

Synchronous Hyperedge Replacement Graph Grammars

Corey Pennycuff, Satyaki Sikdar, Catalina Vajiac,
David Chiang, and Tim Weninger

*Department of Computer Science and Engineering
University of Notre Dame*

Context Free String Generation

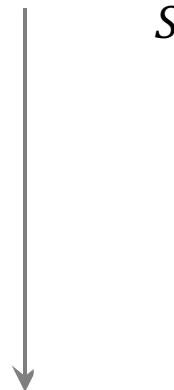


Context Free Grammars

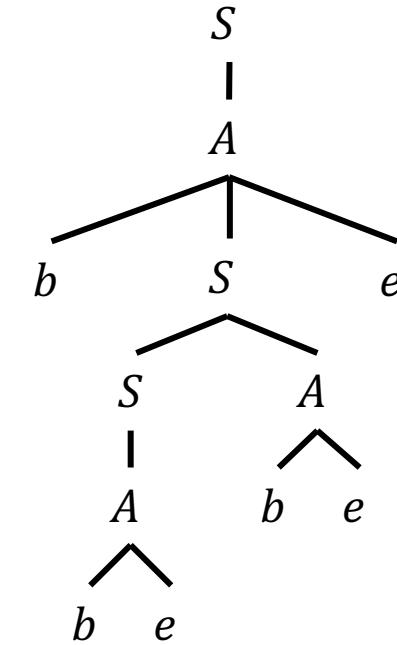
Production Rules

- R1: $S \rightarrow SA$
- R2: $S \rightarrow A$
- R3: $A \rightarrow bSe$
- R4: $A \rightarrow be$

Derivation of “bbebee”



Parse Tree



Context Free Grammars

Where did the Production Rules come from? Can we learn them?

Production Rules

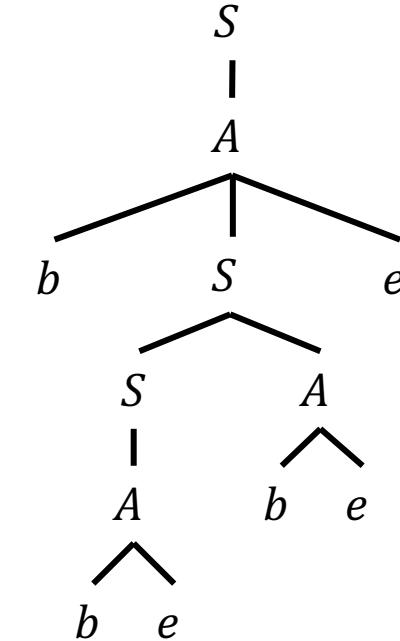
- R1: $S \rightarrow SA$
- R2: $S \rightarrow A$
- R3: $A \rightarrow bSe$
- R4: $A \rightarrow be$

Learning production rules from *bbebee*

- S
- A
- bSe
- $bSAe$
- $bAAe$
- $bbeAe$
- $bbebee$

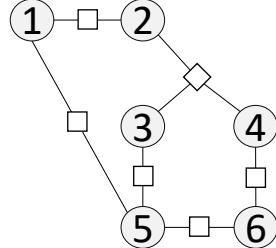
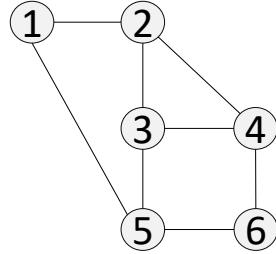
NLP

Parse Tree



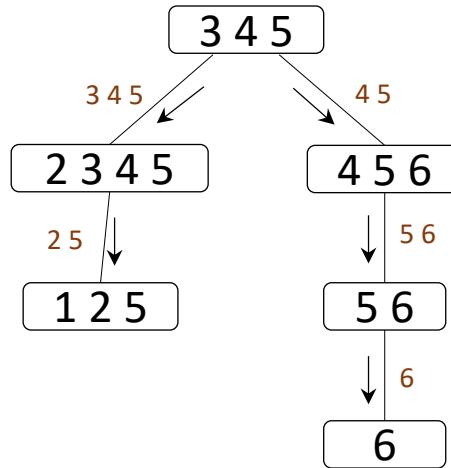
Learning Hyperedge Replacement Grammars

Graph



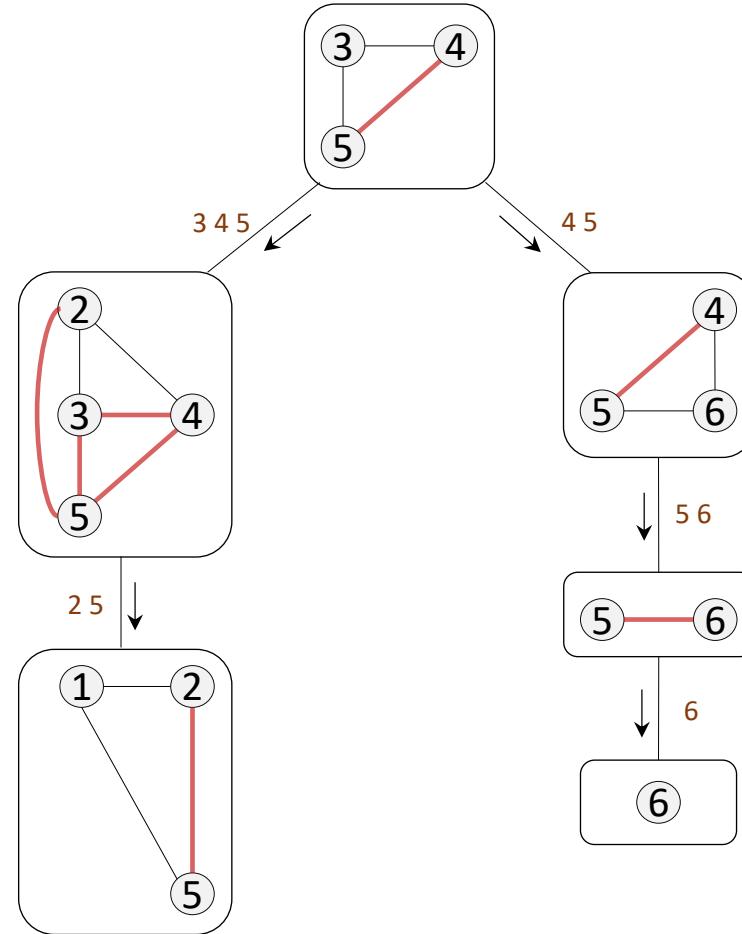
Tree Decomposition

used for many things:
Exact inference in probabilistic graphical models
Viterbi Algorithm runs on a tree decomposition

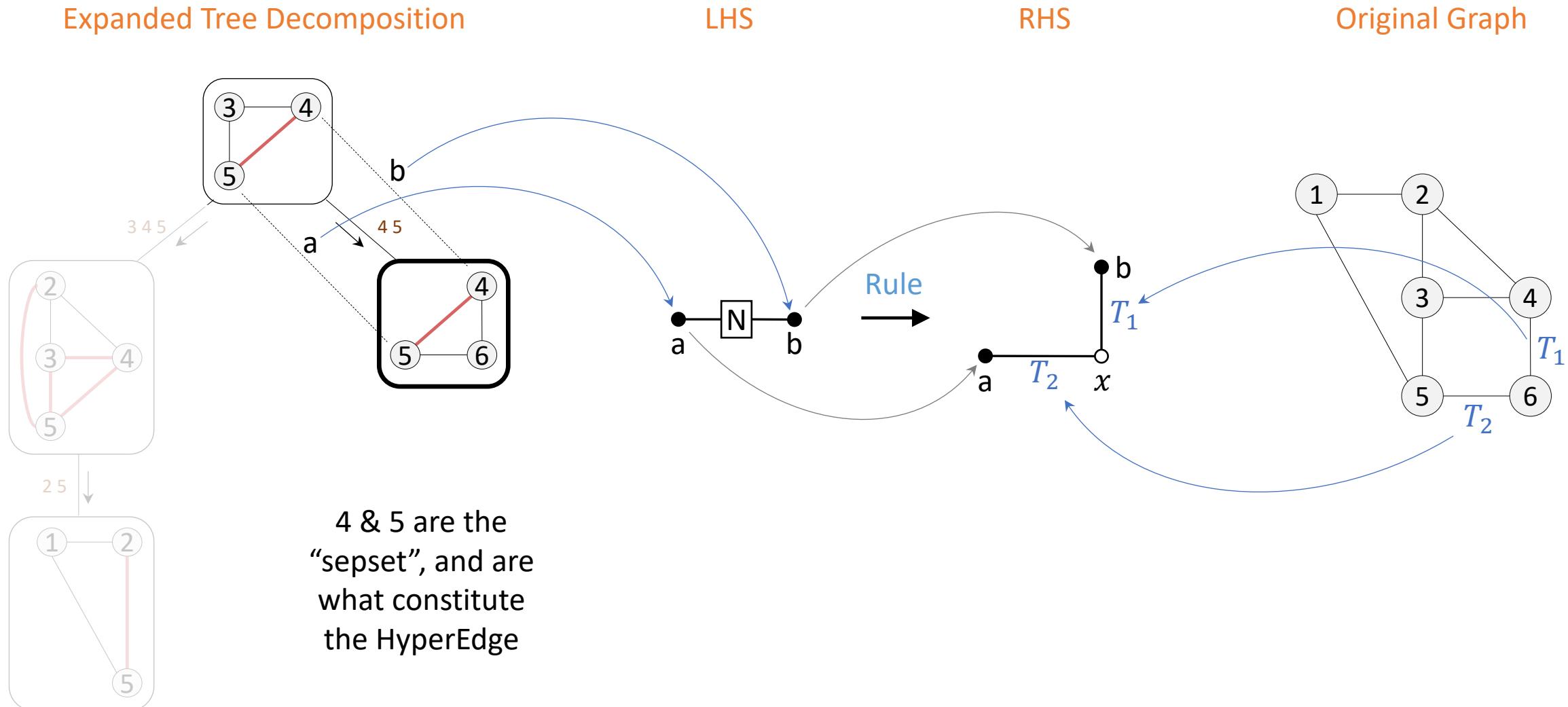


created via elimination ordering
Maximal Cardinality Search (MCS) Heuristic

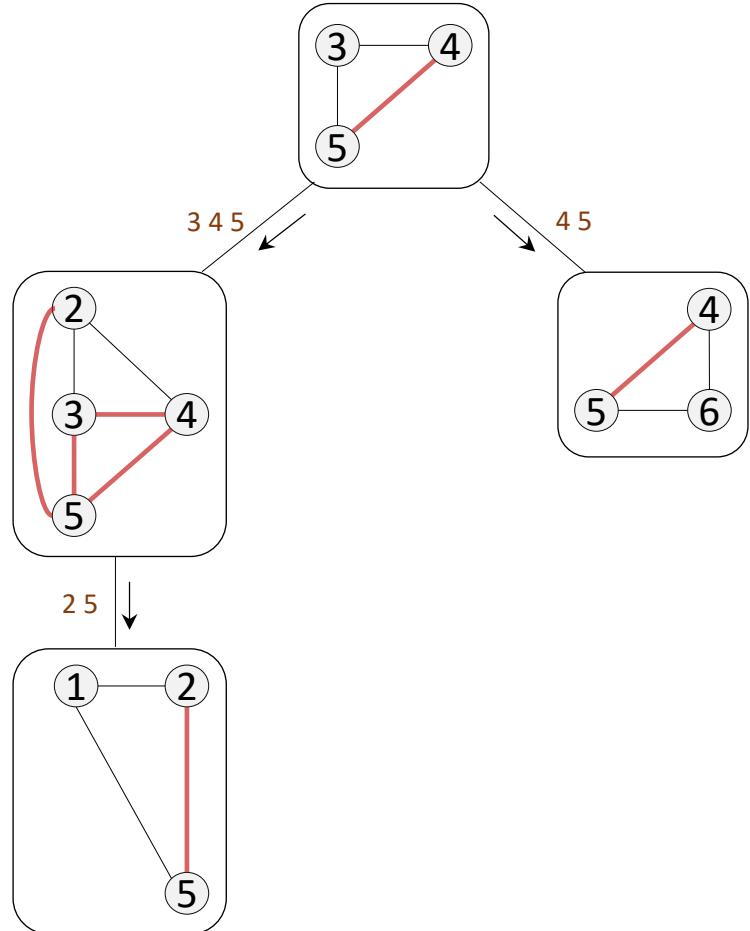
Expanded Tree Decomposition



Learning Hyperedge Replacement Grammars

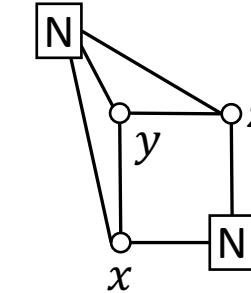


The HRG

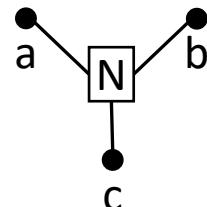
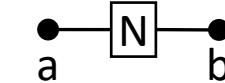


S

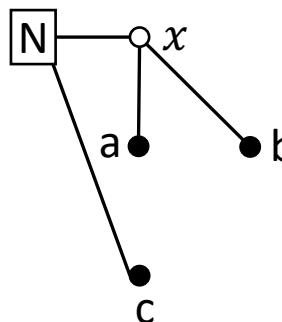
Rule 1



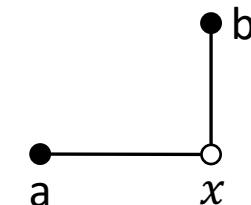
Rule 3
x2



Rule 2



Rule 4



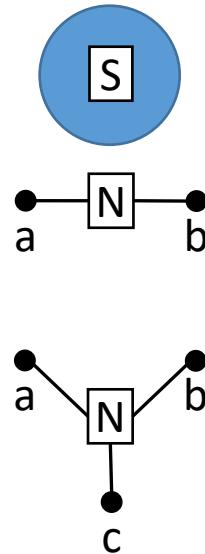
Growing a Graph

Always start with S

Current Graph



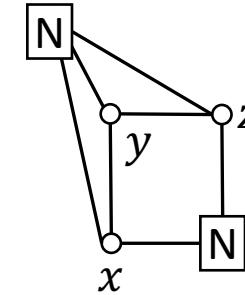
Match LHS



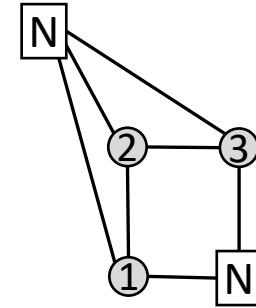
S

Pick Rule

Rule 1 →

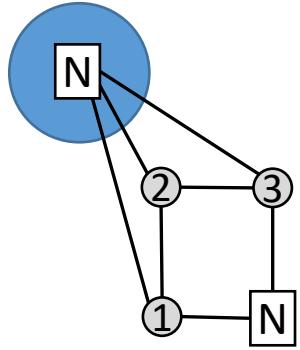


New Graph

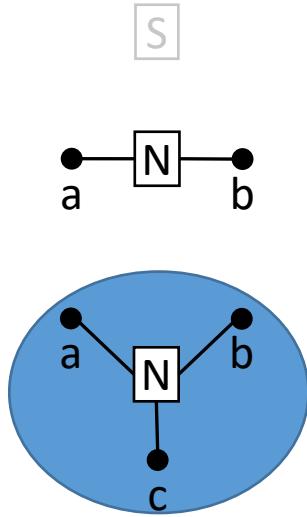


Growing a Graph

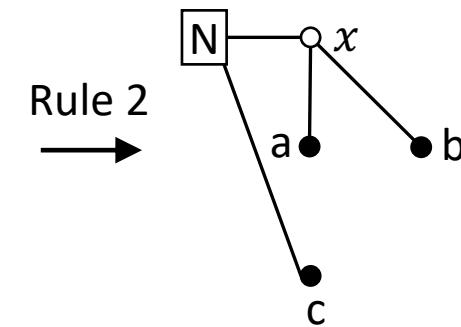
Current Graph



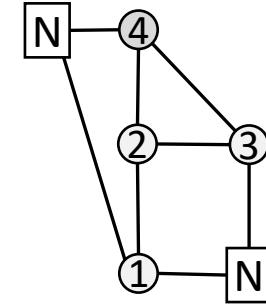
Match LHS



Pick Rule



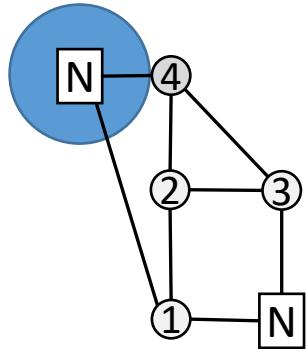
New Graph



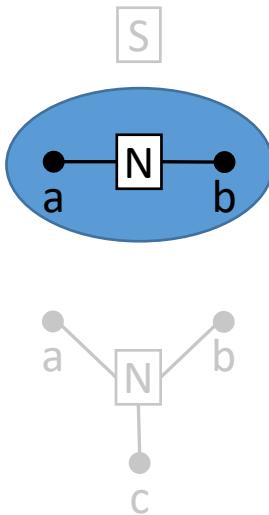
Growing a Graph



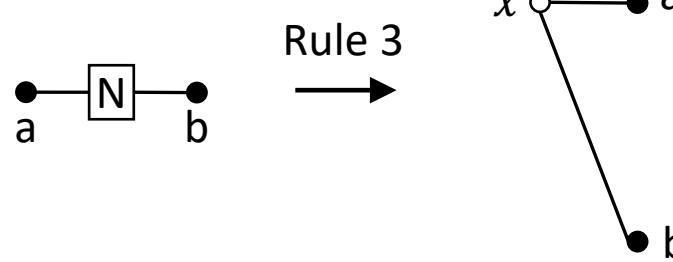
Current Graph



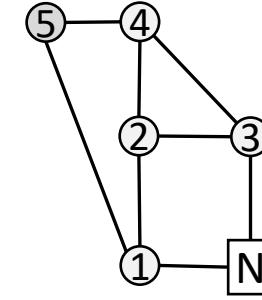
Match LHS



Pick Rule



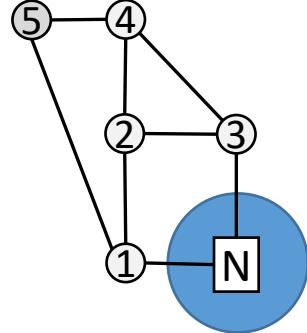
New Graph



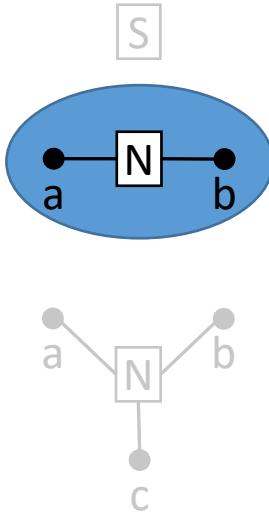
Growing a Graph



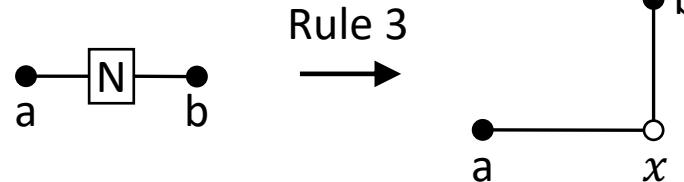
Current Graph



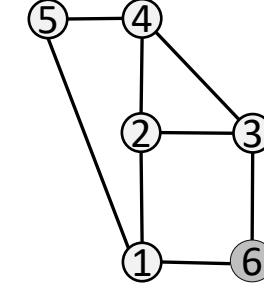
Match LHS



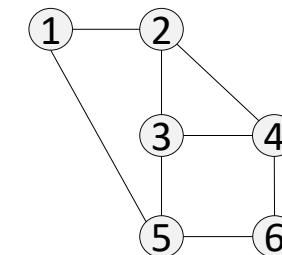
Pick Rule



New Graph



Original Graph



What's Missing



HRG is not learned from the evolution of the actual graph
(Neither are most other graph generators)

- Tree Decomposition of the static, global graph is unnatural and clumsy
- Rules don't *mean* anything

A panoramic view of Segovia, Spain, featuring the ancient Roman aqueduct in the foreground. The aqueduct's iconic stone arches and columns are visible against a backdrop of a city skyline with numerous buildings and a range of mountains in the distance under a cloudy sky.

Synchronous HRGs

Synchronous CFGs

Given 2 equivalent sentences in different languages:

- English: I open the box.
- Japanese: Watashi ha hako wo akemasu.
- Synchronous grammars map the **syntactic structure** and **vocabulary** for each language, and **pairs** them into a single rule.
- A sentence **decomposed** with one grammar can be **reconstituted** using the corresponding rules from the other language, and is thereby **translated**.
- How do we apply these to graphs?

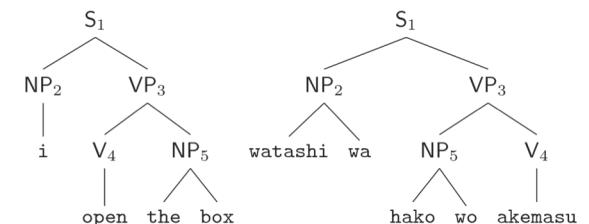
Production Rules

($LHS \rightarrow RHS_{English} : RHS_{Japanese}$)

$S \rightarrow NP_1 VP_2 : NP_1 VP_2$
 $VP \rightarrow V_1 NP_2 : NP_2 V_1$
 $NP \rightarrow i : watashi ha$
 $NP \rightarrow \text{the box} : hako wo$
 $V \rightarrow \text{open} : akemasu$

Applying Rules Synchronously

Rule	English	Japanese
1	S_1	S_1
2	$NP_2 VP_3$	$NP_2 VP_3$
3	$NP_2 V_4 NP_5$	$NP_2 NP_5 V_4$
4	$i V_4 NP_5$	$watashi ha NP_5 V_4$
5	$i V_4 \text{the box}$	$watashi ha hako wo V_4$
	$i \text{ open the box}$	$watashi ha hako wo akemasu$

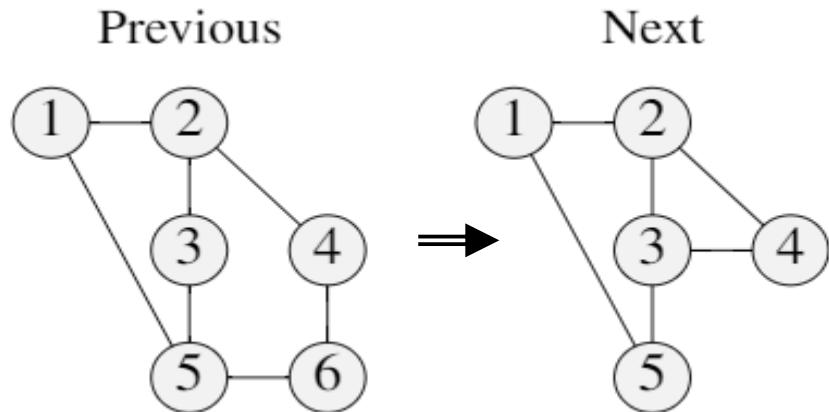


Example from Chiang, D.: An Introduction To Synchronous Grammars

Applying SCFGs to Graphs (Intuition)

- How can we “Translate” a graph?
 - We translate from one timestep to the next
- English : Japanese :: $H^{(t)} : H^{(t+1)}$

I open the box. \Rightarrow Watashi ha hako wo akemasu.

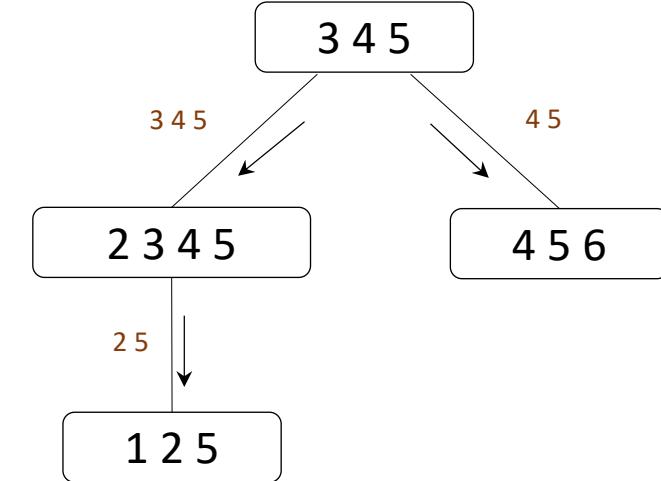
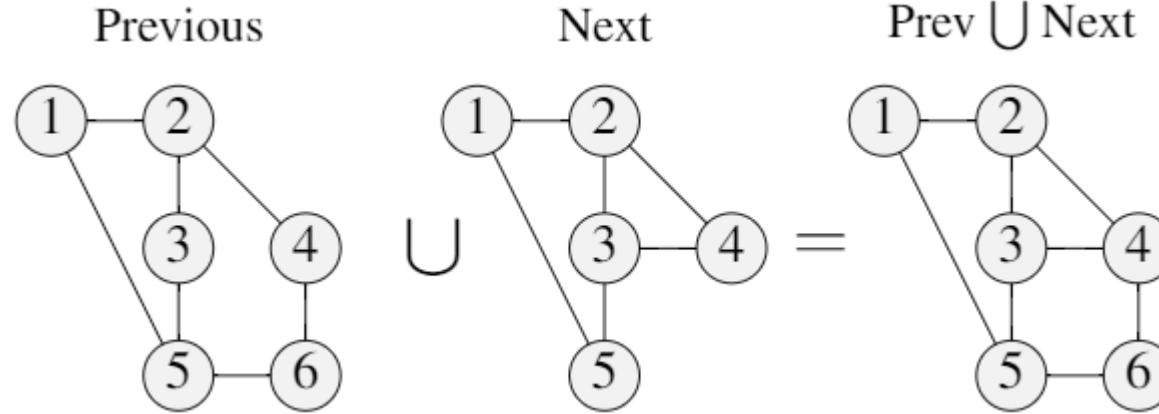


Tree Decomposition from the Union

How can two distinct, temporal snapshots of a graph be “equivalent”?

Work from a union of the two graphs

- Union of Graphs

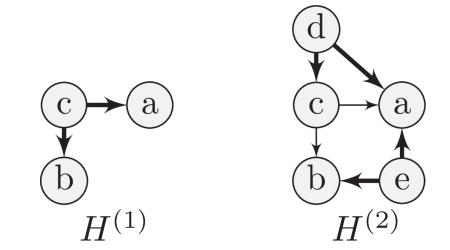


Generate synchronous rules from the context of the individual timestamps

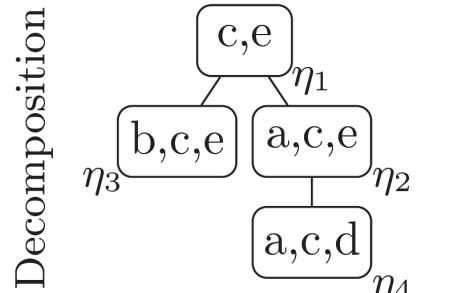
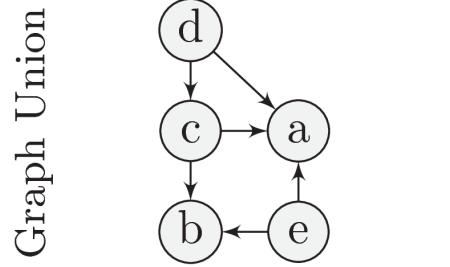
Generated rules must have identical size and number of non-terminals in each paired rule.

Differences in a synchronous rule can model addition or removal of edges.

Extracting a Synchronous Hyperedge Replacement Grammar

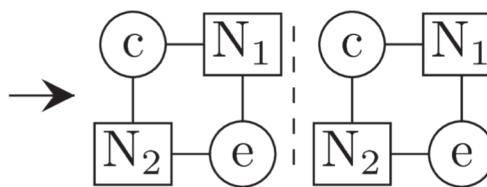


$H^{(1)} \cup H^{(2)}$



LHS

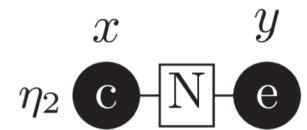
η_1 S



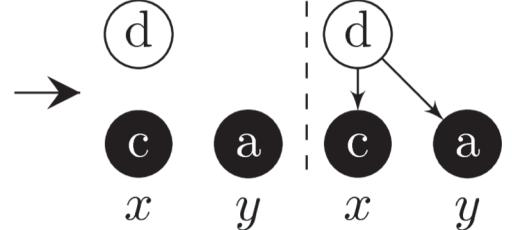
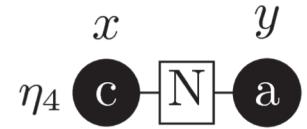
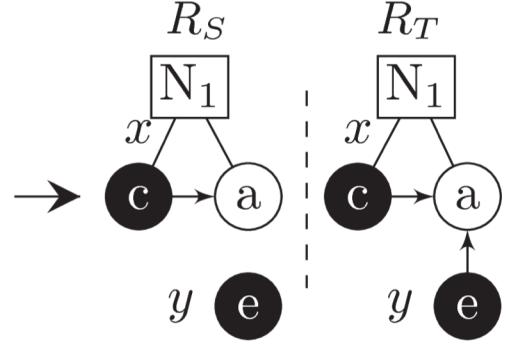
RHS

R_S R_T

LHS



RHS



Graph Union

Tree Decomposition

Is the SHRG Meaningful?

Generate 1000 of each Graph Type

- BA with $n = 10$ and $m = 2$, over $n - m$ timesteps
- ER: create n vertices & add edges between two vertices with probability p
 - ER_1 : $n(n - 1)$ timesteps, with $n(1 - p)$ timesteps expected to contain no changes
 - ER_2 : same as ER_1 , but skips timesteps where no changes occur
 - ER_3 : create 2 directed edges per timestep
 - $pn(n - 1)$ edges over $pn(n - 1)/2$ timesteps

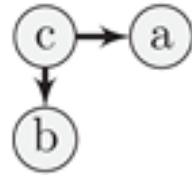
Inspect rules to see if consistent with BA growth process

- Rule 4 – persistence of wedges
- Rule 8 – creation of wedges
- Rule 9 – Preferential attachment
- Rules 3 & 12 – Impossible for BA

	R_S	R_T	BA	ER_1	ER_2	ER_3
Static Rules	1	• : •	0.051	0.157	0.171	0.153
	2	• → • : • → •	0.133	0.545	0.542	0.412
	3	• → • : • → •	0.000	0.006	0.017	0.007
	4	• → • : • → •	0.260	0.047	0.034	0.029
	5	• → • : • → •	0.032	0.028	0.030	0.017
	6	• → • : • → •	0.025	0.009	0.000	0.002
BA Rules	7	• : • → •	0.243	0.164	0.166	0.325
	8	• : • → •	0.097	0.000	0.000	0.001
	9	• → • : • → •	0.155	0.012	0.004	0.014
Other Rules	10	• → • : • → •	0.001	0.000	0.000	0.002
	11	• → • : • → •	0.000	0.021	0.031	0.028
	12	• → • : • → •	0.000	0.006	0.006	0.008
	13	• → • : • → •	0.000	0.006	0.005	0.007

Can we Predict Future Graph Changes?

Current Graph



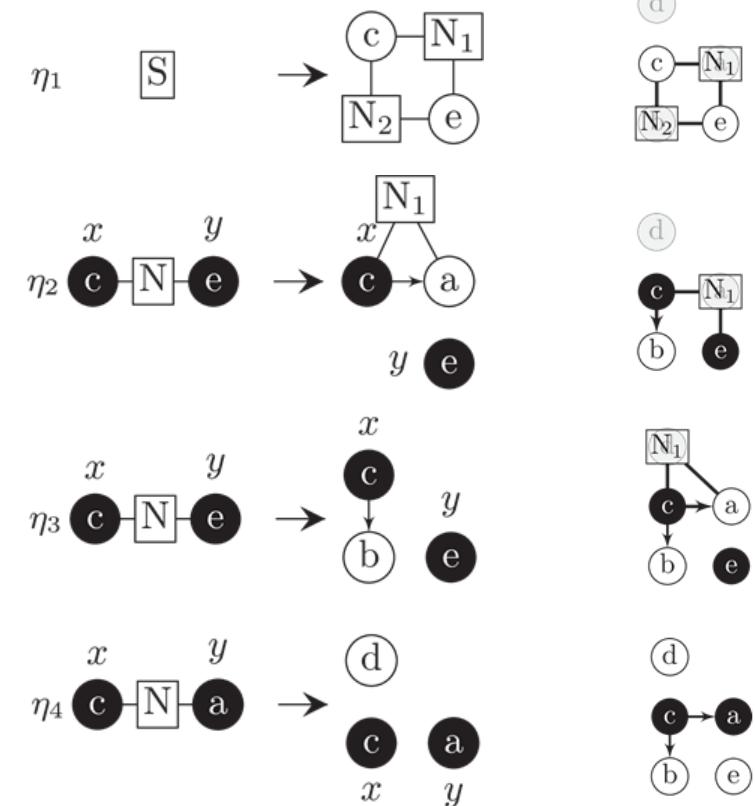
Parse Current Graph using R_S

Algorithm adapted from hypergraph parsing algorithm (reverse-CYK algorithm) by Chiang et al (2013)

Algorithm produces a rule ordering (π) which can be used to generate the Source graph

$$\pi = R_1, R_2, R_3, R_4$$

L R_S



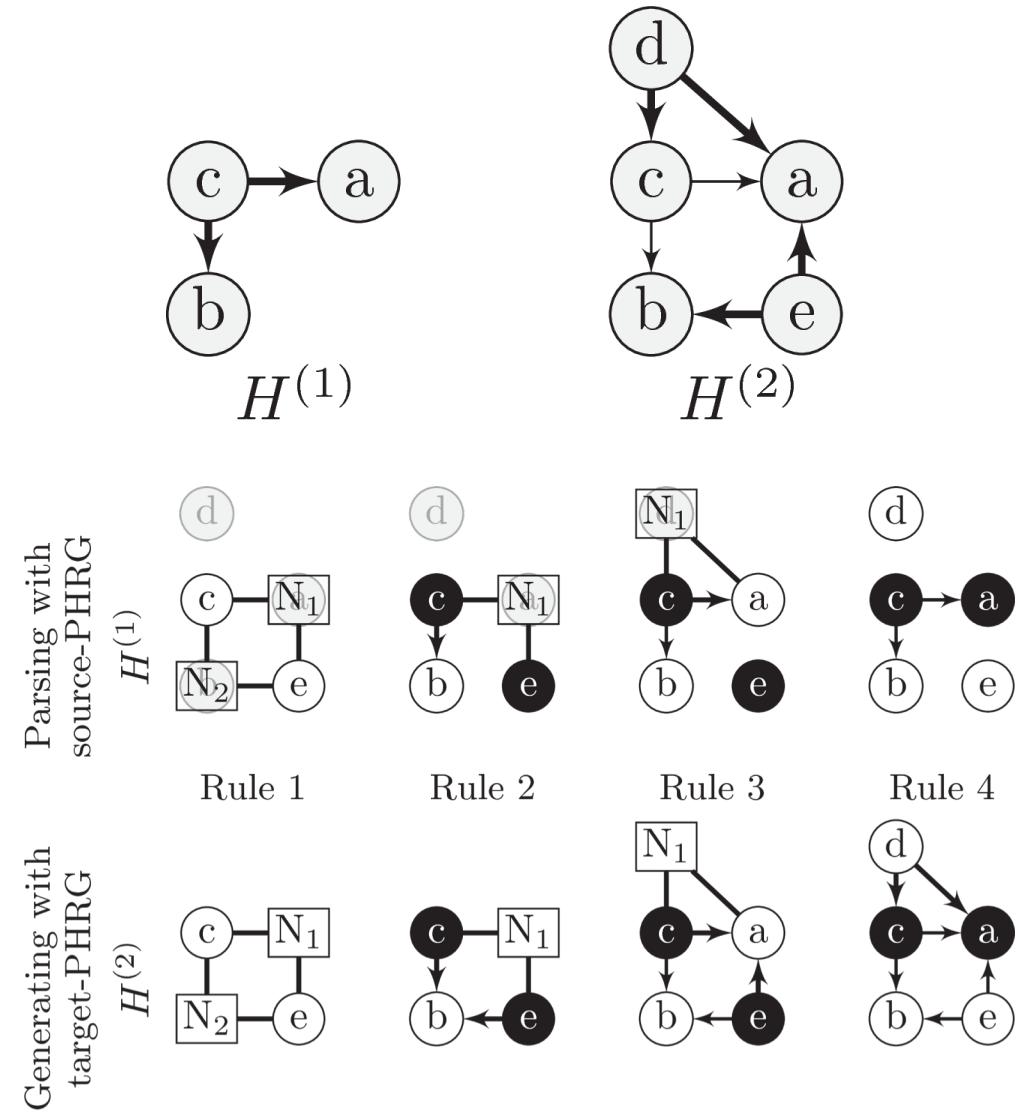
Generating $t + 1$

Generate $H^{(t+1)}$ using R_T

- Use π as the rule ordering, applying RHS rules R_T

Limitations

- Cannot predict unseen structures
- Multiple (valid & optimal) π
- Hypergraph parsing limits graph size (computationally)



Experiments



Methodology

Given a dynamic graph H with n timesteps, extract PSHRG grammar from $H^{(1)} \dots H^{(n-1)}$.

- Extract π using Chiang algorithm and R_S on $H^{(n-1)}$
- Execute π ordering with R_T , creating H^*
- Compare H^* to $H^{(n)}$

Graphs are small

- 5 – 12 nodes
- Limitation of parsing tools

Cramér–von Mises Statistic

- For comparing distributions

Repeat 50 times and plot the mean

Comparisons

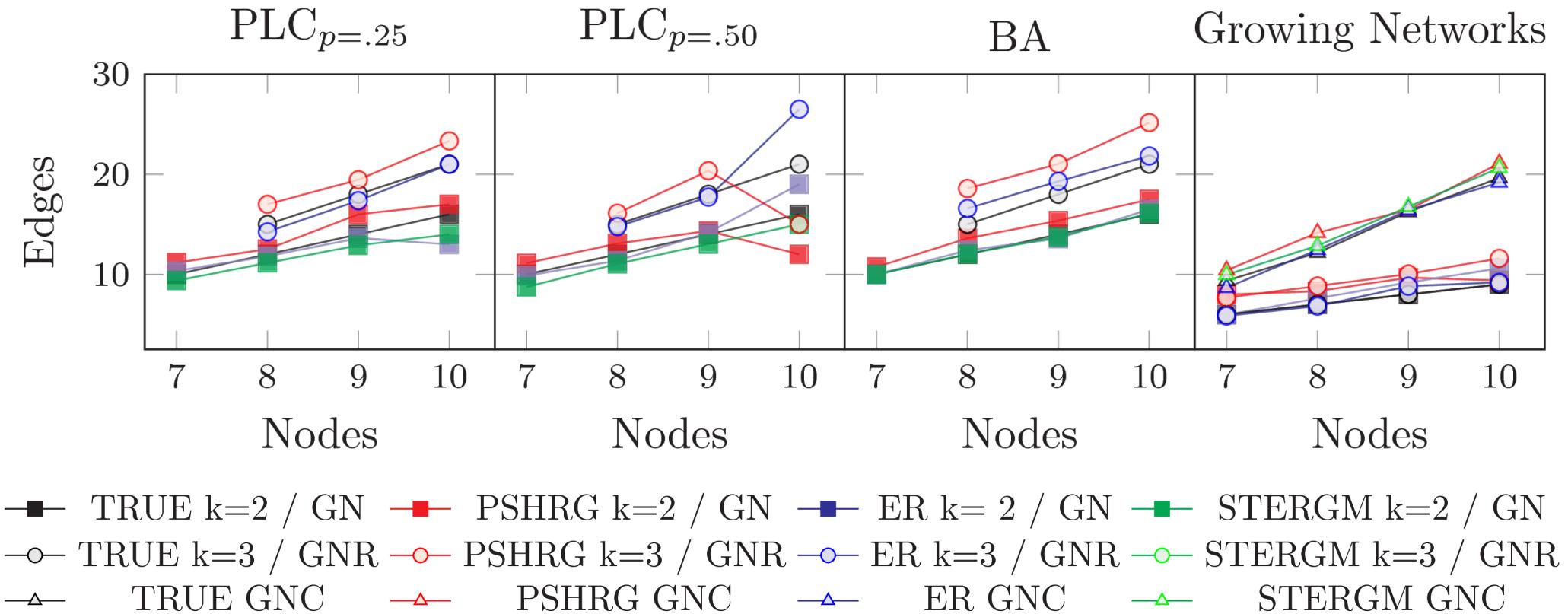
- In-degree
- Out-degree
- PageRank
- Graphlet Correlation Distance (GCD)

Other Graph Generators

- Barabási–Albert (BA) model
 - $k = 2$ and $k = 3$
- Powerlaw-Cluster graph (PLC)
 - $p = 0.25$ and $p = 0.5$
- “Growing networks”
 - GN, GNR, and GNC, $p = 0.5$
- Erdős–Rényi
- Separable Temporal Exponential Random Graph Model (STERGM)

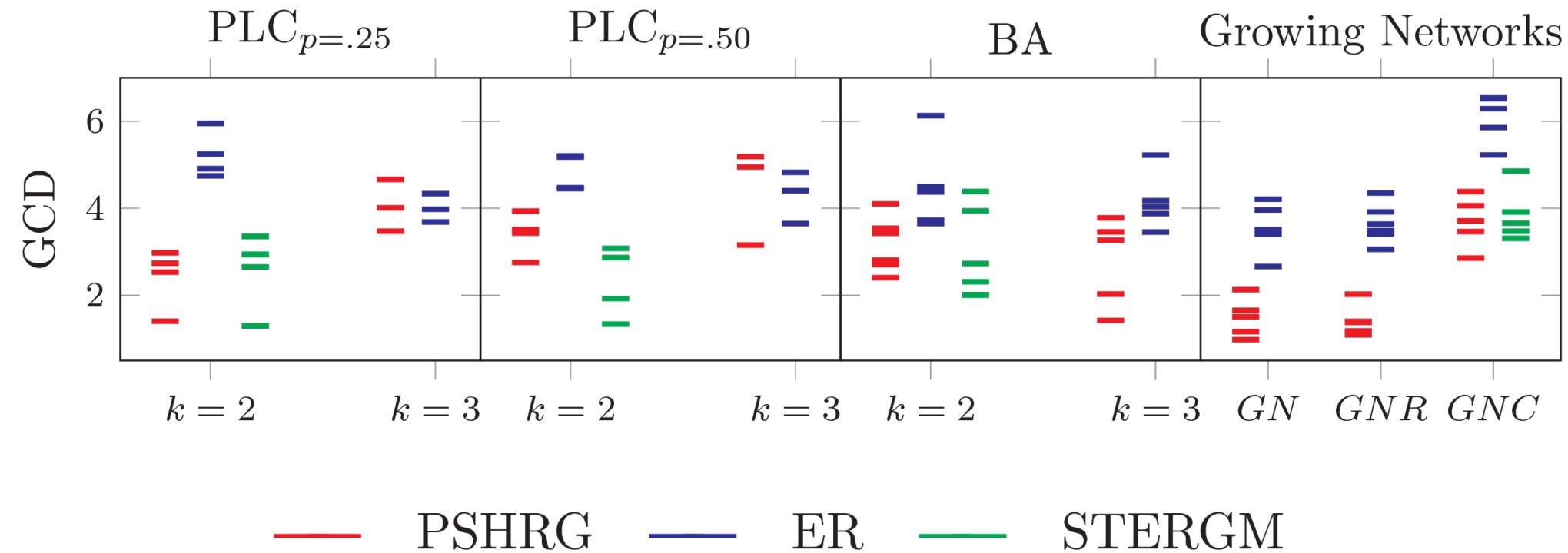
Results: Edge Count

Number of edges generated by ER, STERGM, and PSHRG graph generators for each of the PLC, BA, and Growing Networks (GN, GNR, GNC) graph processes.



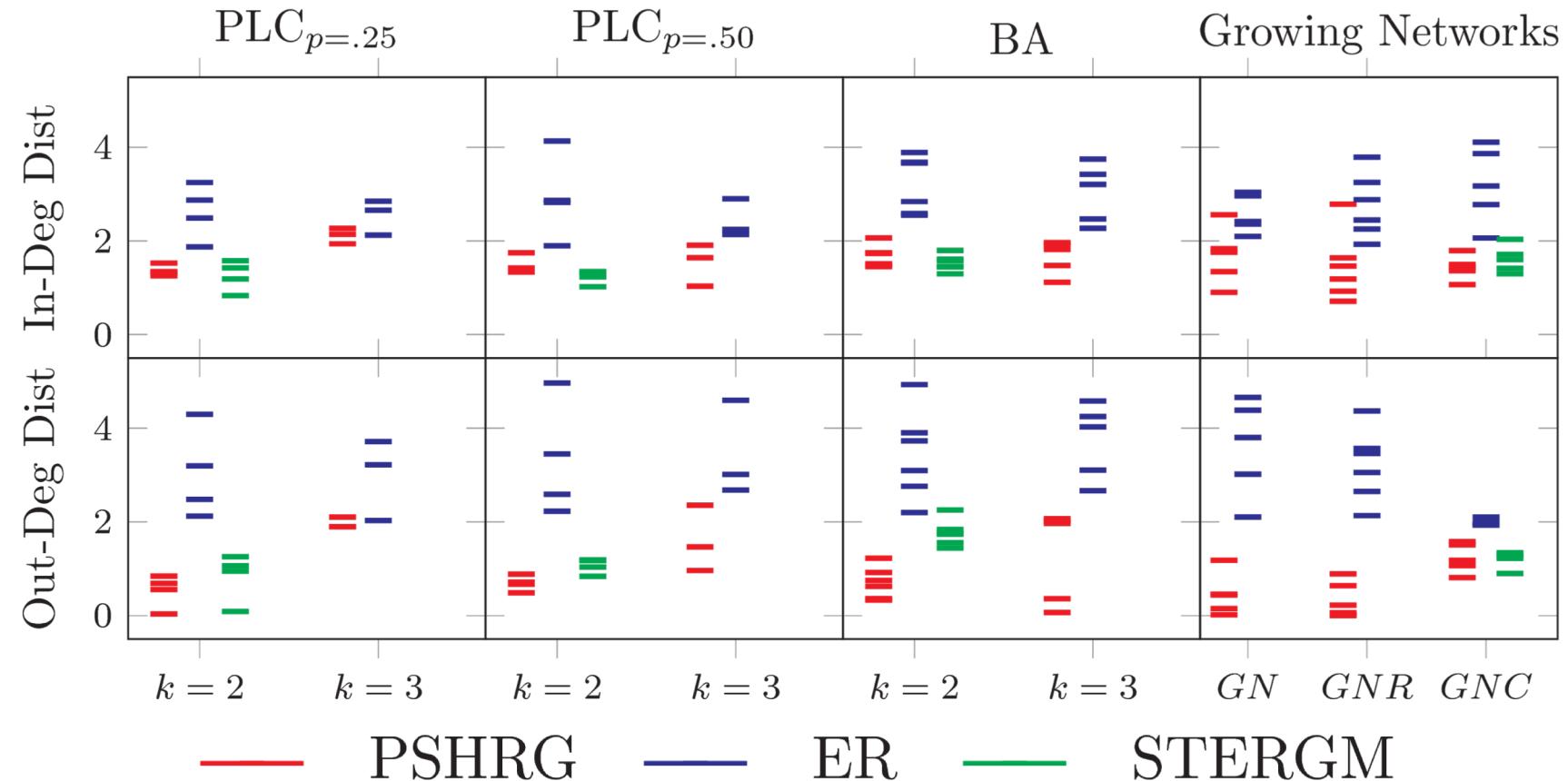
Results: Graphlet Correlation Distance

Graphlet Correlation Distance (GCD). Dashes represent mean GCD scores for various graph sizes (bottom-to-top almost always represents smaller-to-larger graphs), parameters, and models. Lower is better. PSHRG is usually the best.



Results: Degree Distribution

CVM-test statistics of in-degree (top) and out-degree (bottom) distributions for various graph sizes (bottom-to-top almost always represents smaller-to-larger graphs), parameters, and models. Lower is better. PSHRG and STERGM (when available) results are competitive.



Questions?

