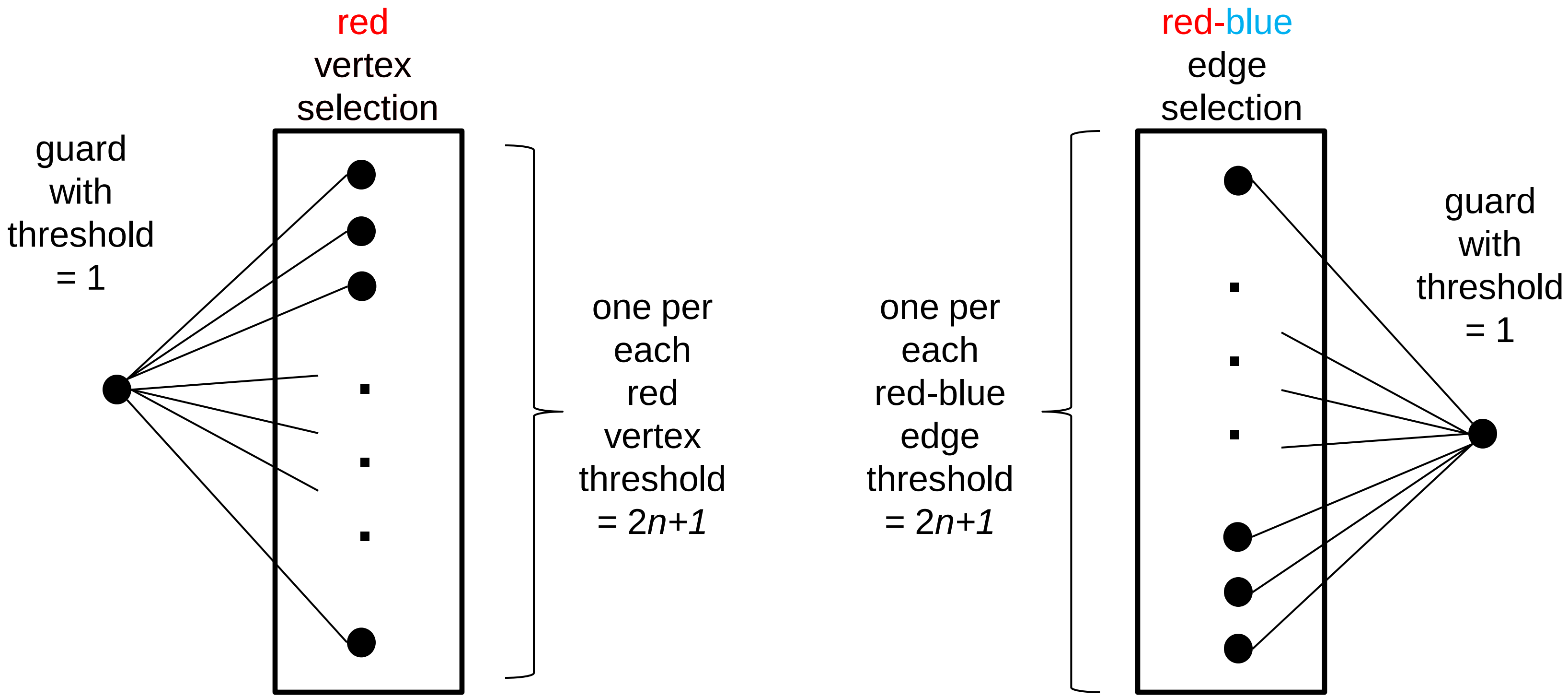
- **Lemma [folklore]**: An instance (H,k) of k-Clique can be reduced to an instance (H',k) of k-Multicolored Clique, and vice-versa.
  - $\Leftarrow$: Remove all edges in each color class, then remove colors.
  - $\Rightarrow$: Create k copies of H, and add adjacencies in a natural manner:



max planck institut
informatik

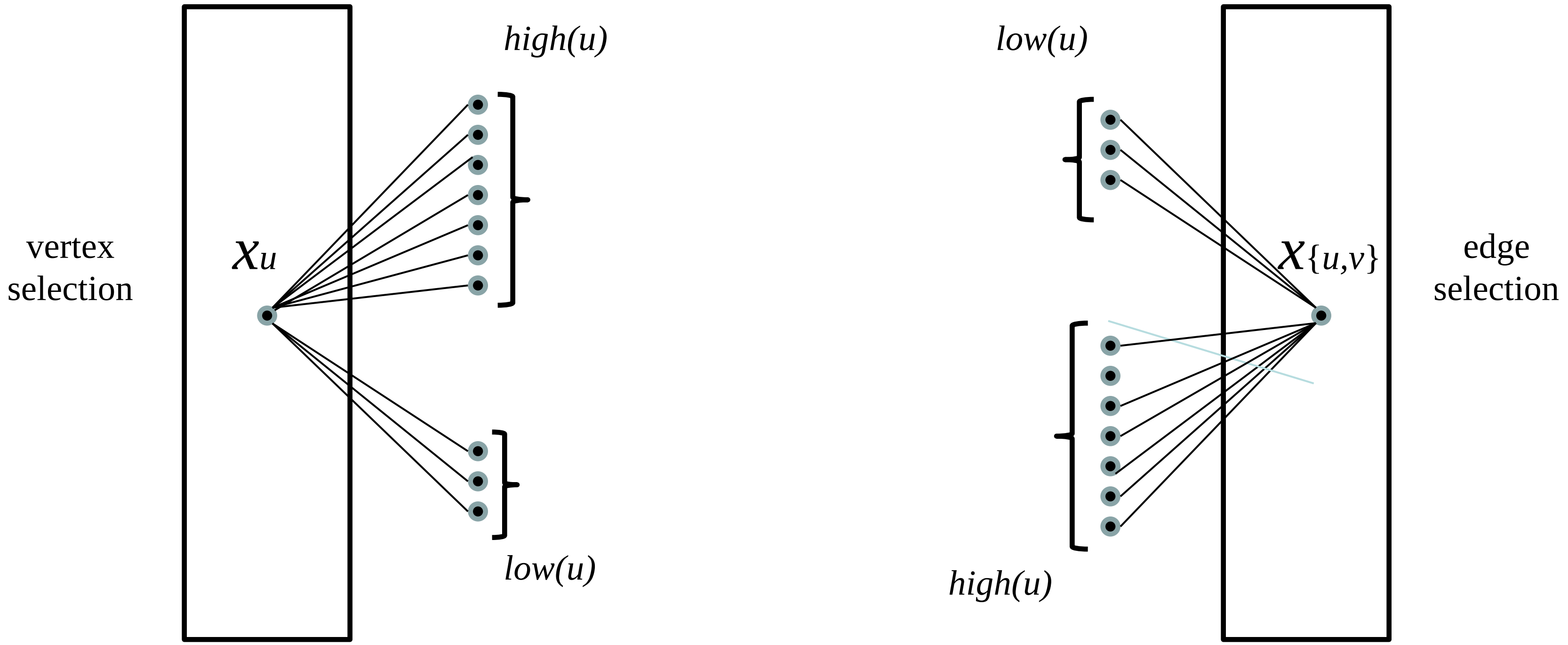
Lower Bounds

- From k-Multicolored Clique to TSS:
  - Selection gadget for each k color class and each $\binom{k}{2}$ edge-color class.



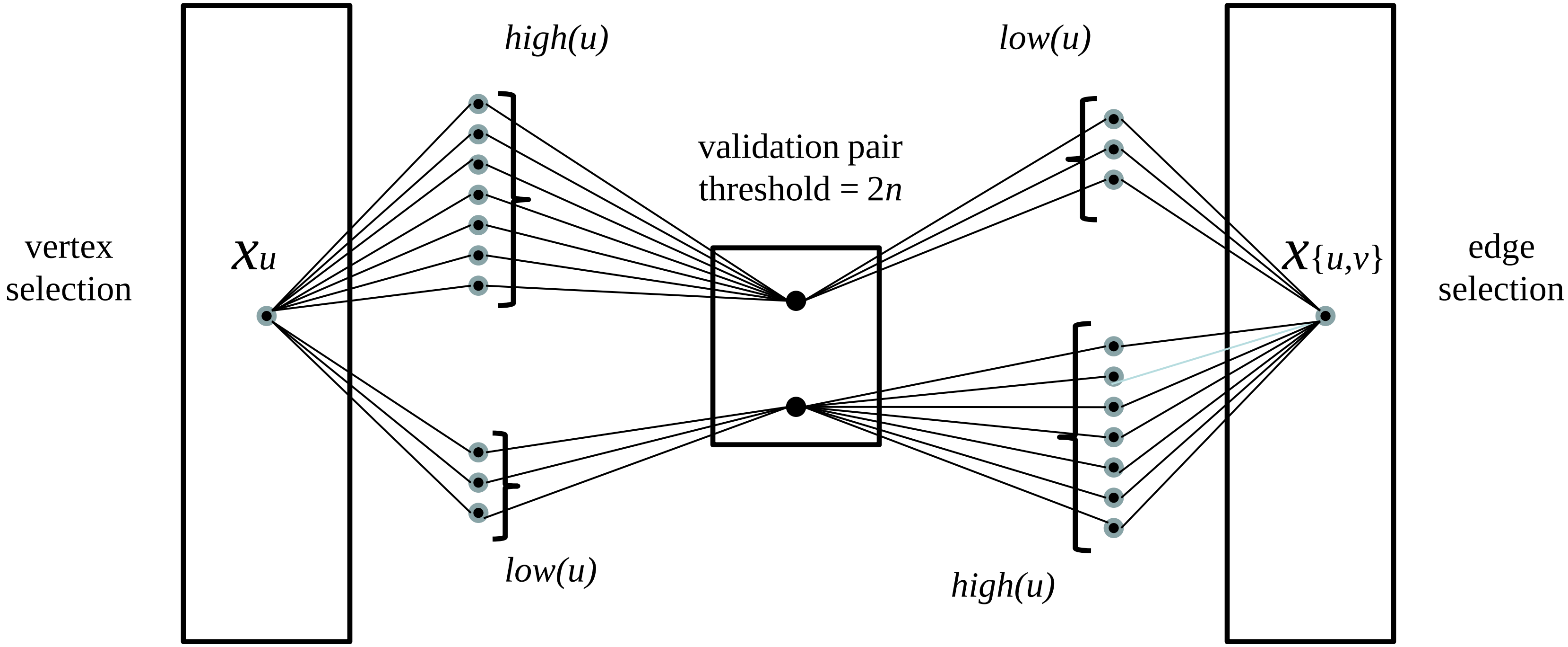
Lower Bounds

- From *k*-Multicolored Clique to TSS:
  - Assign each vertex unique ids
    - $low(v) \in \{1,\ldots,n\}$ and $high(v) := 2n - low(v)$
  - Add connectors from selection vertices



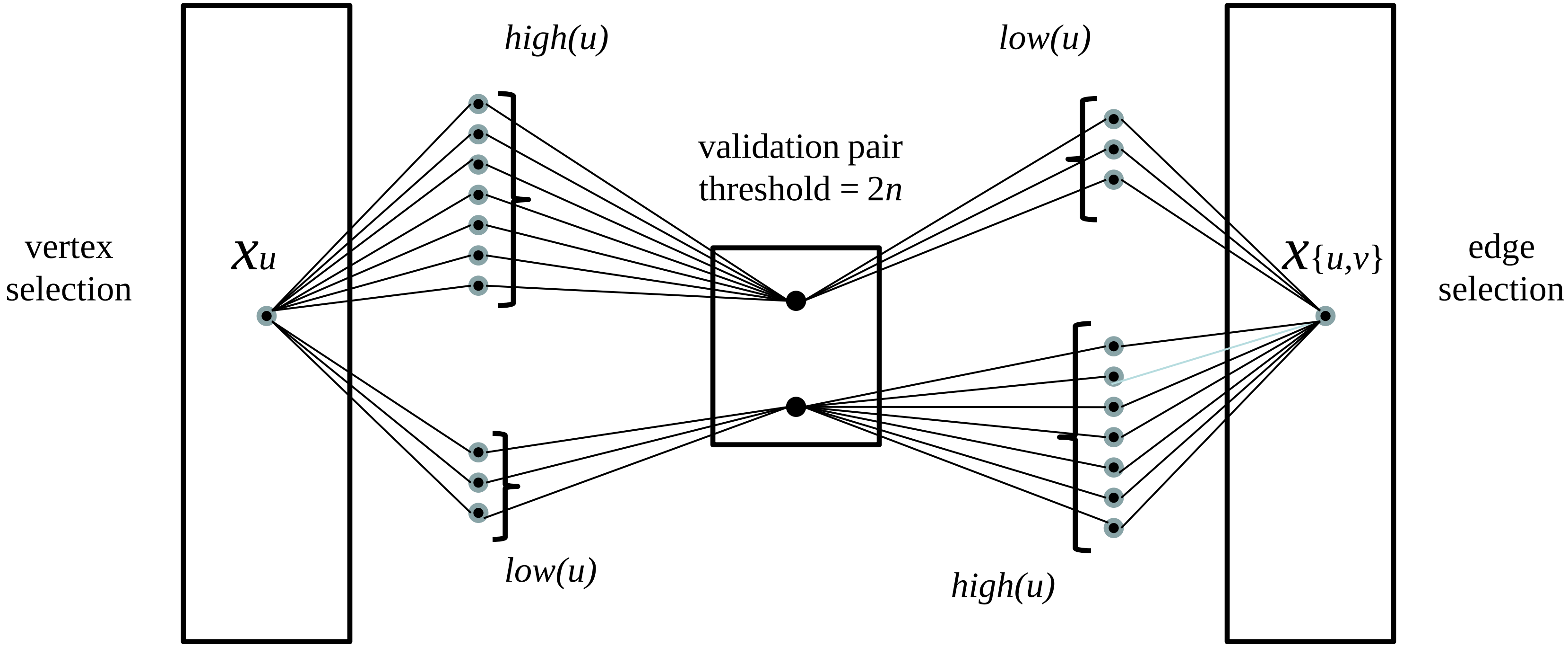
Lower Bounds

- From k-Multicolored Clique to TSS:
  - Assign each vertex unique ids
    - $low(v) \in \{1,\ldots,n\}$ and $high(v) := 2n - low(v)$
  - Create validation gadgets between vertex and edge selections



Lower Bounds

- From k-Multicolored Clique to TSS:
  - H has k-multicolored-clique iff G has $k + k(k-1)/2$ target-set
  - Without validation pairs G is a forest
    - $O(k^2)$ validation pairs $\Rightarrow$ $tw(G) = O(k^2)$




That's all Folks!