Counting the Number of Matchings in Chordal and Chordal Bipartite Graph Classes

Yoshio Okamoto (Tokyo Institute of Technology)
Ryuhei Uehara (Japan Advanced Institute of Science and Technology)
uehara@jaist.ac.jp
Takeaki Uno (National Institute of Informatics)

Counting Problems

Counting analogue of NP

- Background…

  [1979] Valiant introduced the class #P;

  problems counting the number of acceptance paths in a poly-time computations by an NTM.

  [2000] Vadhan showed many #P-complete problems on restricted graph classes;

NP-hard to find

- #Vertex Covers
- #Independent Sets
- #Cliques

Easy to find

- #Perfect Matchings
- #Matchings
- #Maximal Matchings

It's fun to count, <u>because</u>
<u>much different view from P≠NP world!!</u>

Counting Problems

- Background…

  [WG 2005] Okamoto, Uno, and Uehara

  consider counting the number of independent sets in chordal graphs:

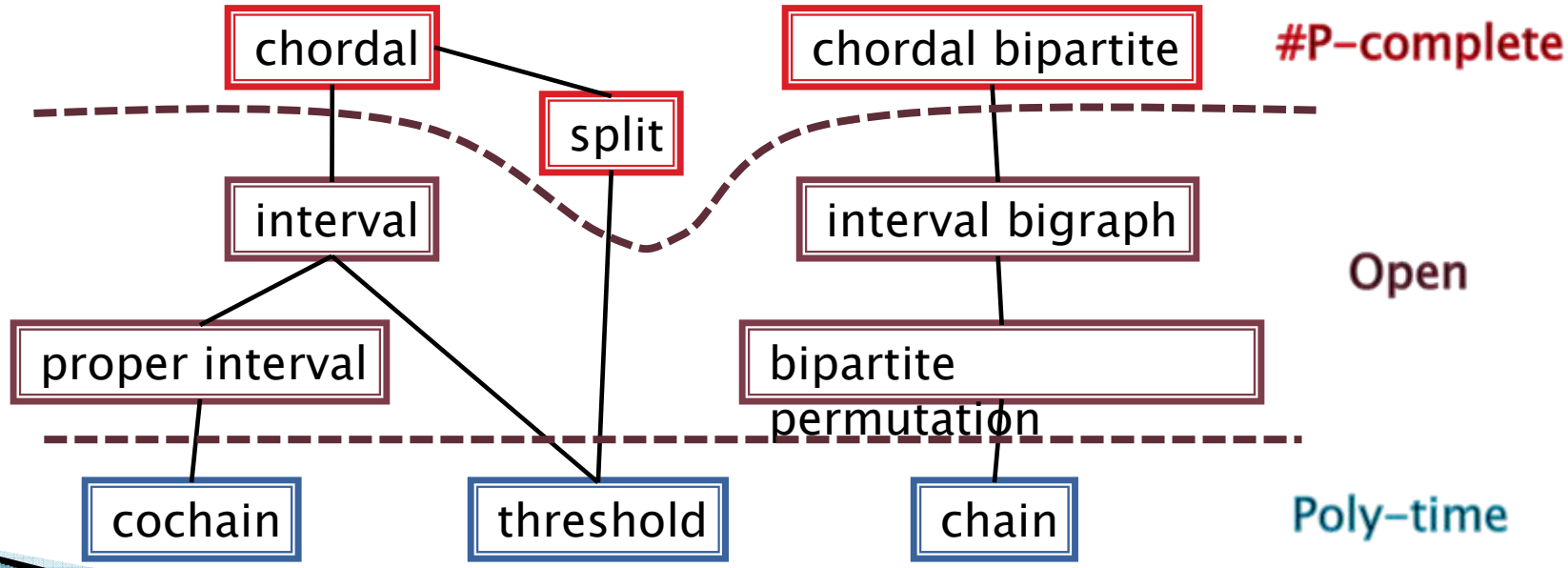
  Poly-time: independent sets,
  maximum independent sets,
  independent sets of size k

  #P-complete:
  maximal independent sets,
  maximum maximal independent sets

Counting Problems

- New Results in this talk

  [WG 2009] Okamoto, Uno, and Uehara consider counting the number of ~~independent sets~~ matchings in chordal graphs and bipartite one.



Counting Problems

- New Results in this talk
  - #matchings in chordal and related graph classes…
    - Polynomial time solvable…
      Classes; threshold, chain, cochain
      Method; Dynamic Programming with recursion

They do not have linear structures.

    - #P-hard…
      Classes; chordal, split, chordal bipartite
      Method; reduction with counting

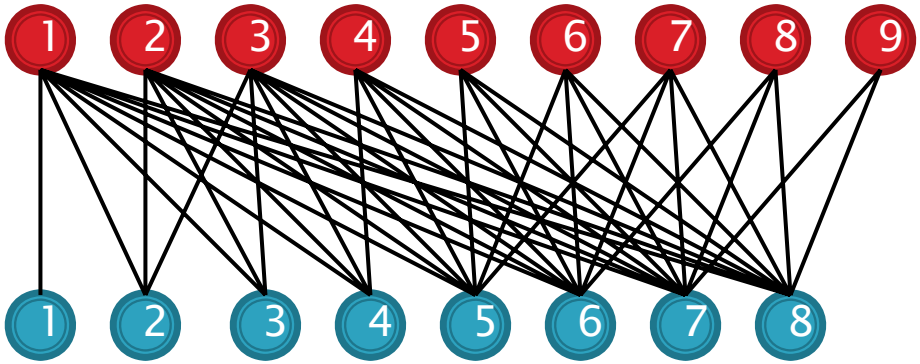
Combination of reductions.

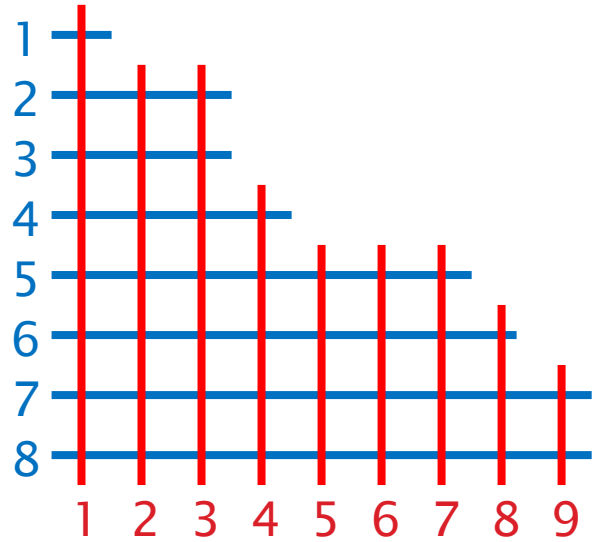
Counting Problems

- New Results in this talk
  - #matchings in chordal and related graph classes…
    - Polynomial time solvable…$O(n^2 \log n)$ time
      Classes; threshold, chain, cochain
      Method; Dynamic Programming with recursion
    - Related results; #matchings can be solved in poly time if the graph has bounded cliquewidth;
      - threshold… $O(n^5)$
      - chain … $O(n^7)$
      - cochain …$O(n^{13})$

Polynomial time algorithms for chain, threshold, and cochain

- Chain graph is a bipartite graph $G=(X, Y, E)$ that has an intersection model of line segments…



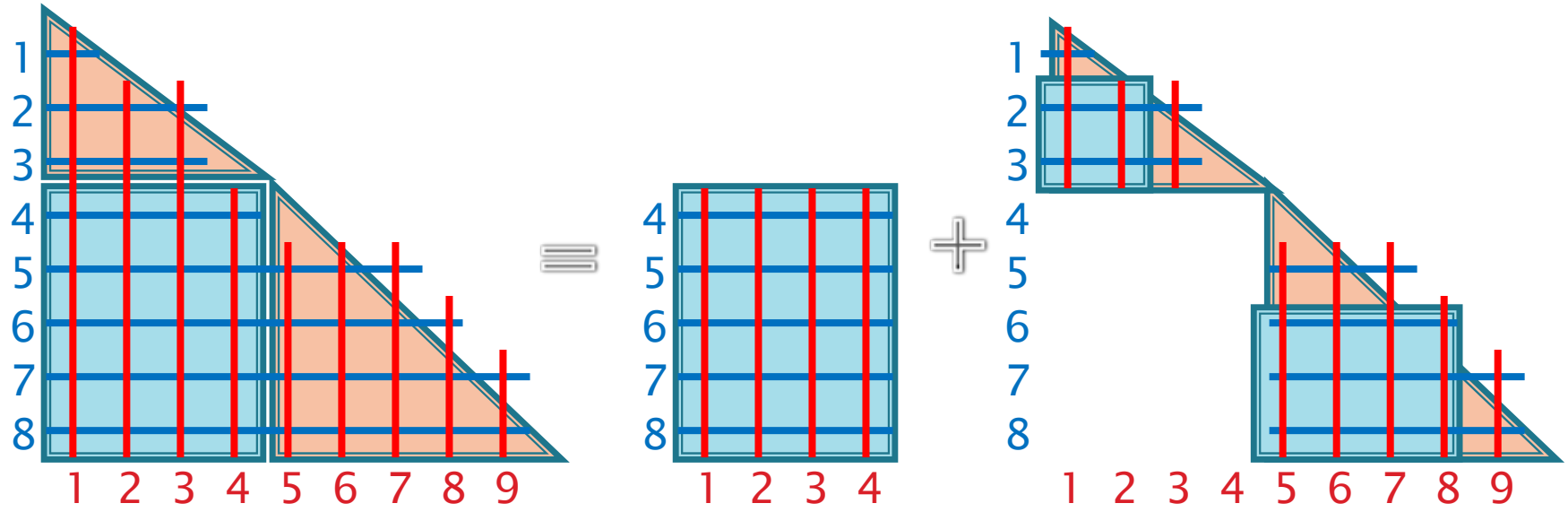


$$N(x_9) \subseteq N(x_8) \subseteq \cdots \subseteq N(x_2) \subseteq N(x_1)$$

$$N(y_1) \subseteq N(y_2) \subseteq \cdots \subseteq N(y_7) \subseteq N(y_8)$$

Polynomial time algorithms for chain, threshold, and cochain

- Chain graph is a bipartite graph $G=(X, Y, E)$ that has a recursive structure for taking a suitable bipartite complete graph $K_{a,b}$



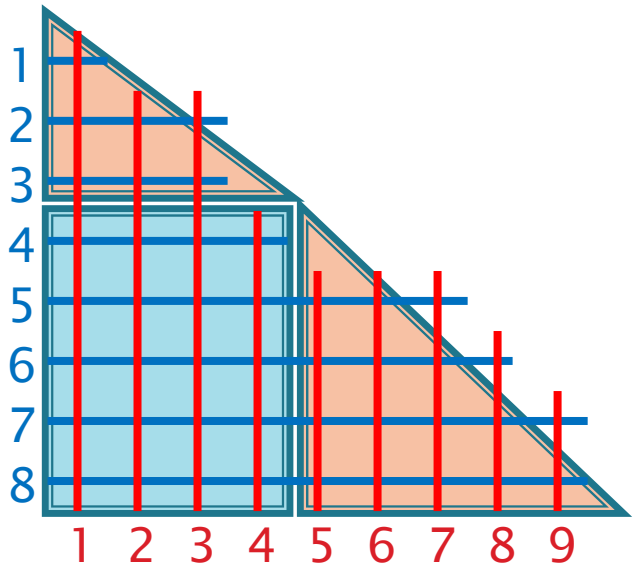
$K_{a,b}$

2 chain graphs

Recursive Decomposition

Polynomial time algorithms for chain, threshold, and cochain

- Poly algorithm for a chain graph:
  1. fix a and b
  2. $\sum_{\text{each } k}$ (#matchings of size k in $K_{a,b}$) × (#matchings in the Upper chain graph missing k vertices from X) × (#matchings in the Right chain graph missing k vertices from Y)



Upper/Right chain graphs are independent, so we can use the DP technique with recursion.

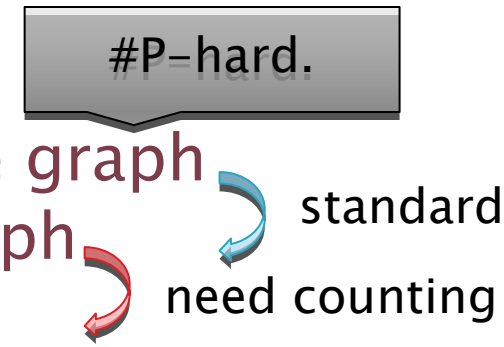
Threshold graphs and cochain graphs similar structure, so we can modify the algorithm to compute for them.

#P-hardness of counting for chordal graphs and related classes

- #P-hardness for chordal, split, chordal bipartite…

  Method; reduction with counting

  - Split graph is a graph $G=(X, Y, E)$ *s.t.*
    - X induces a clique
    - Y induces an independent set
  - Reductions
    - #perfect matchings in a bipartite graph
    - #perfect matchings in a split graph
    - #matchings in a split graph



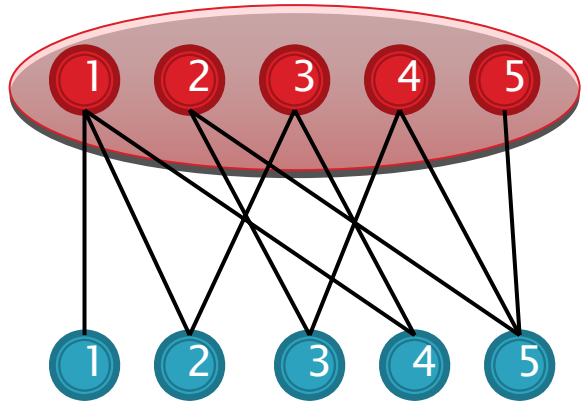
#P-hardness of counting for split graphs

- #P-hardness for a split graph $G=(X, Y, E)$ *s.t.*
  - X induces a clique
  - Y induces an independent set
  - Reduction 1
    - #perfect matchings in a bipartite graph
    - #perfect matchings in a split graph

#P-hard.

standard

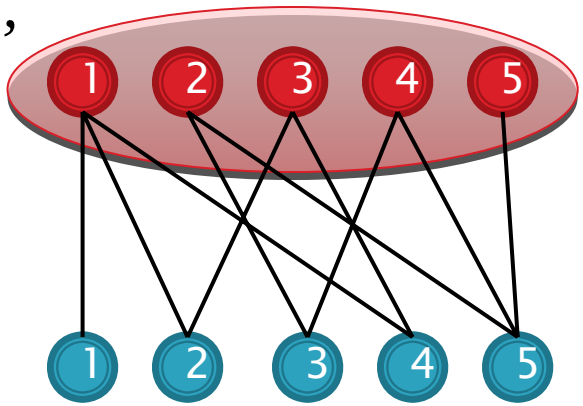
Any given $G=(X, Y, E)$ with $|X|=|Y|$, G' is obtained by adding all edges joining every pair in X.



#P-hardness of counting for split graphs

- #P-hardness for a split graph $G=(X, Y, E)$
  - Reduction 1
    - #perfect matchings in a split graph
    - #matchings in a split graph

Any given $G=(X, Y, E)$ with $|X|=|Y|$, G' is obtained by adding all edges joining every pair in X.



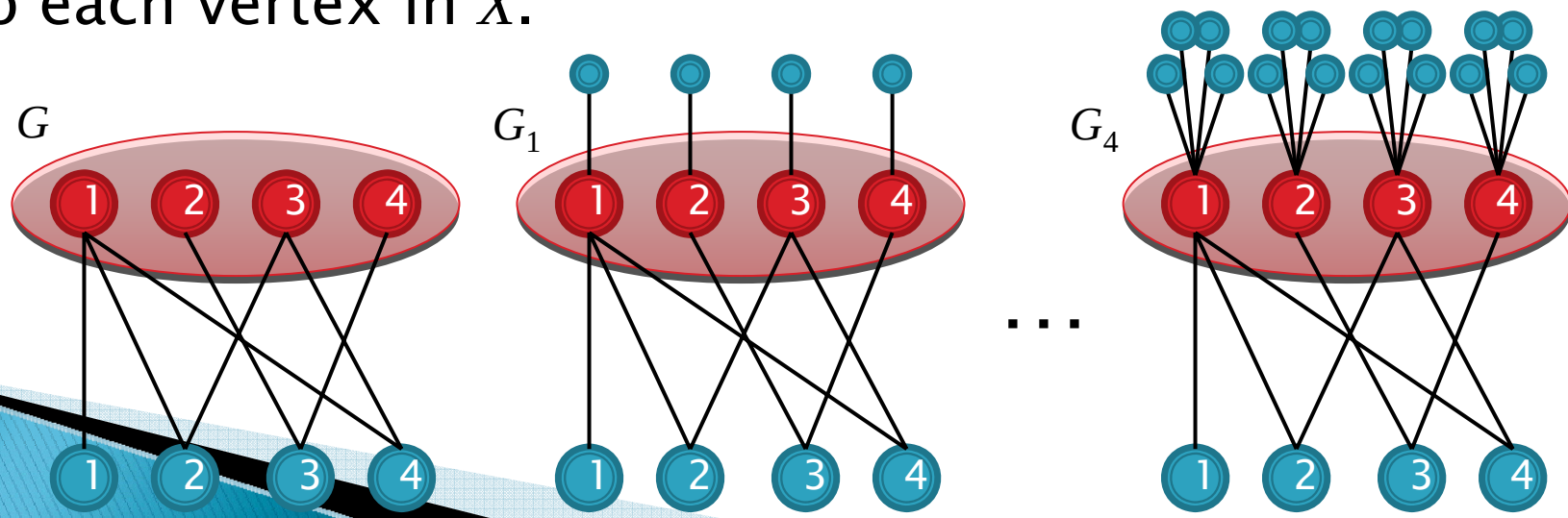
G' is a split graph

#perfect matchings in G = #perfect matchings in G'

#P-hardness of counting for split graphs

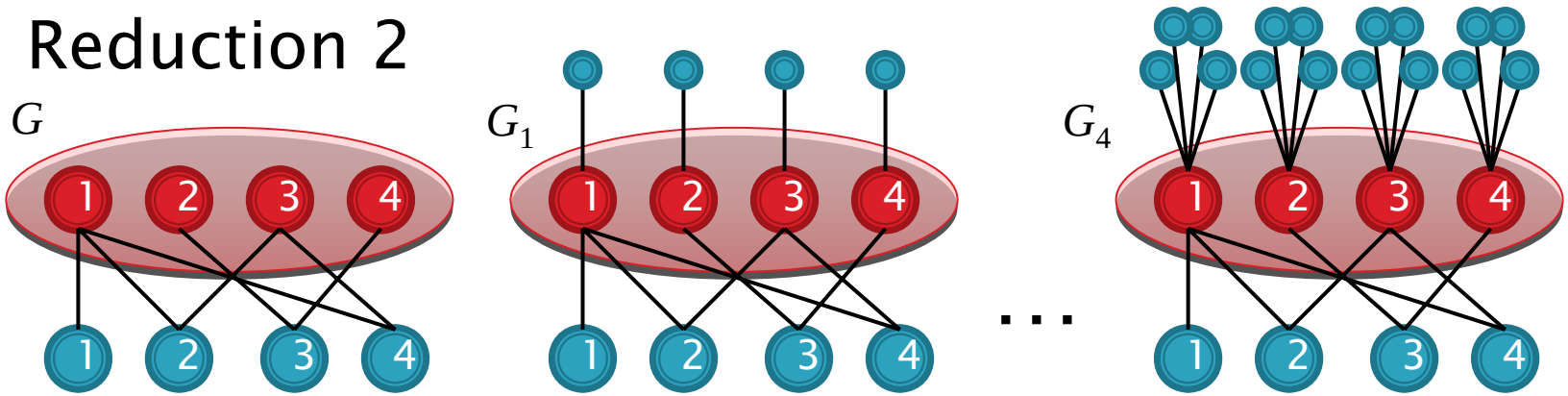
- #P-hardness for a split graph $G=(X, Y, E)$
  - Reduction 2
    - #perfect matchings in a bipartite graph
    - #perfect matchings in a split graph

Any given split graph $G=(X, Y, E)$ s.t. $G[X]$ is clique, G_i is a split graph obtained by adding i pendants to each vertex in X.



#P-hardness of counting for split graphs

- #P-hardness for a split graph $G=(X, Y, E)$
  - Reduction 2



$\mu(G_i) :=$ #matchings in G_i

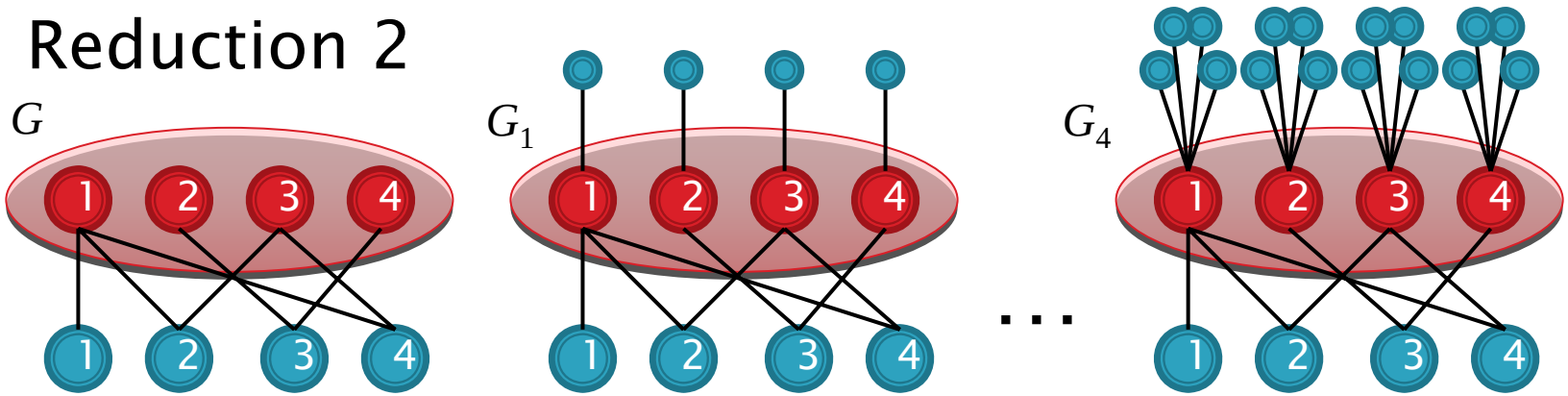
$\mu_i(G) :=$ #matchings of size i in G

$$\begin{pmatrix} \mu(G_1) \\ \mu(G_2) \\ \vdots \\ \mu(G_{n+1}) \end{pmatrix} = A \begin{pmatrix} \mu_0(G) \\ \mu_1(G) \\ \vdots \\ \mu_n(G) \end{pmatrix}$$

- A is constructible
- A has its inverse

#P-hardness of counting for split graphs

- #P-hardness for a split graph $G=(X, Y, E)$
  - Reduction 2



$\mu(G_i)$:= #matchings in G_i

$\mu_i(G)$:= #matchings of size i in G

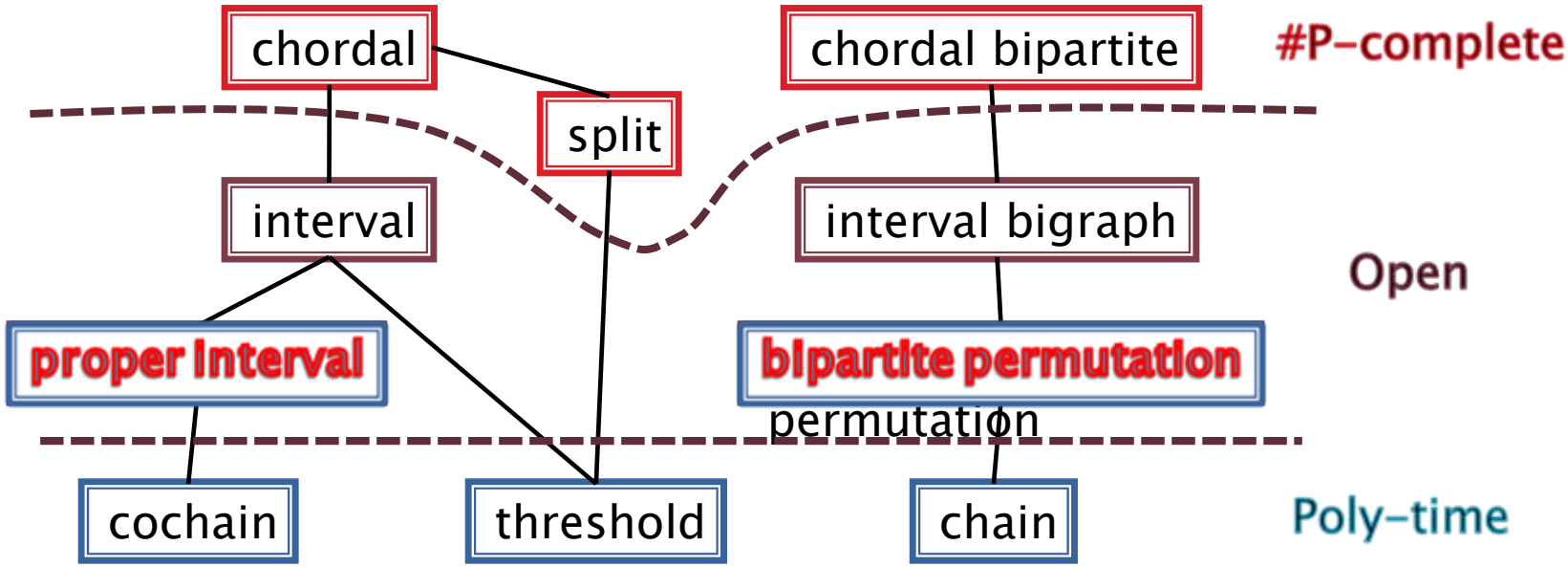
$$A^{-1}\begin{pmatrix}\mu(G_1)\\ \mu(G_2)\\ \vdots\\ \mu(G_{n+1})\end{pmatrix}=\begin{pmatrix}\mu_0(G)\\ \mu_1(G)\\ \vdots\\ \mu_n(G)\end{pmatrix}$$

$\mu_n(G)$ is computable from $\mu(G_i)$s.

$\mu_n(G)$ ← #perfect matchings

Concluding Remarks

- New Results in this talk
  - The number of matchings in chordal graphs and related classes

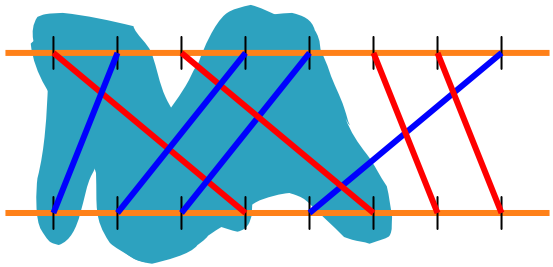


Deadline: July, 24

- Bipartite permutation graph is…

  a intersection graph of two sets of line segments

  decomposed into a sequence of chain graphs



combination of

- recursive decomposition of chain graphs
- linear decomposition of bipartite permutation graph

… yields exponential algorithm ;-)

⇒fragments of chain graphs have relation to the neighbor chain graphs…

Bipartite permutation — Open

Chain — Poly-time