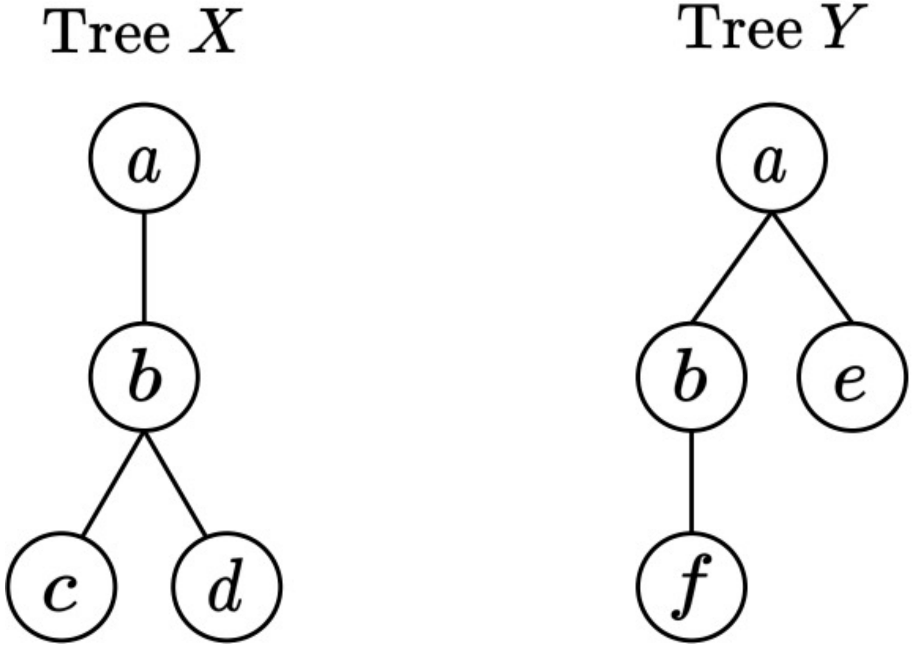
X-TED: Massive Parallelization of Tree Edit Distance

Dayi Fan†**,** Rubao Lee*, Xiaodong Zhang†

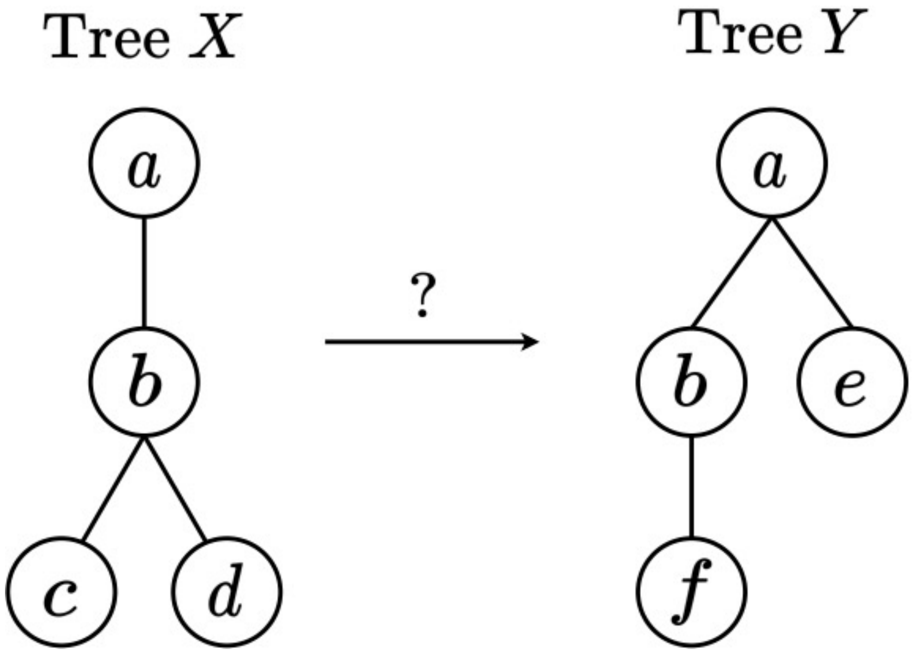
† The Ohio State University

* Freelance

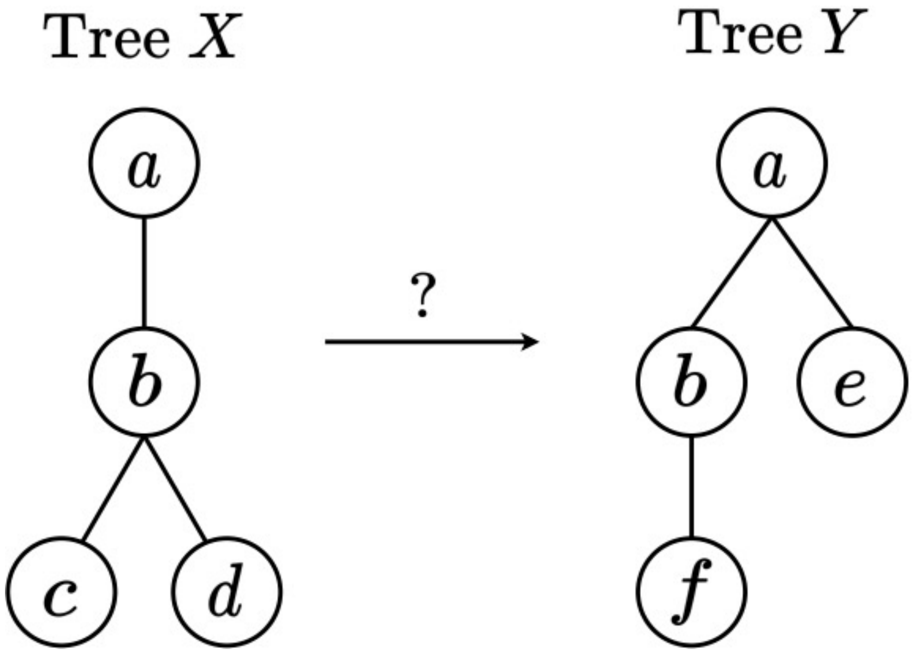
What is Tree Edit Distance (TED)?



What is Tree Edit Distance (TED)?



What is Tree Edit Distance (TED)?



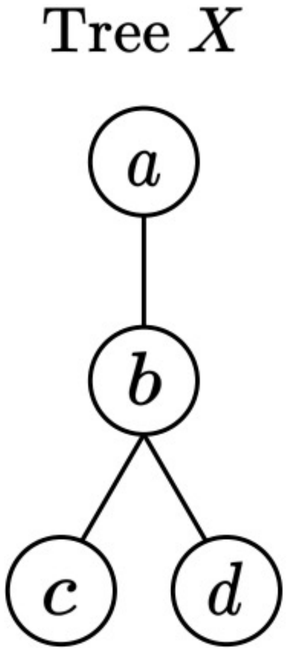
- The **minimum cost** of transforming one tree into another by *edit operations*

Tree Edit Distance (TED)

- Three edit operation types: delete, insert, rename

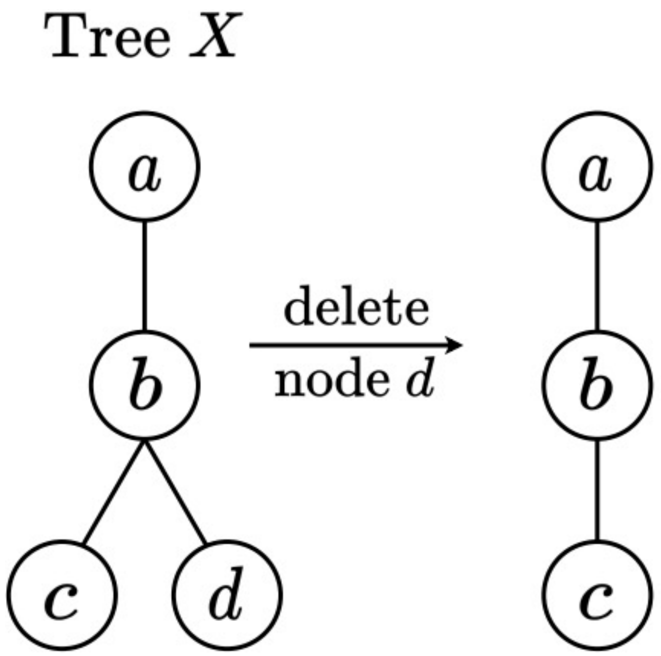
Tree Edit Distance (TED)

- Three edit operation types: delete, insert, rename



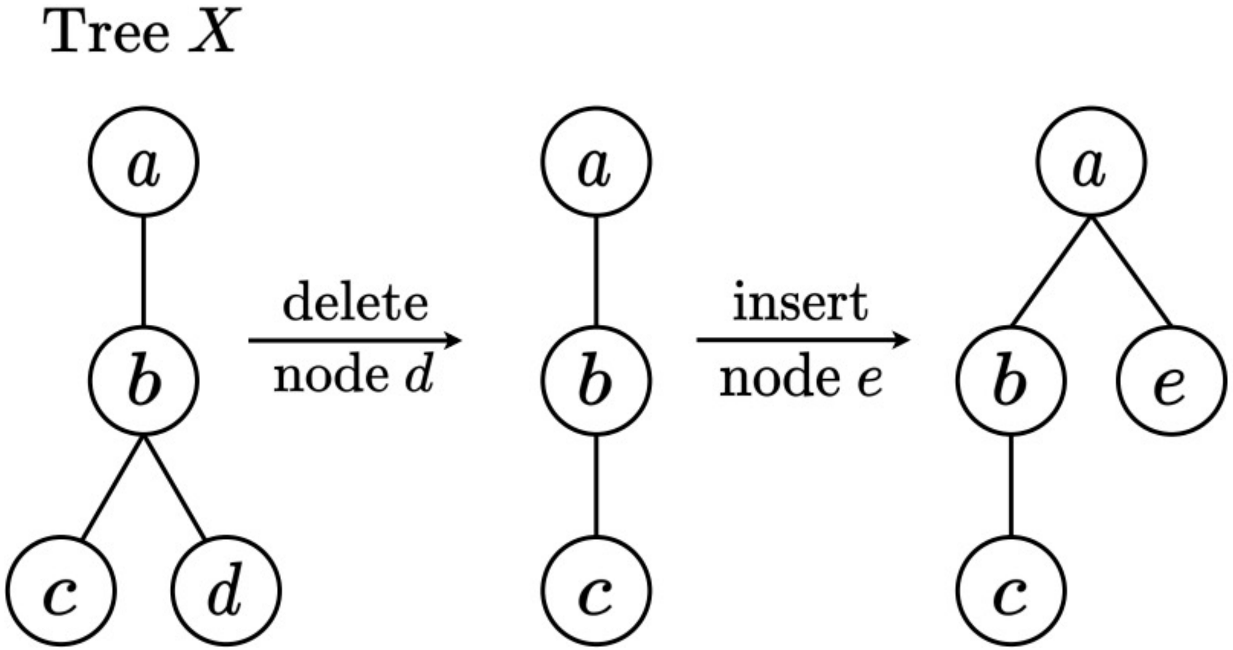
Tree Edit Distance (TED)

- Three edit operation types: delete, insert, rename



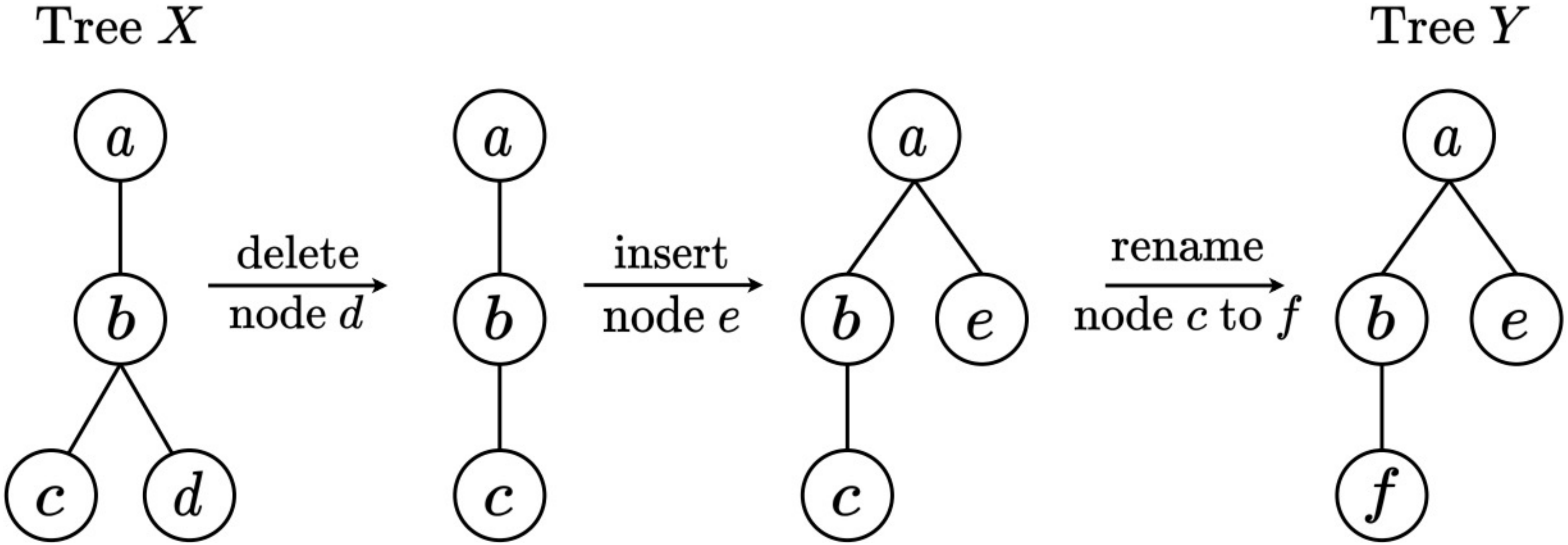
Tree Edit Distance (TED)

- Three edit operation types: delete, insert, rename



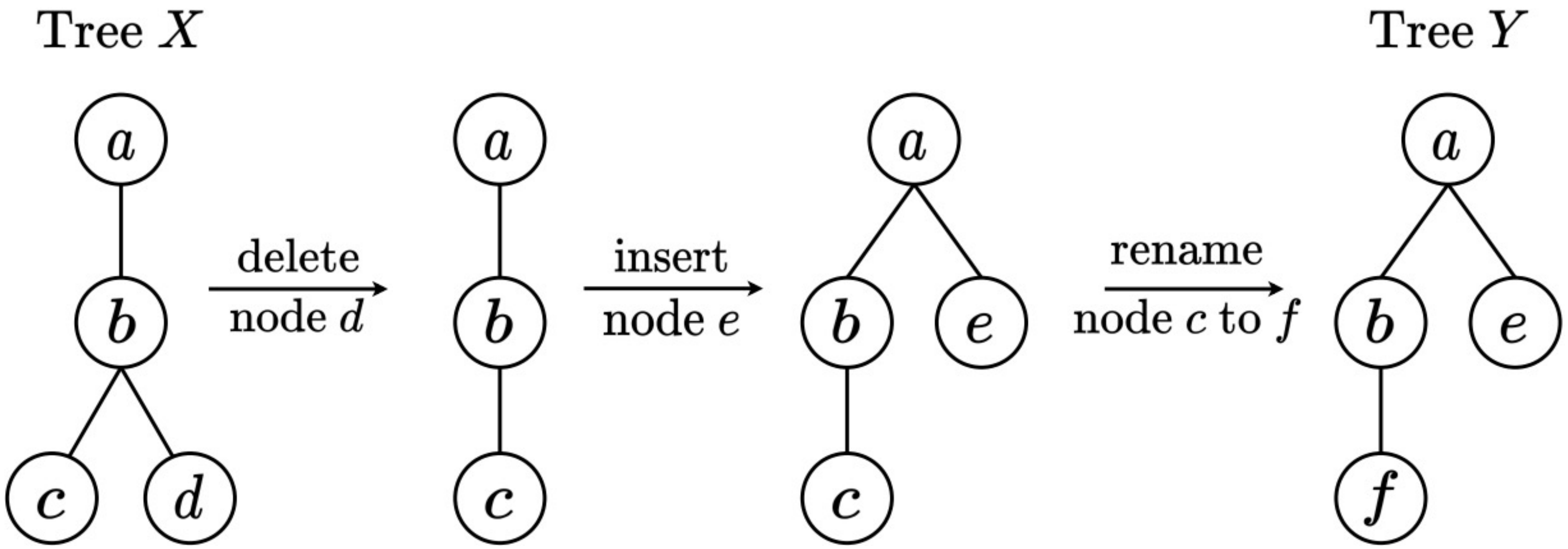
Tree Edit Distance (TED)

- Three edit operation types: delete, insert, rename



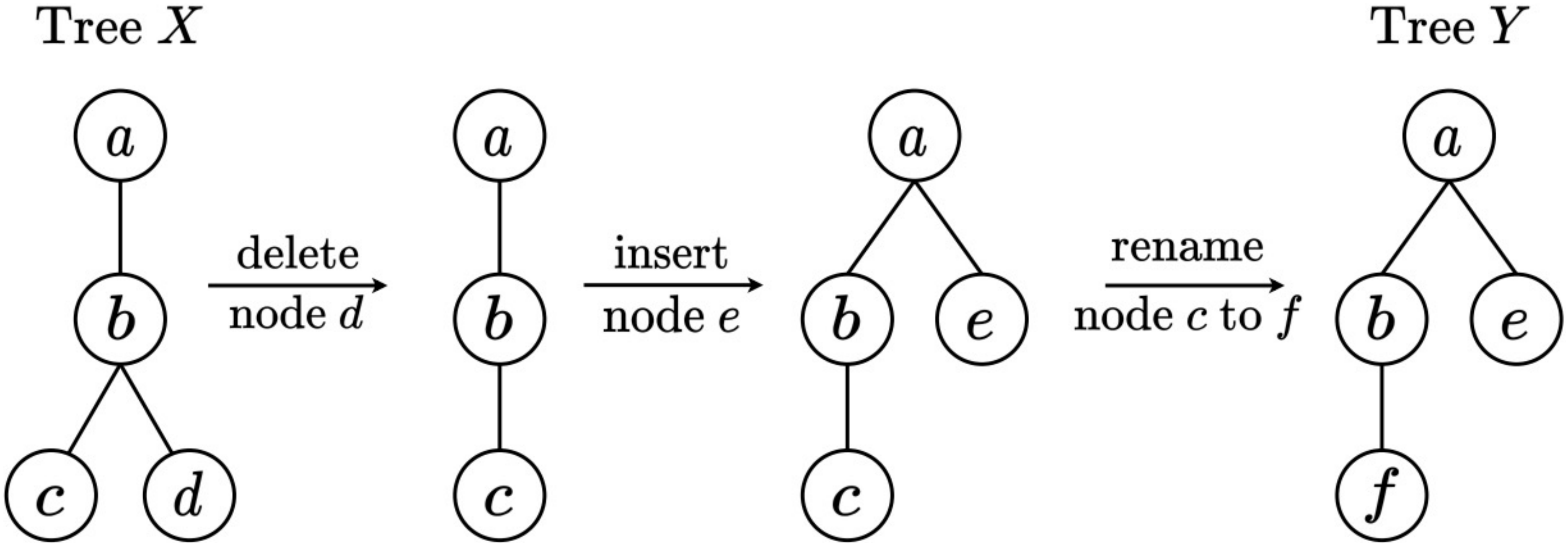
Tree Edit Distance (TED)

- If the cost of each edit operation is 1, …



Tree Edit Distance (TED)

- If the cost of each edit operation is 1, …

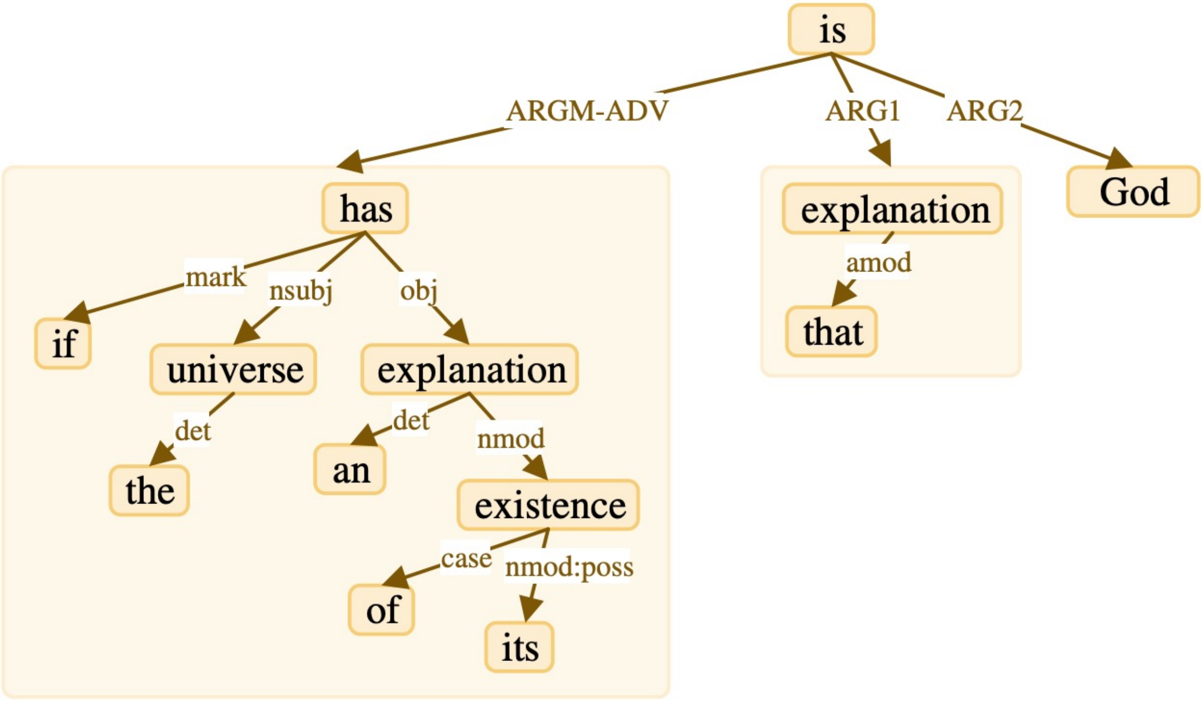


$$\mathrm{TED}(X, Y) = 3$$

TED has a wide range of applications

TED has a wide range of applications

- Natural language processing
  - AI Generated Content



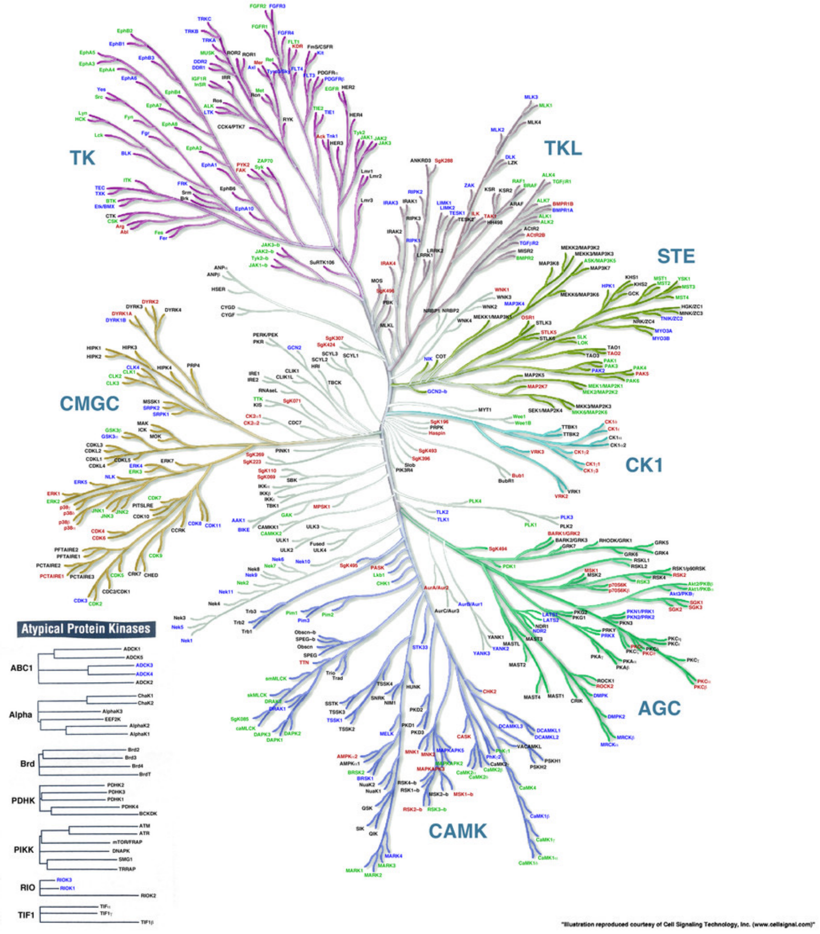
TED has a wide range of applications

- Natural language processing

  - AI Generated Content

- <u>Bioinformatics</u>

  - Protein structure and evolutionary



TED has a wide range of applications

- Natural language processing

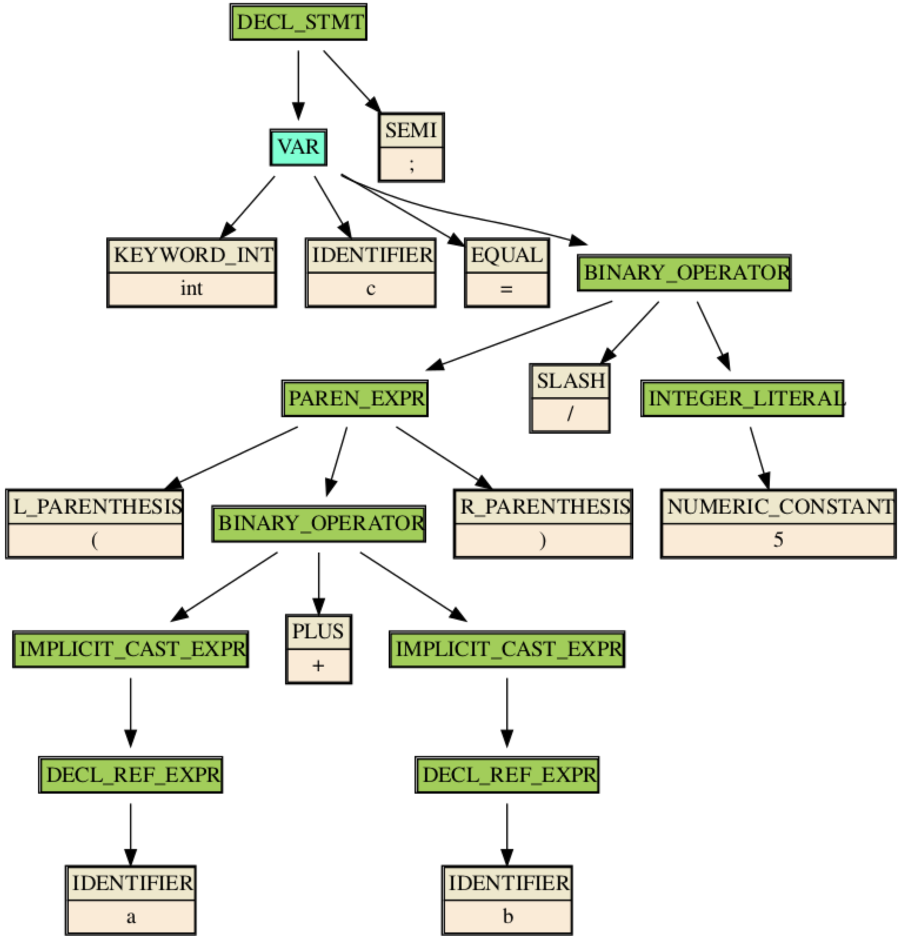
    - AI Generated Content

- Bioinformatics

    - Protein structure and evolutionary

- <u>Software Engineering</u>

    - Code similarity detection



TED has a wide range of applications

- Natural language processing

  - AI Generated Content

- Bioinformatics

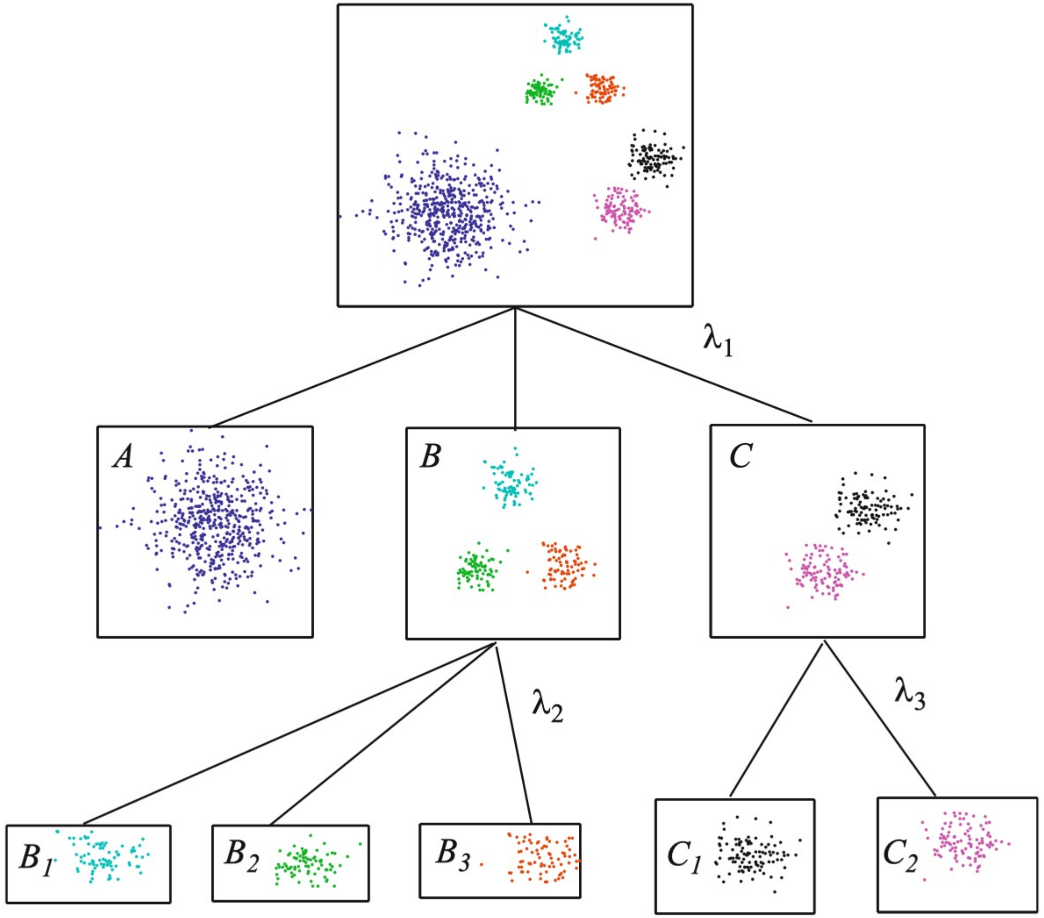
  - Protein structure and evolutionary

- Software Engineering

  - Code similarity detection

- Machine Learning

  - Classifications and clustering



- The first TED algorithm (1979) with complexity $O(n^6)$

The Basic TED Algorithm

- The most widely-used TED algorithm (1989)

The Basic TED Algorithm

- The most widely-used TED algorithm (1989)
- Dynamic programming (DP)

The Basic TED Algorithm

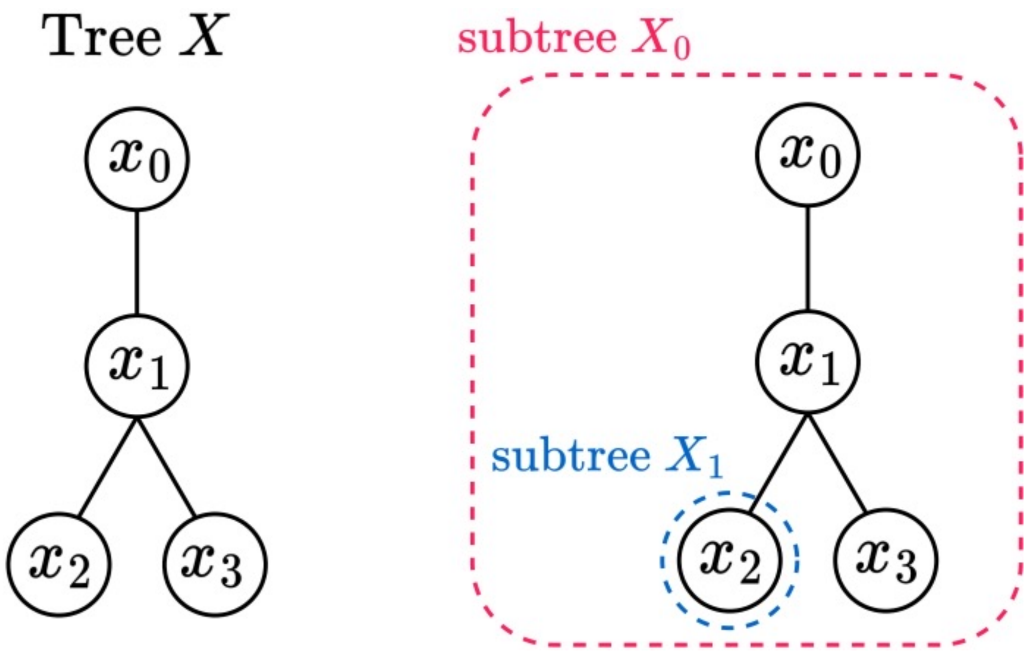
- The most widely-used TED algorithm (1989)
- Dynamic programming (DP)
- The worst-case complexity: $O(n^4)$

The Basic TED Algorithm

- Step 1 – divide tree to subtrees

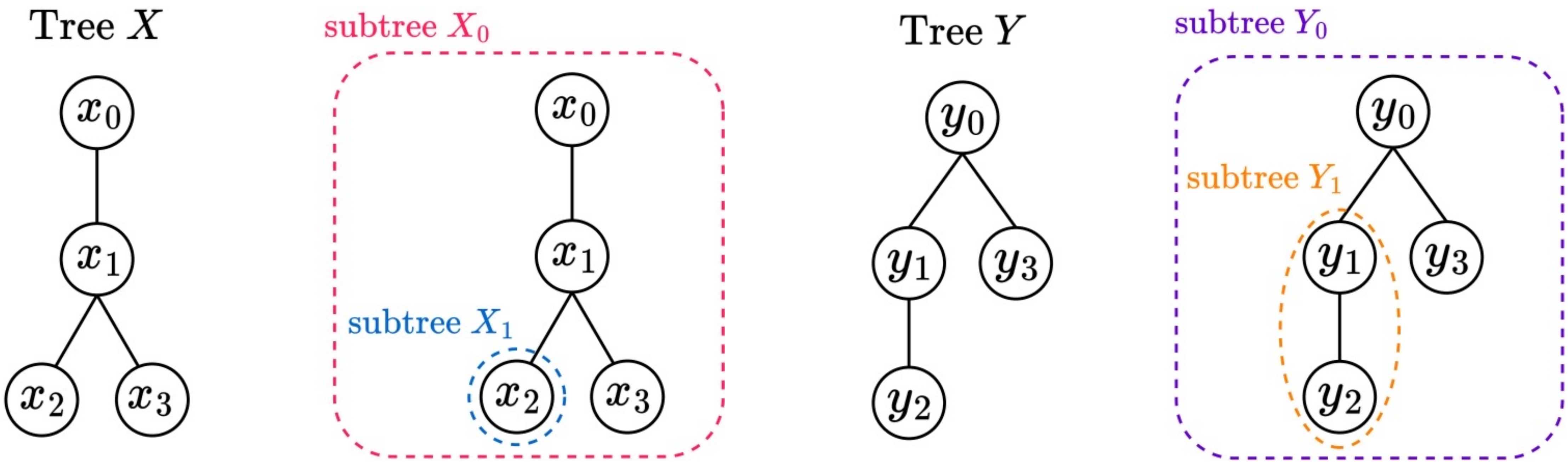
The Basic TED Algorithm

- Step 1 – divide tree to subtrees



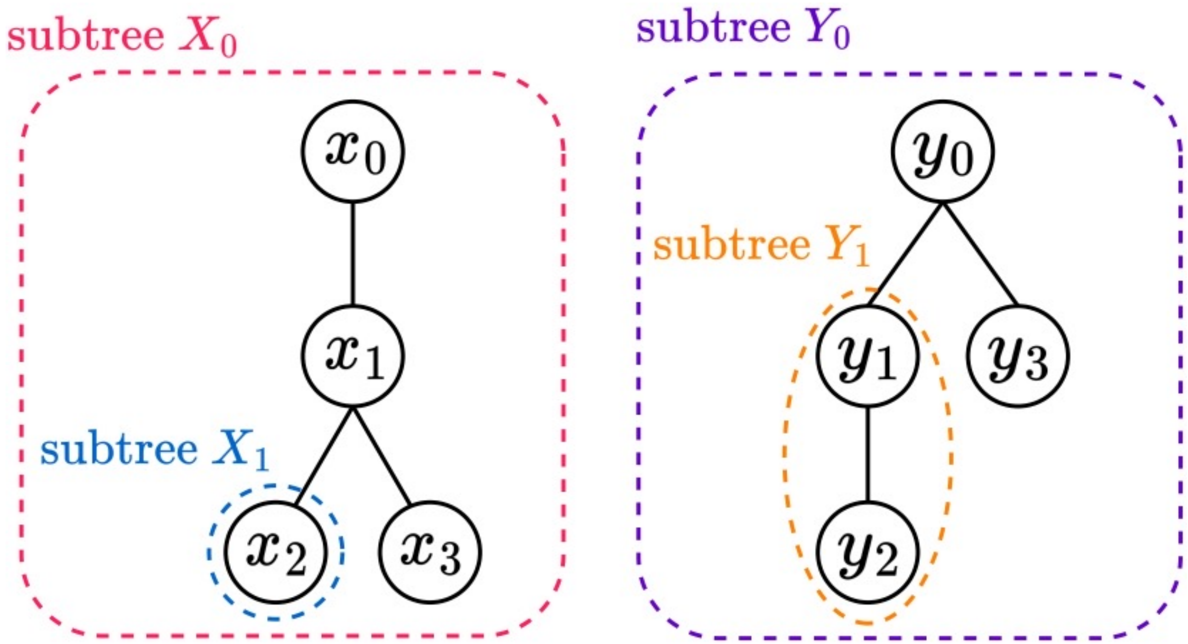
The Basic TED Algorithm

- Step 1 – divide tree to subtrees



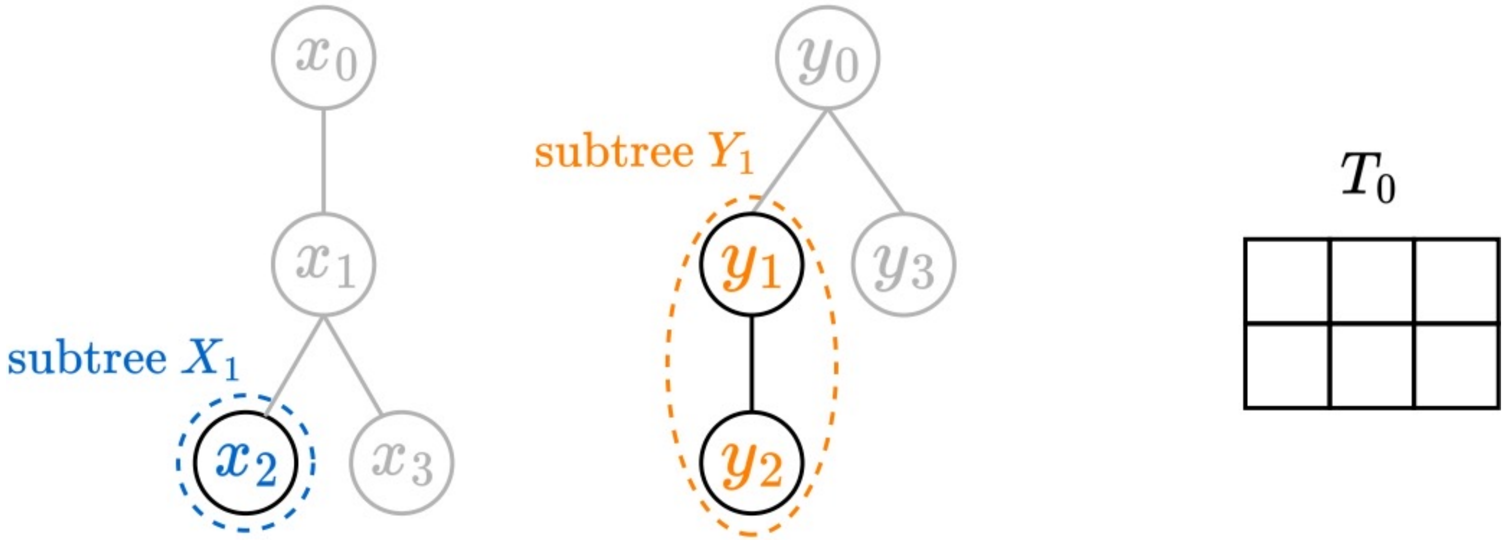
The Basic TED Algorithm

- Step 2 – DP tables to compute distance for each subtree pair



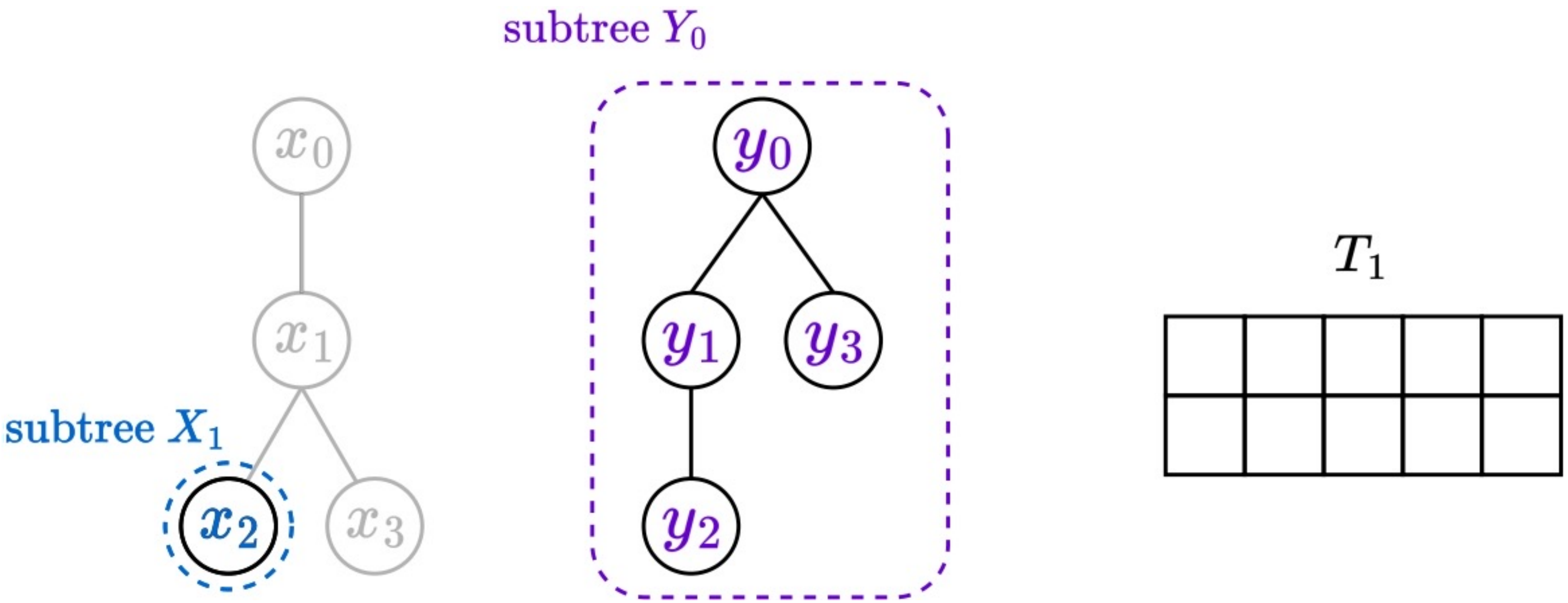
The Basic TED Algorithm

- 1st Table: DP table for subtree X_1 and subtree Y_1



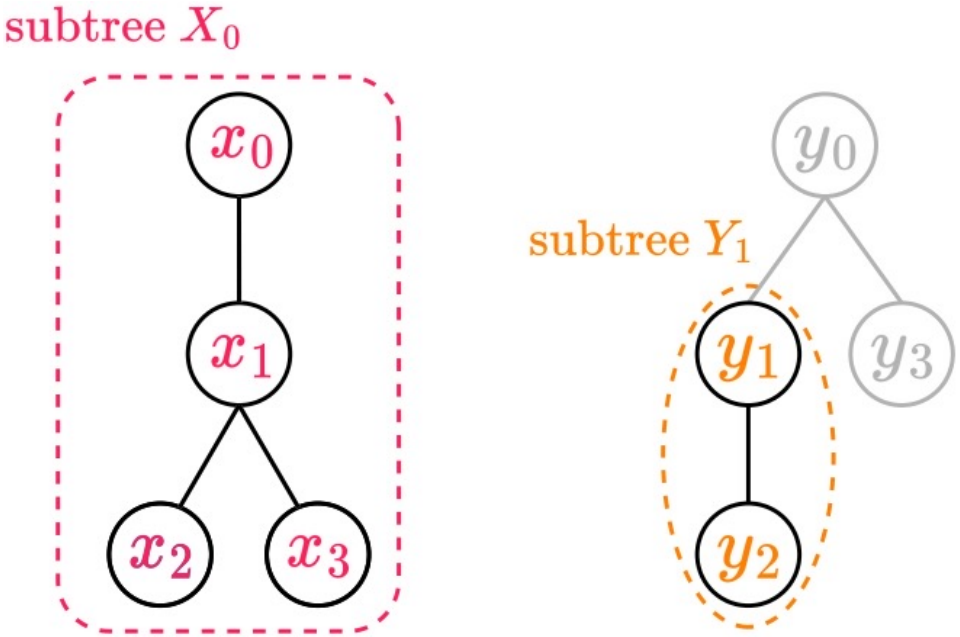
The Basic TED Algorithm

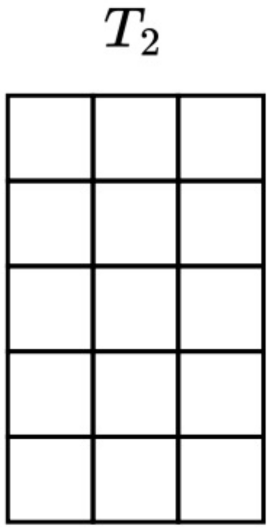
- 2^{nd} Table: DP table for subtree X_1 and subtree Y_0



The Basic TED Algorithm

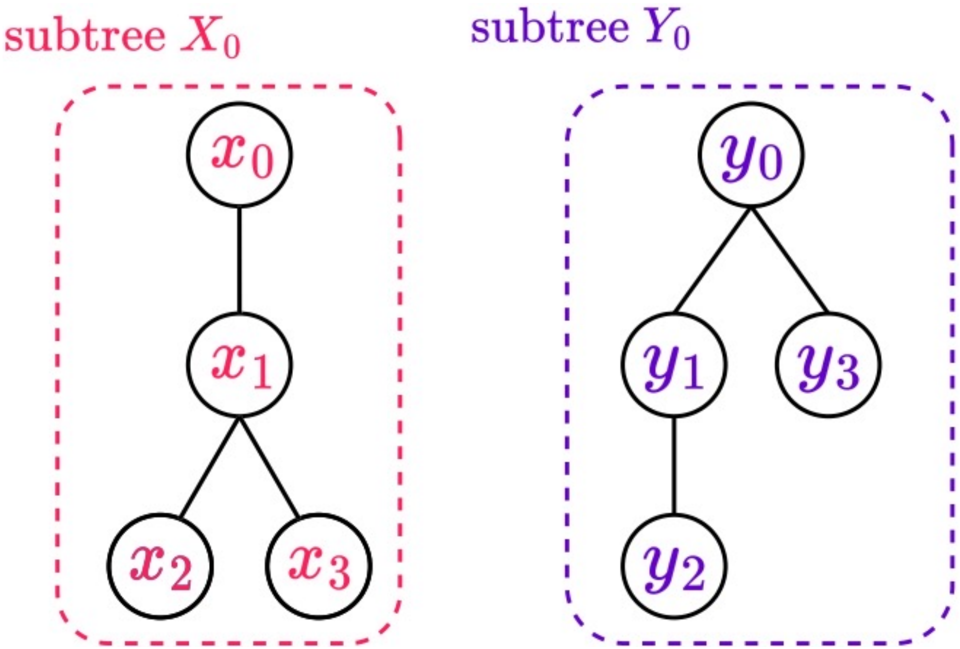
- 3rd Table: DP table for subtree X_0 and subtree Y_1





The Basic TED Algorithm

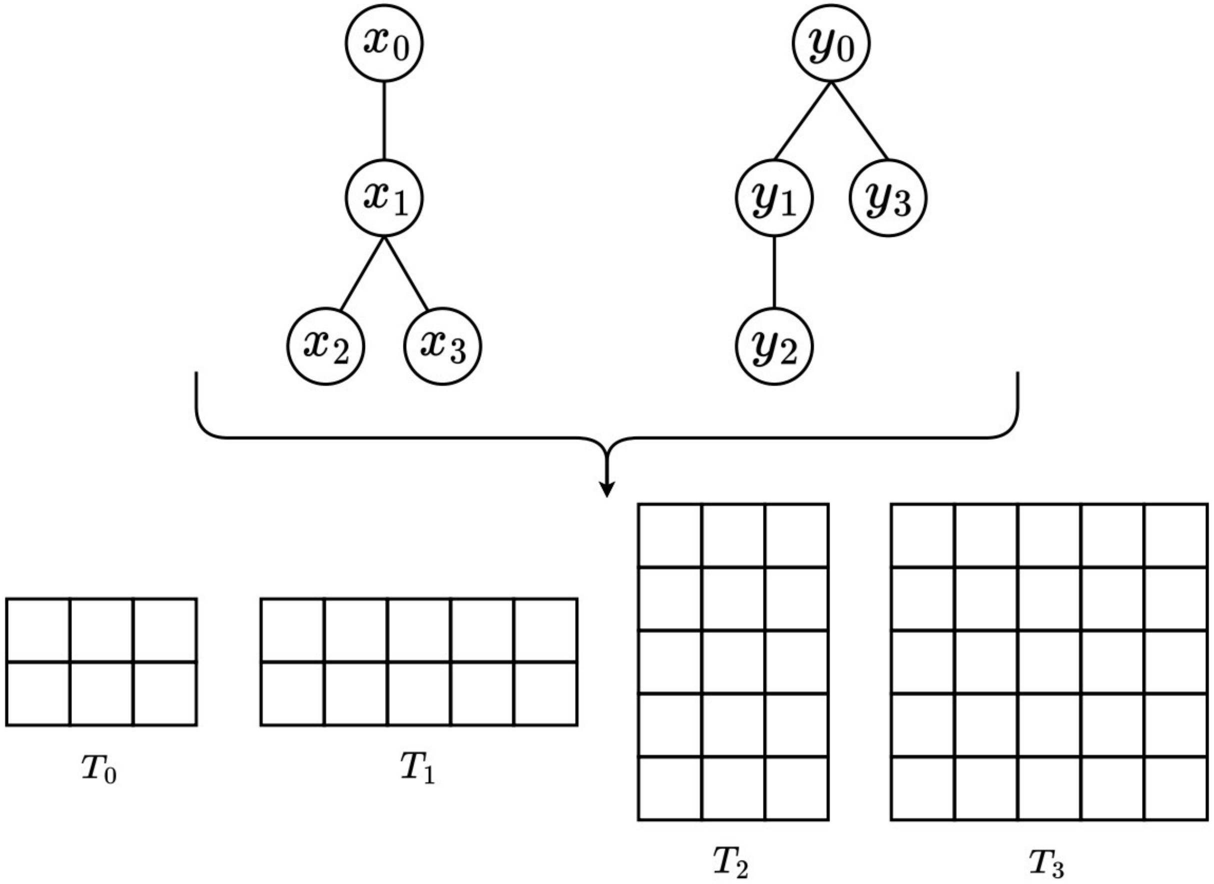
- 4^{th} Table: DP table for subtree X_0 and subtree Y_0



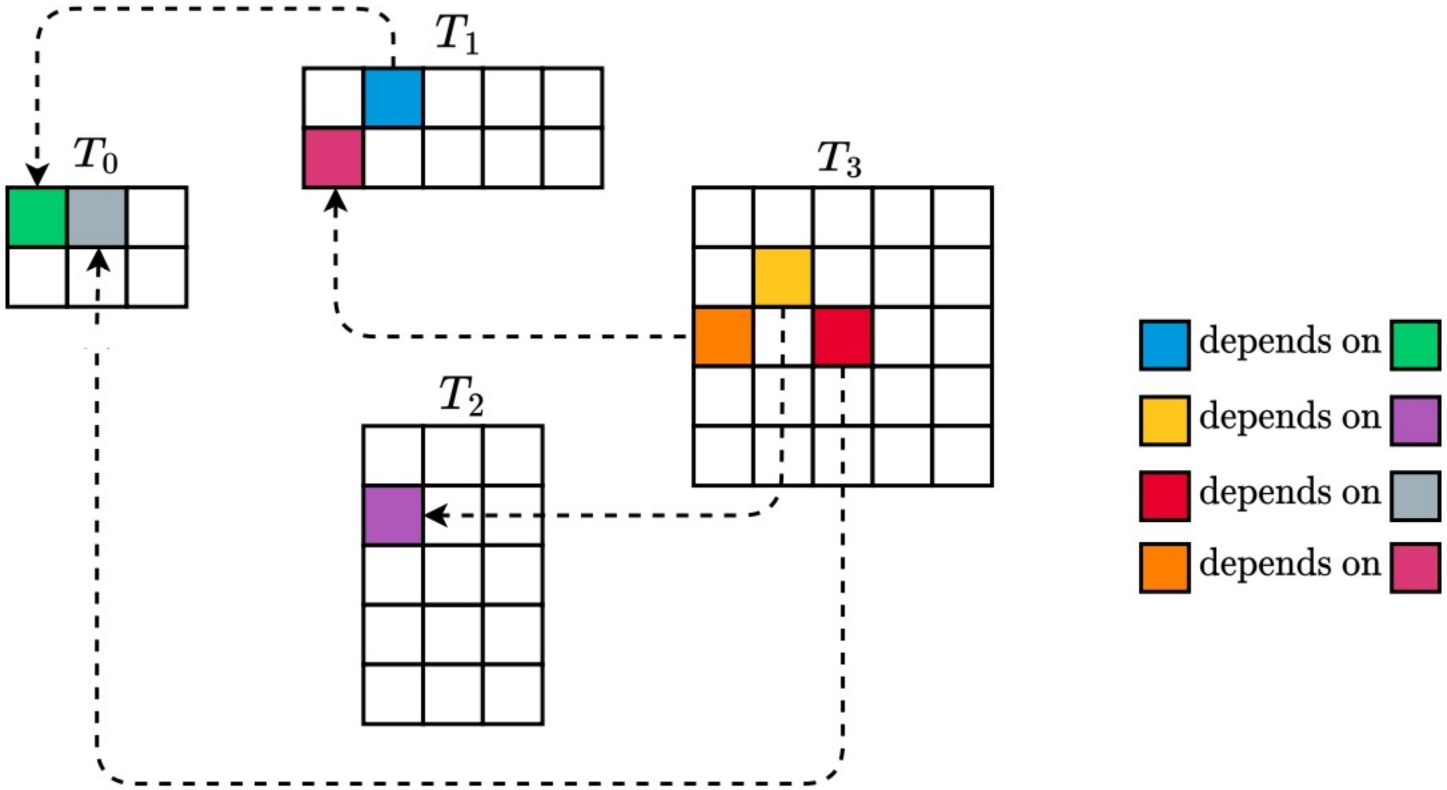
T_3

The Basic TED Algorithm

- Step 3 – return TED result after all DP tables are computed



Data Dependencies Example



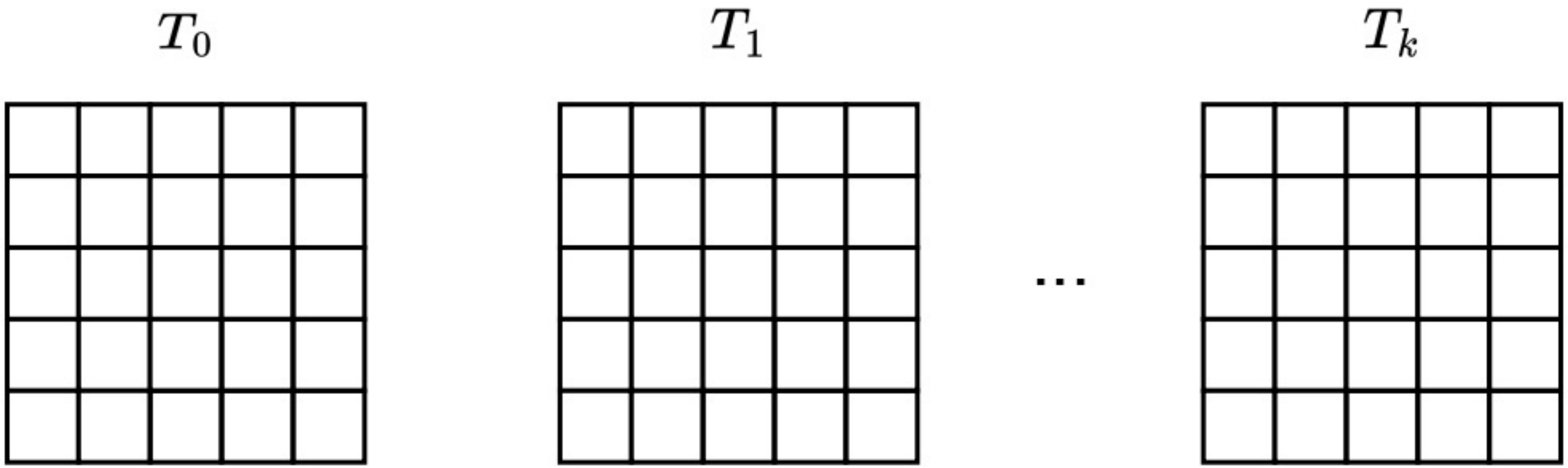
- Hinder parallel processing

Existing Parallel Solution

- Wave-front parallel computing

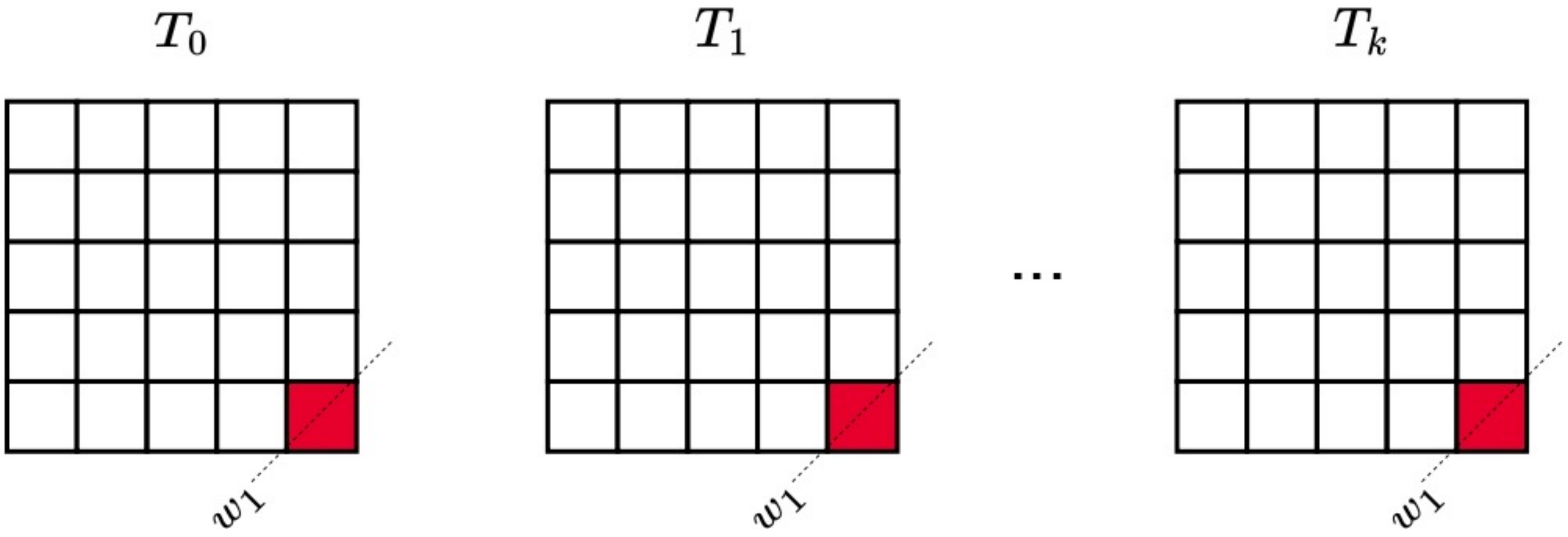
Existing Parallel Solution

- Wave-front parallel computing



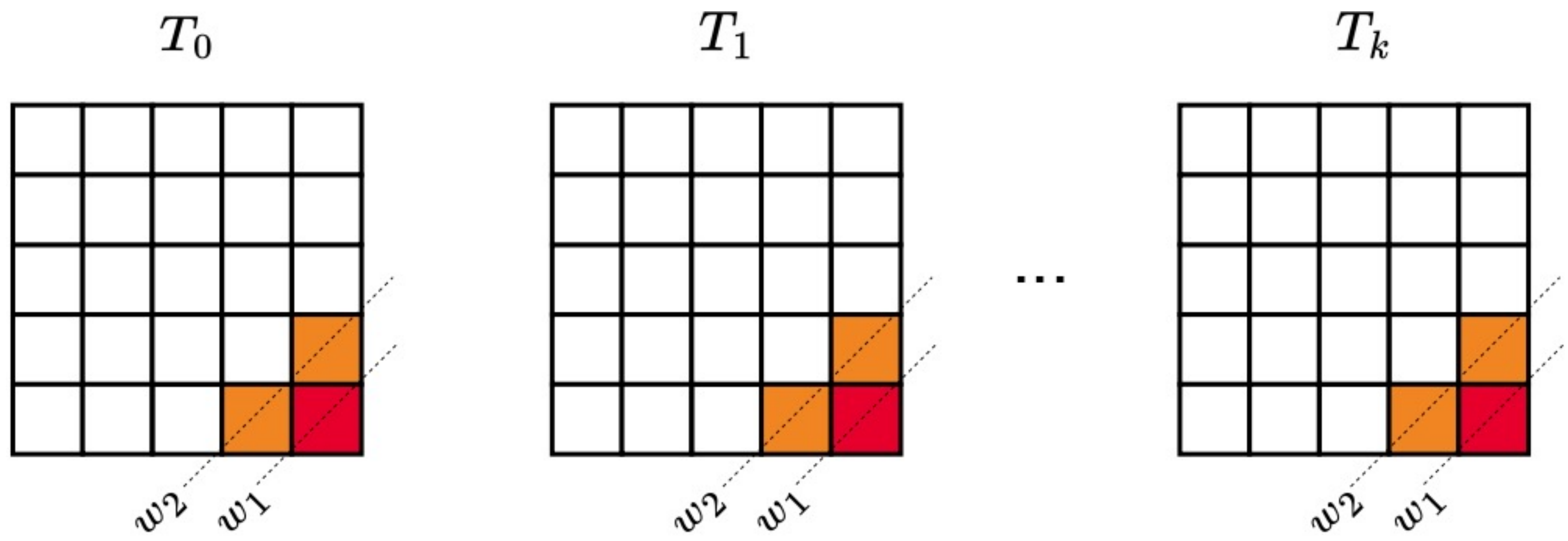
Existing Parallel Solution

- First, compute red units for all tables in parallel



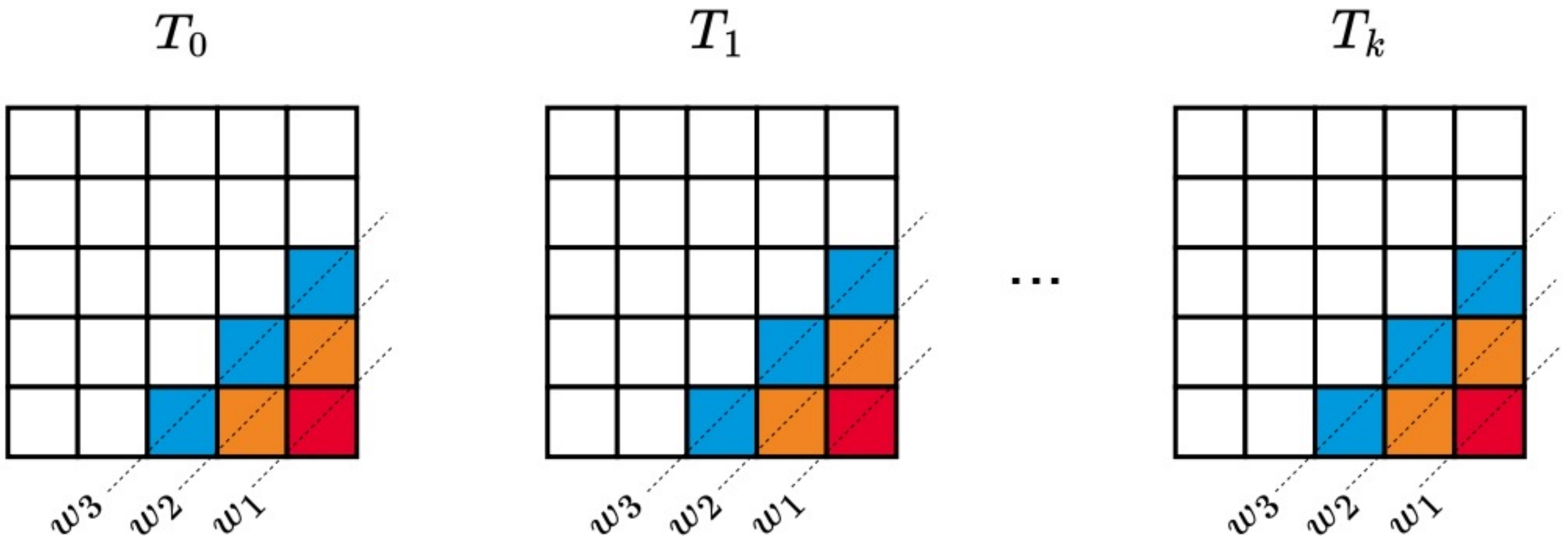
Existing Parallel Solution

- Then, compute orange units for all tables in parallel

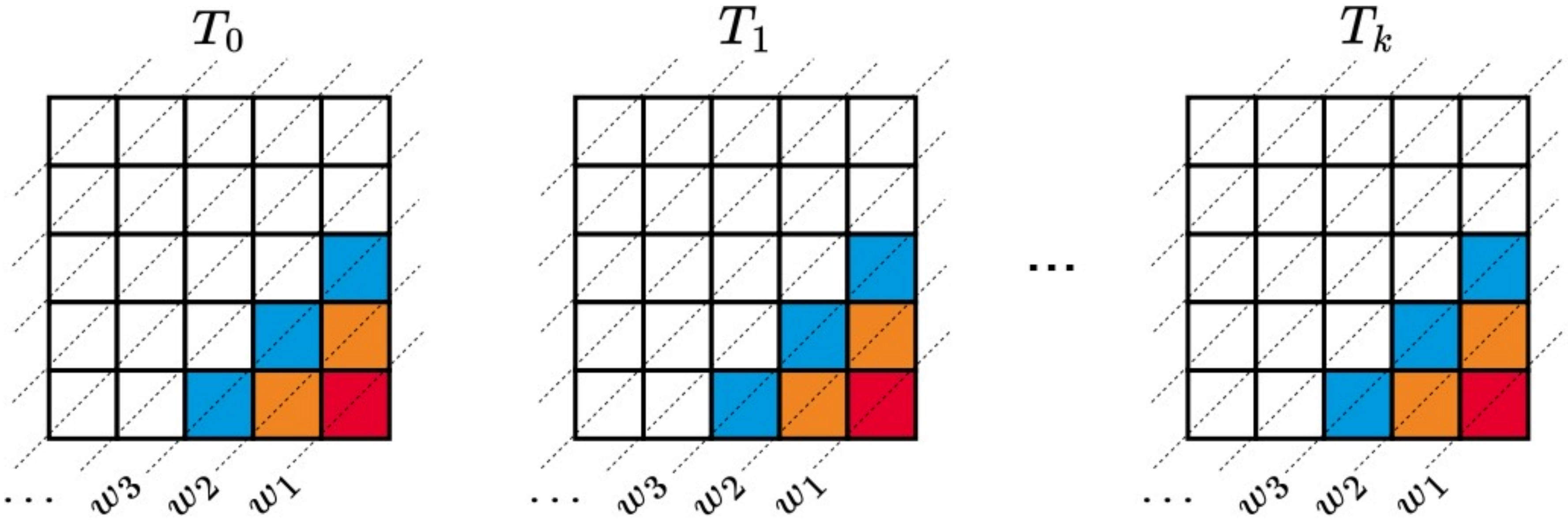


Existing Parallel Solution

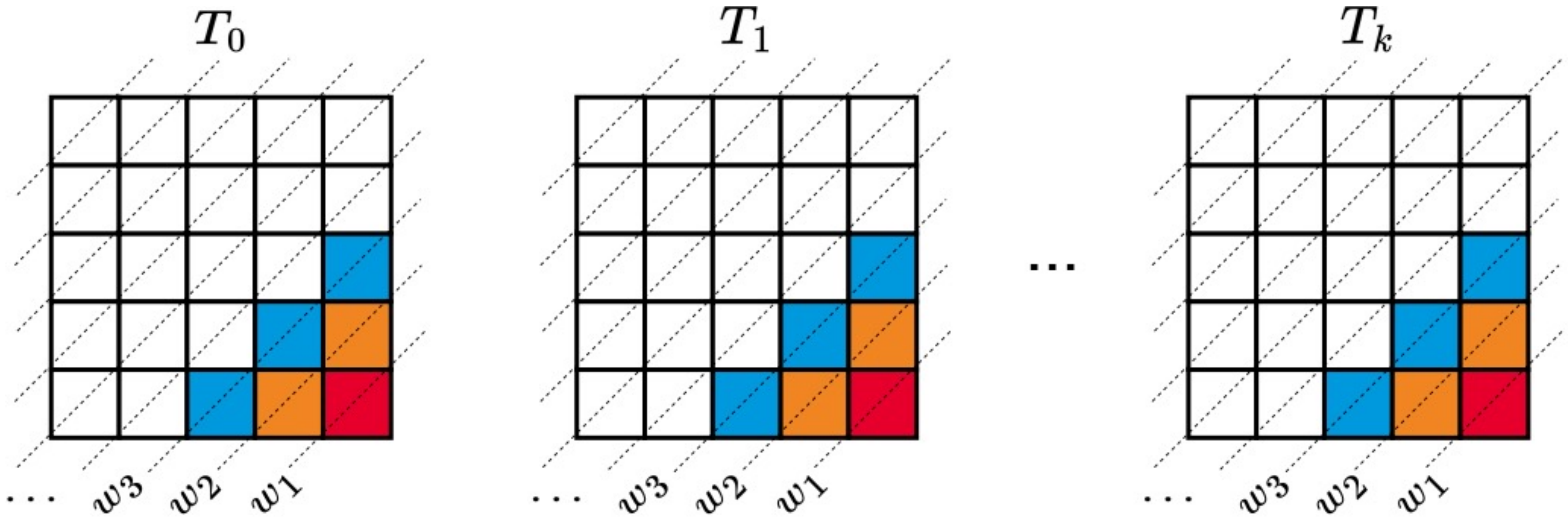
- Next, compute blue units for all tables in parallel



Existing Parallel Solution

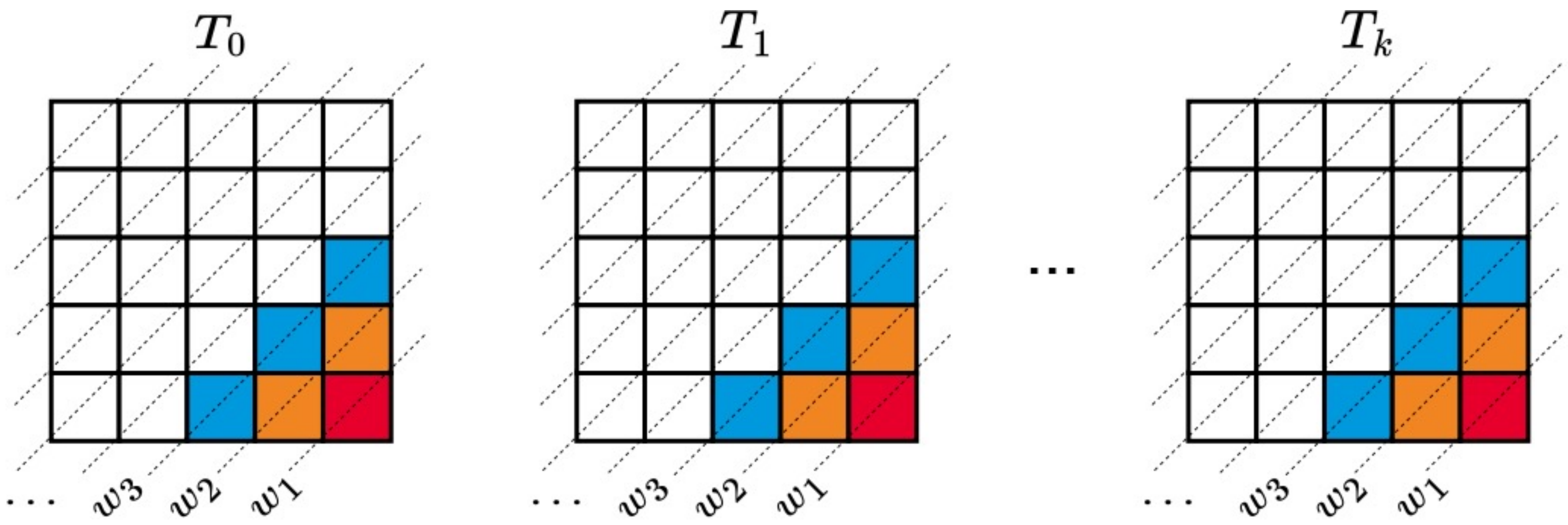


Existing Parallel Solution



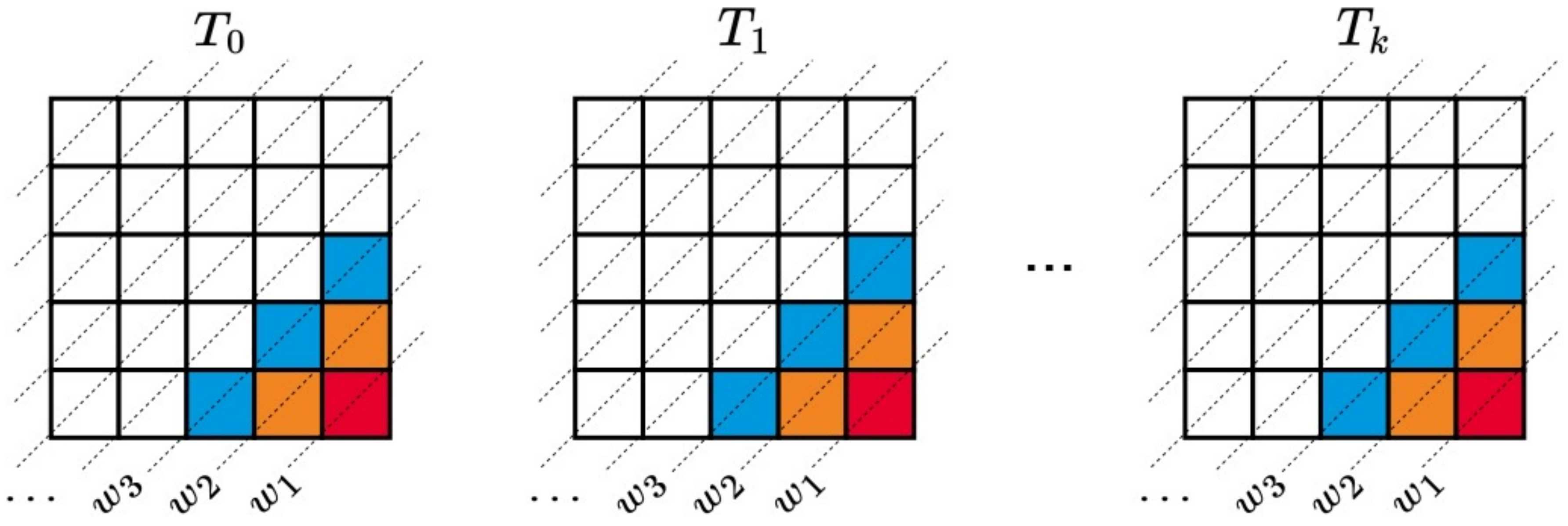
- A huge memory space

Existing Parallel Solution



- A huge memory space
- Frequent synchronizations

Existing Parallel Solution



- A huge memory space
- Frequent synchronizations
- Load-imbalanced

Insights from DP Patterns

Insights from DP Patterns

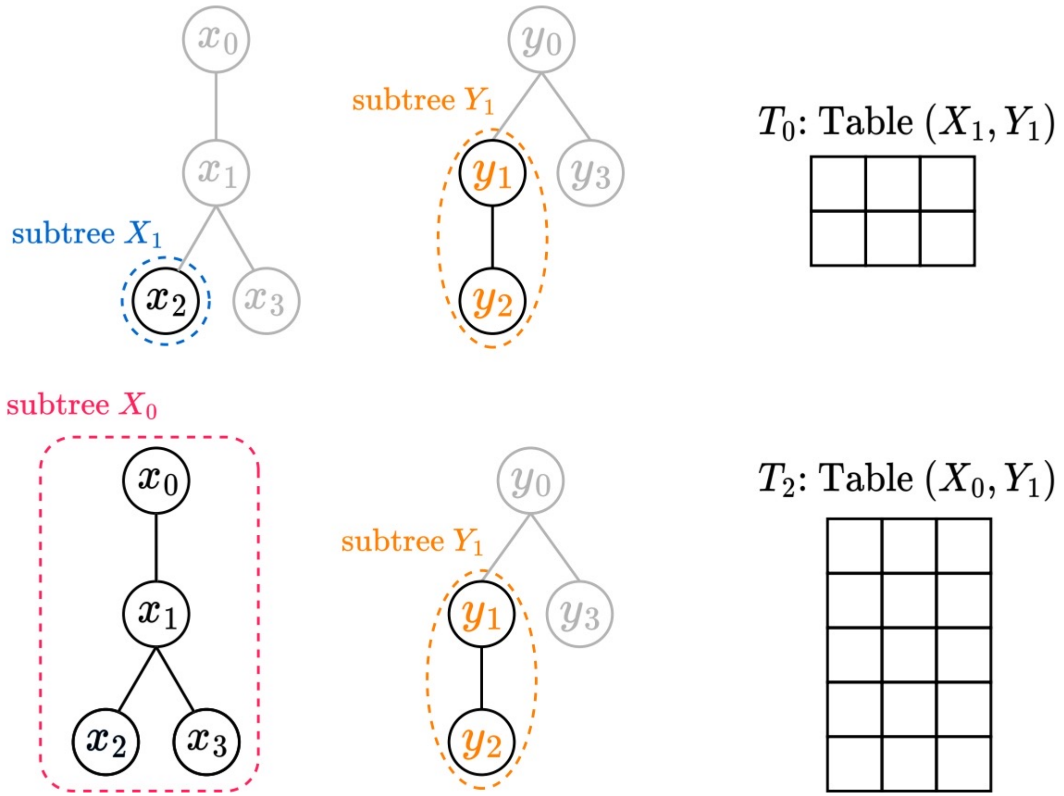
- The computing of one DP table is viewed as a task

Insights from DP Patterns

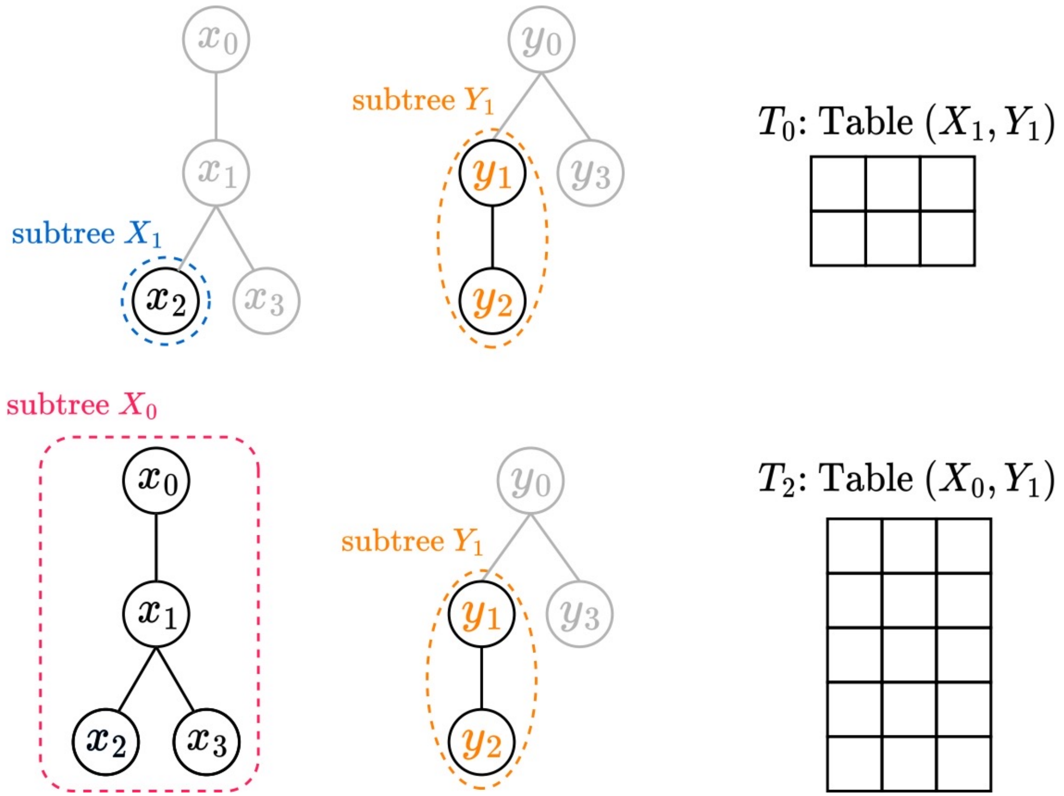
- The computing of one DP table is viewed as a task

- Data dependencies among tables are ***determined by tree structure***

Insights from DP Patterns

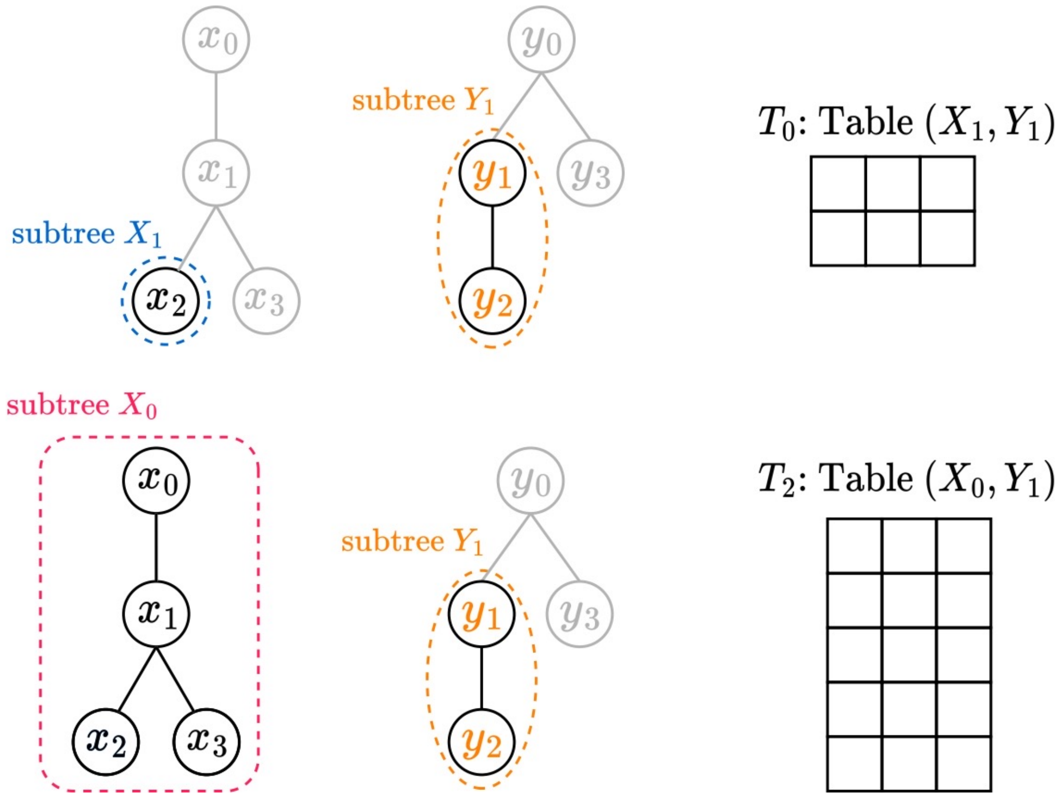


Insights from DP Patterns



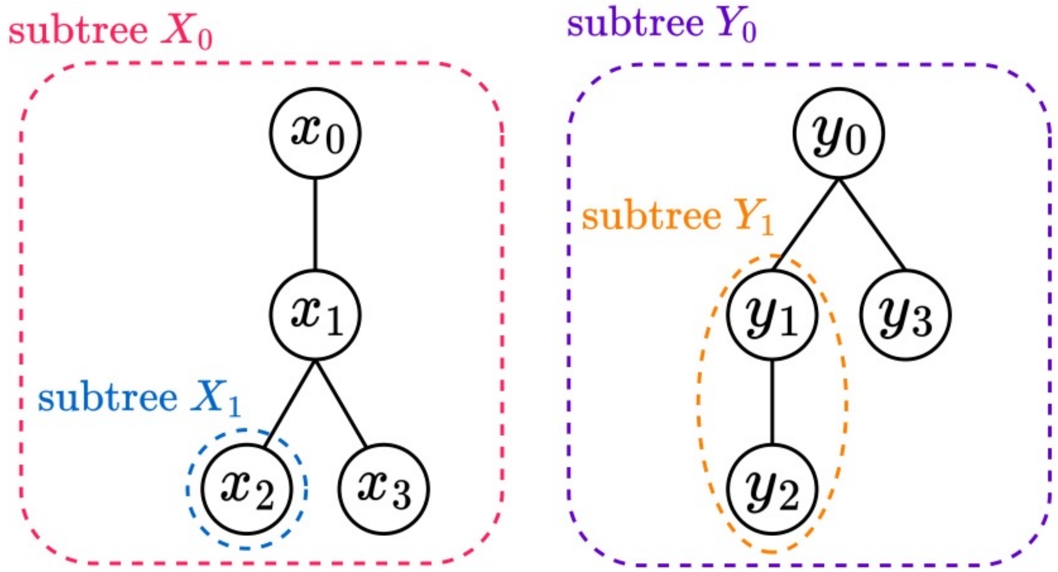
- subtree X_0 contains X_1

Insights from DP Patterns

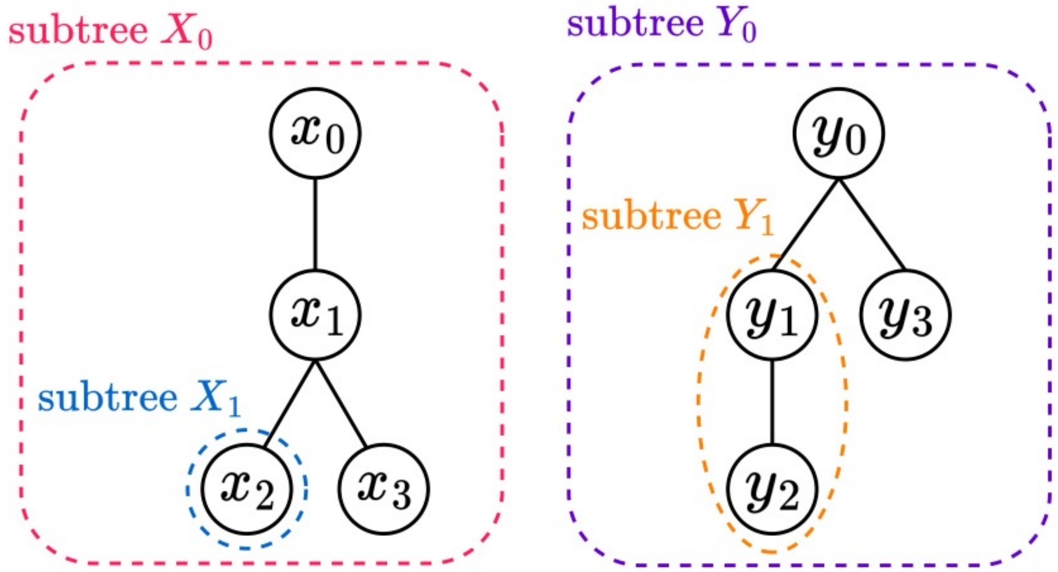


- subtree X_0 contains X_1 ⟶ Table (X_0, Y_1) depends on Table (X_1, Y_1)

Insights from DP Patterns

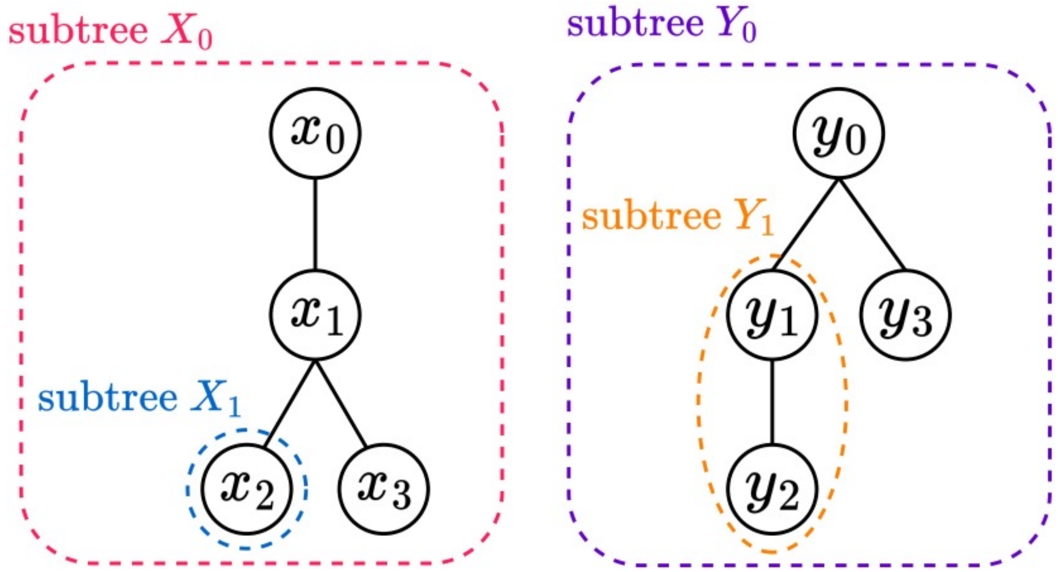


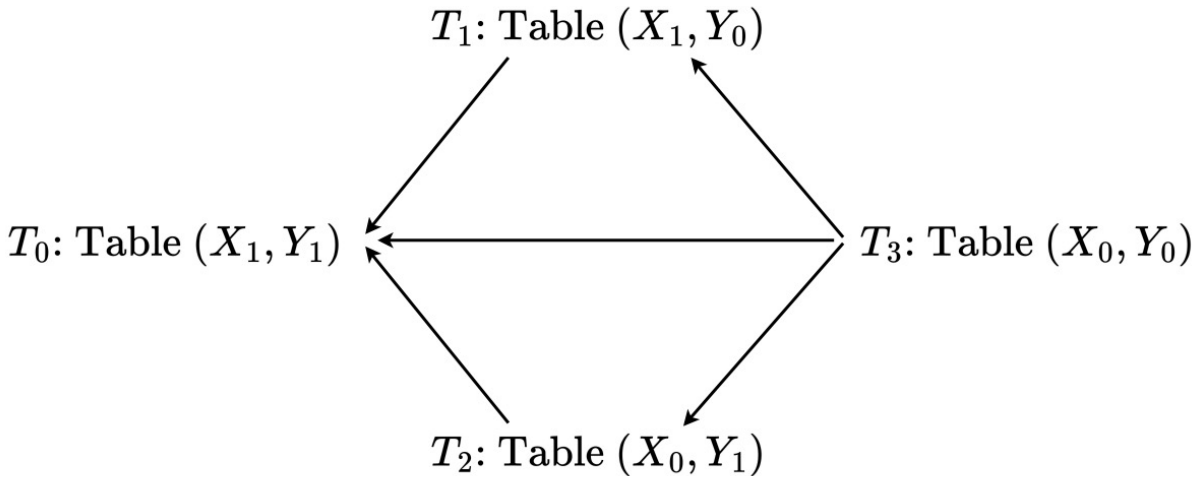
Insights from DP Patterns



- subtree X_0 contains X_1, subtree Y_0 contains Y_1

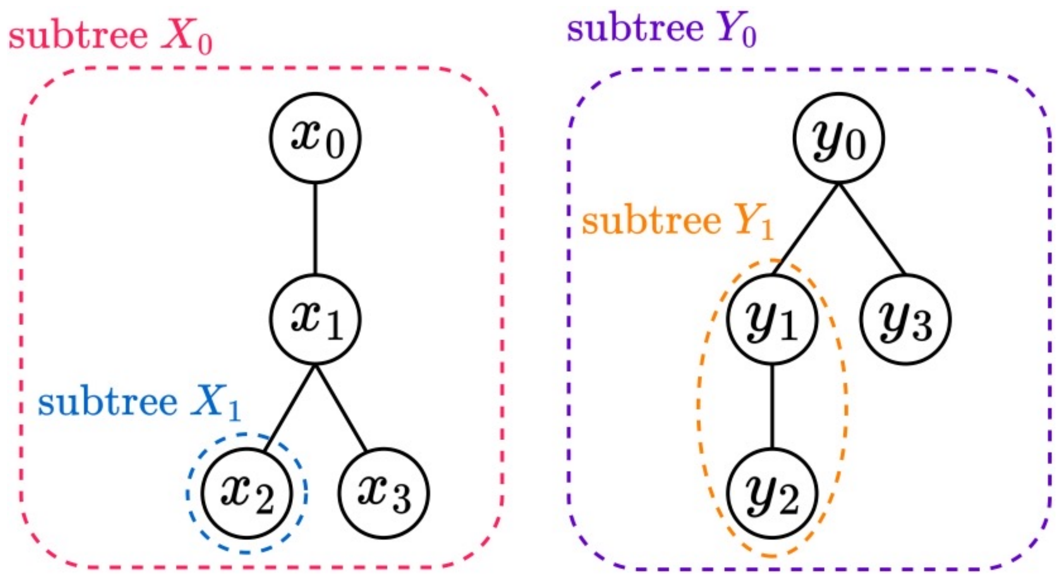
Insights from DP Patterns

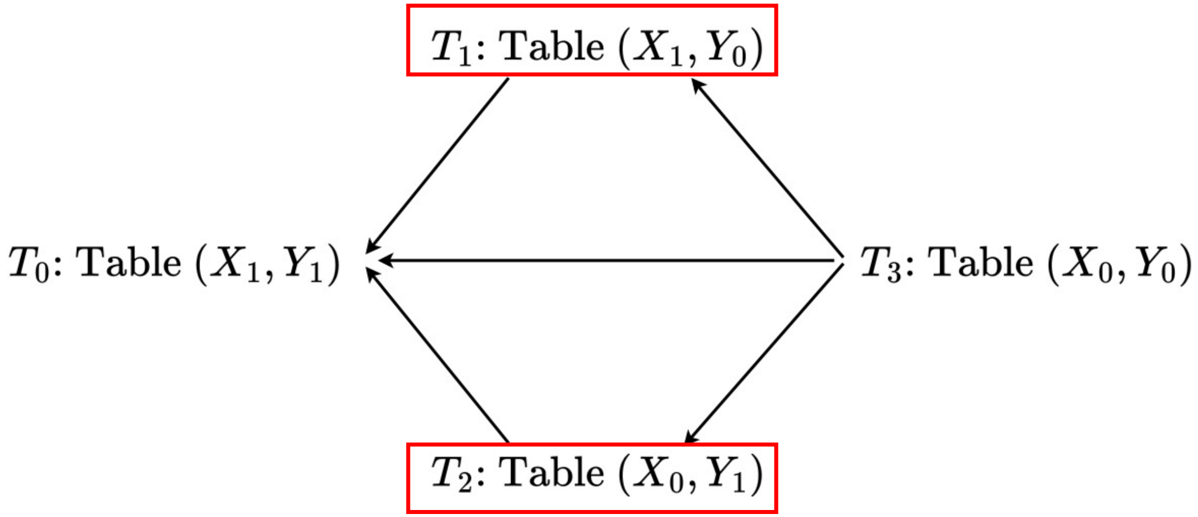




- subtree X_0 contains X_1, subtree Y_0 contains Y_1

Insights from DP Patterns

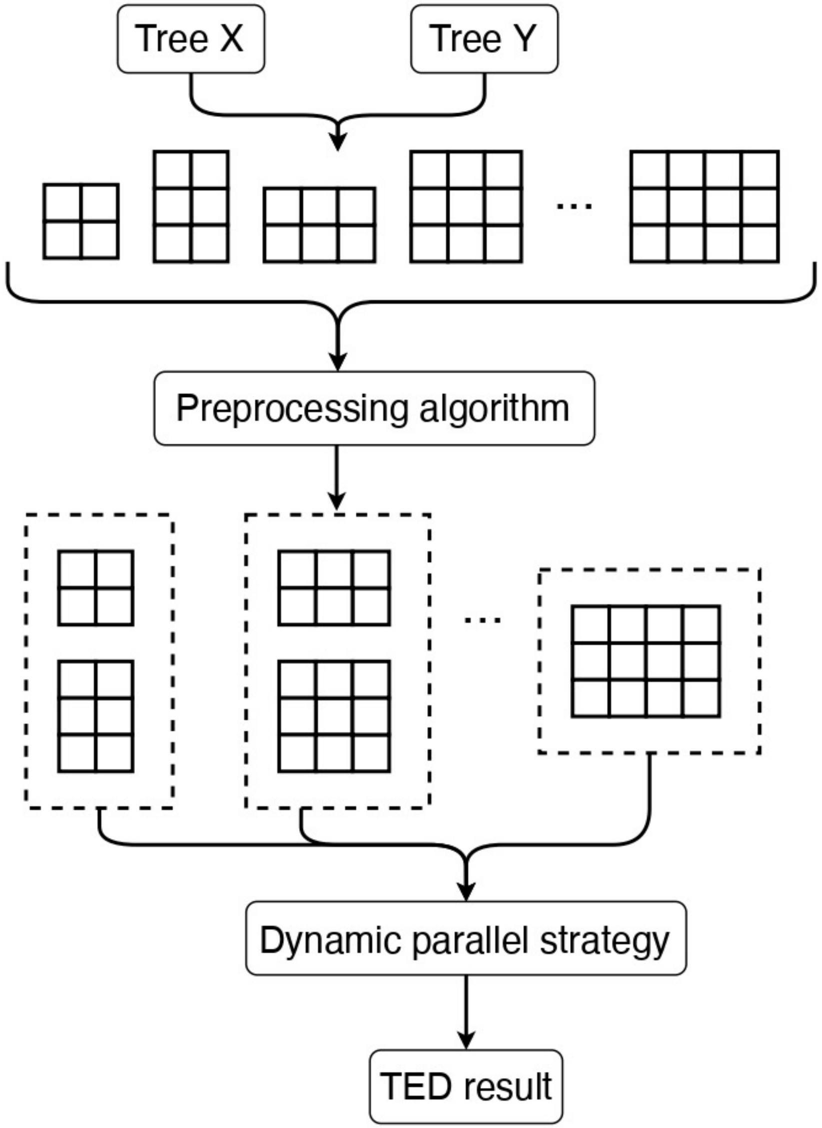




- T_1 and T_2 then can be computed in parallel

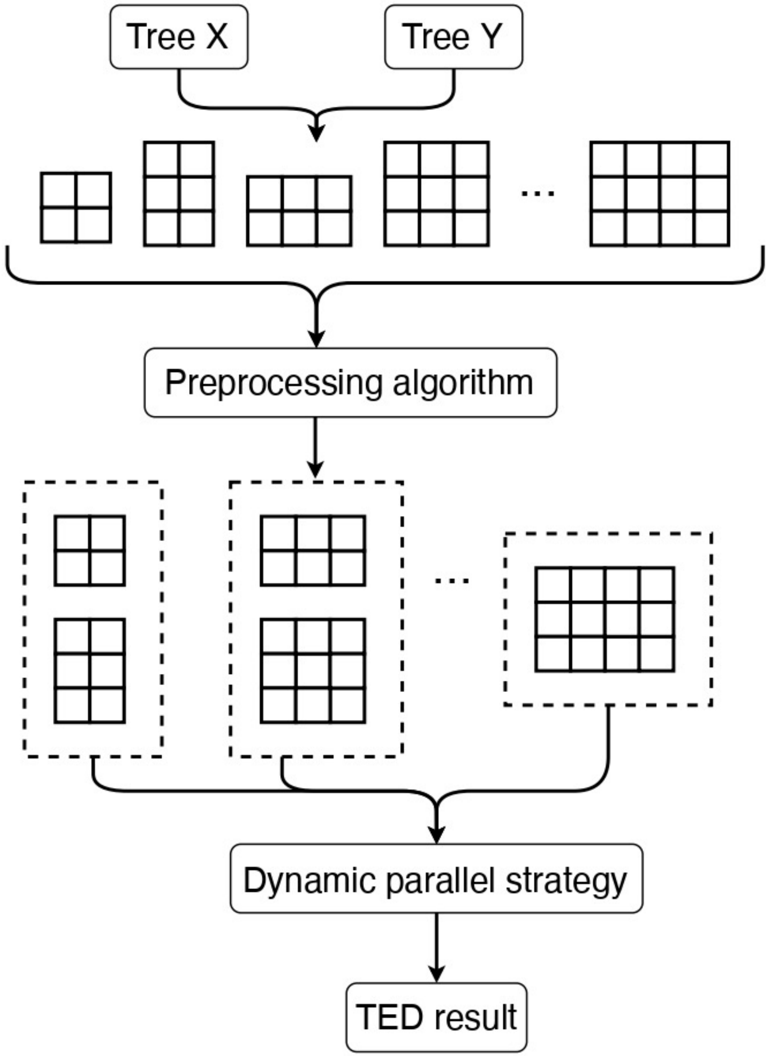
Our Solution: **X-TED** for Massively Parallel Computing

Our Solution: **X-TED** for Massively Parallel Computing



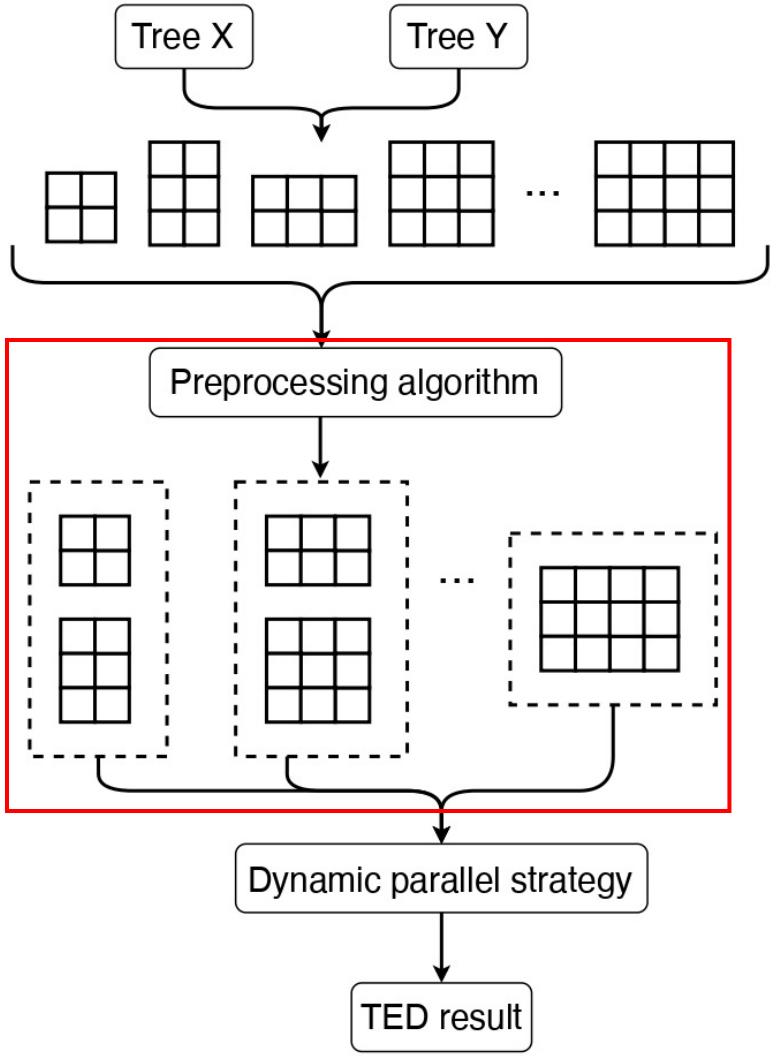
X-TED

- Memory saving



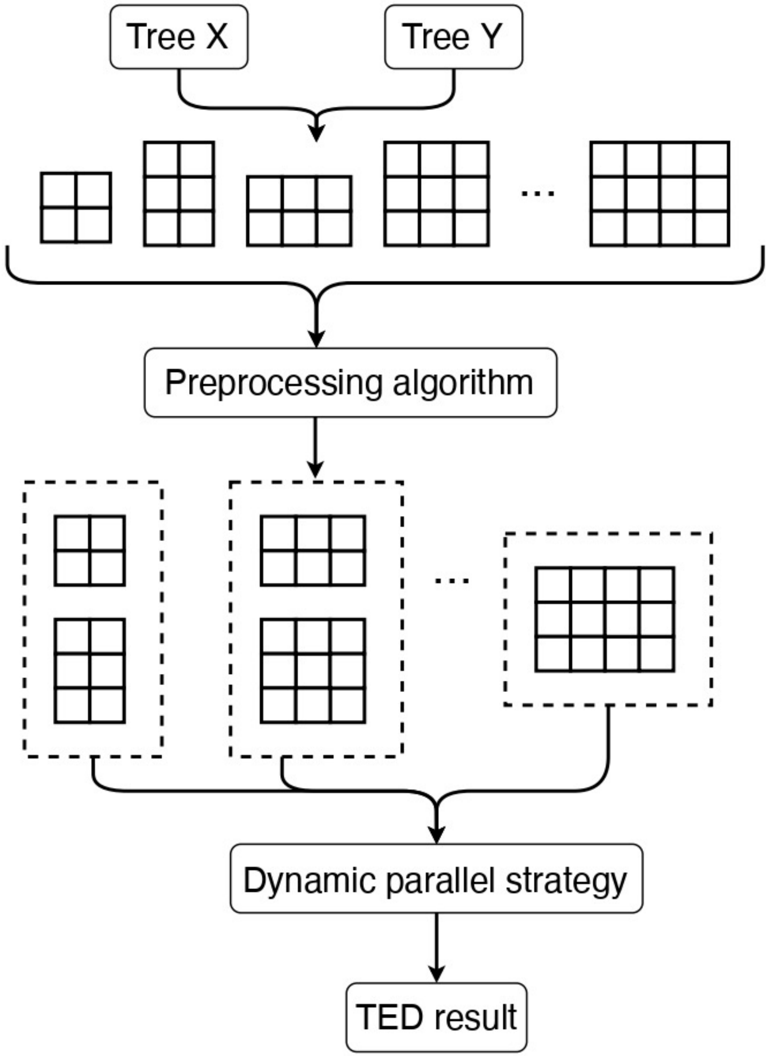
X-TED

- Memory saving
  - Each table is a task
  - Each processor only stores one table



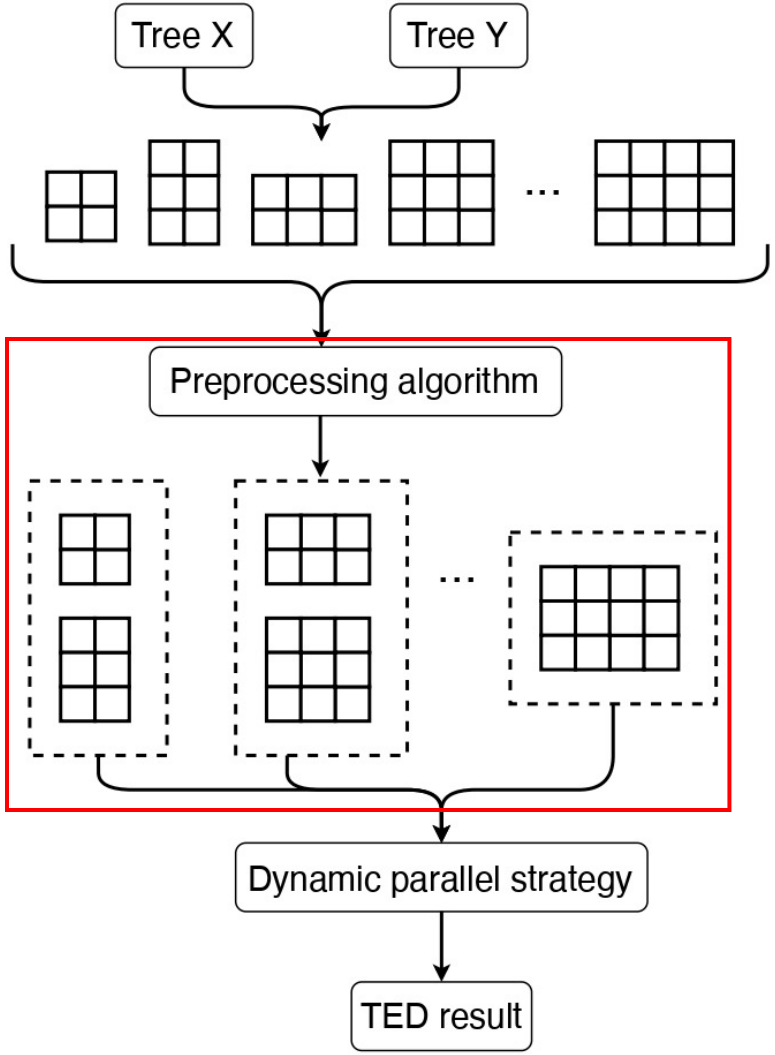
X-TED

- Less synchronizations



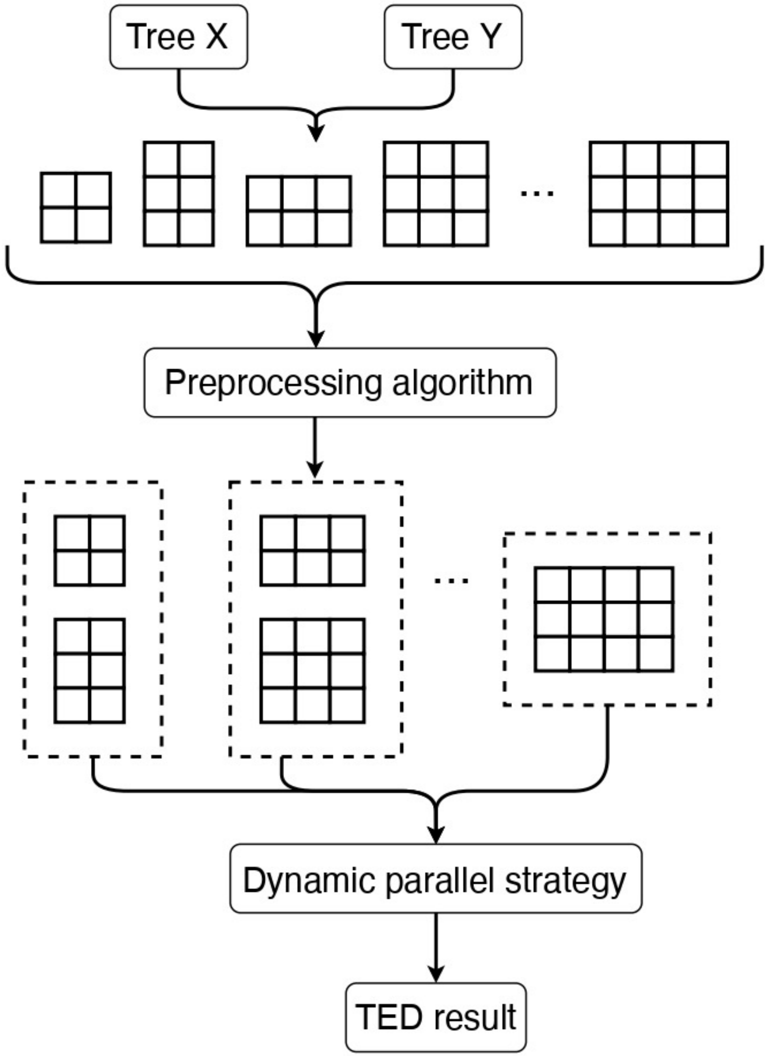
X-TED

- Less synchronizations
  - Only sync. once at each batch



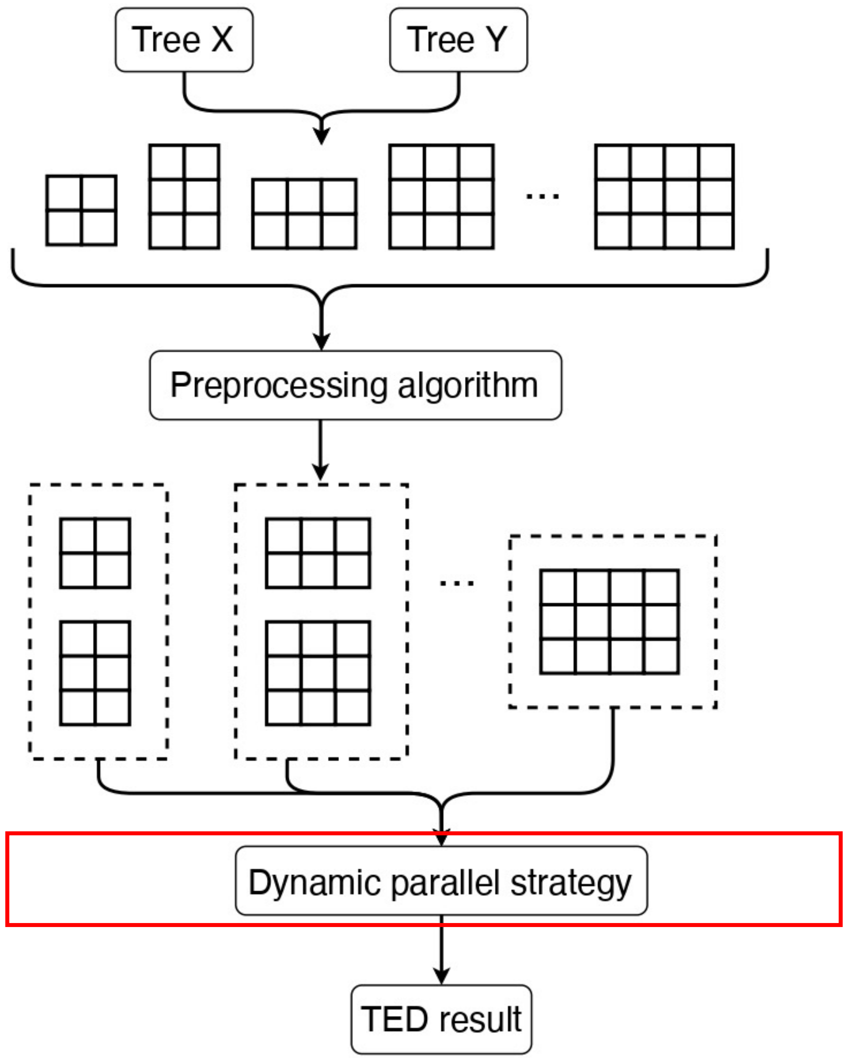
X-TED

- Load-balanced



X-TED

- Load-balanced
  - Different strategies for tables with different sizes



Experiment Setup

Dataset	Max. Depth	Avg. Depth	Avg. Nodes	Max. Nodes
Swissport	9	7.01	988.36	7241
Python	156	13.11	927.41	8516
DBLP	7	3.16	26.05	1186
Bolzano	4	3.82	178.71	2105

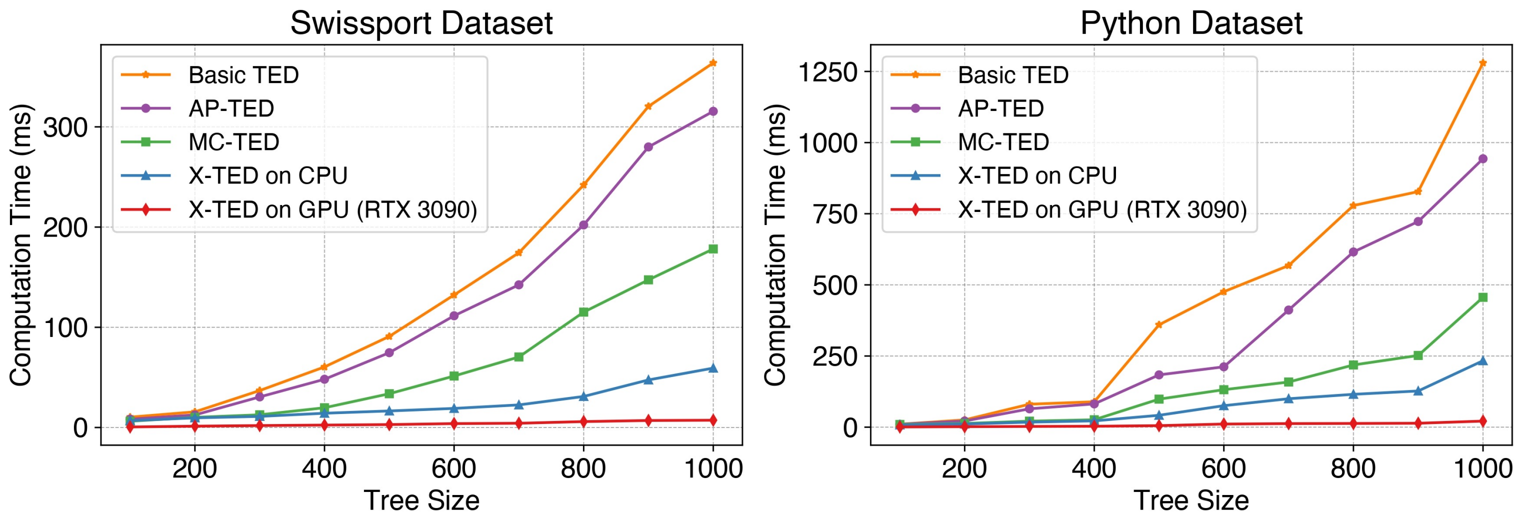
- Synthetic dataset: random recursive trees with 1000 to 9000 nodes
- For CPU baselines: Intel Core i9-12900 CPU, 8 Cores, 64GB
- For GPU baselines: NVIDIA RTX 3090, A100-SXM4, H100-PCIe

High-Performance Results of X-TED

- 3 baselines:
  - Basic TED (1989)
  - State-of-the-art sequential solution (AP-TED, 2016)
  - State-of-the-art multi-core solution (MC-TED, 2020)

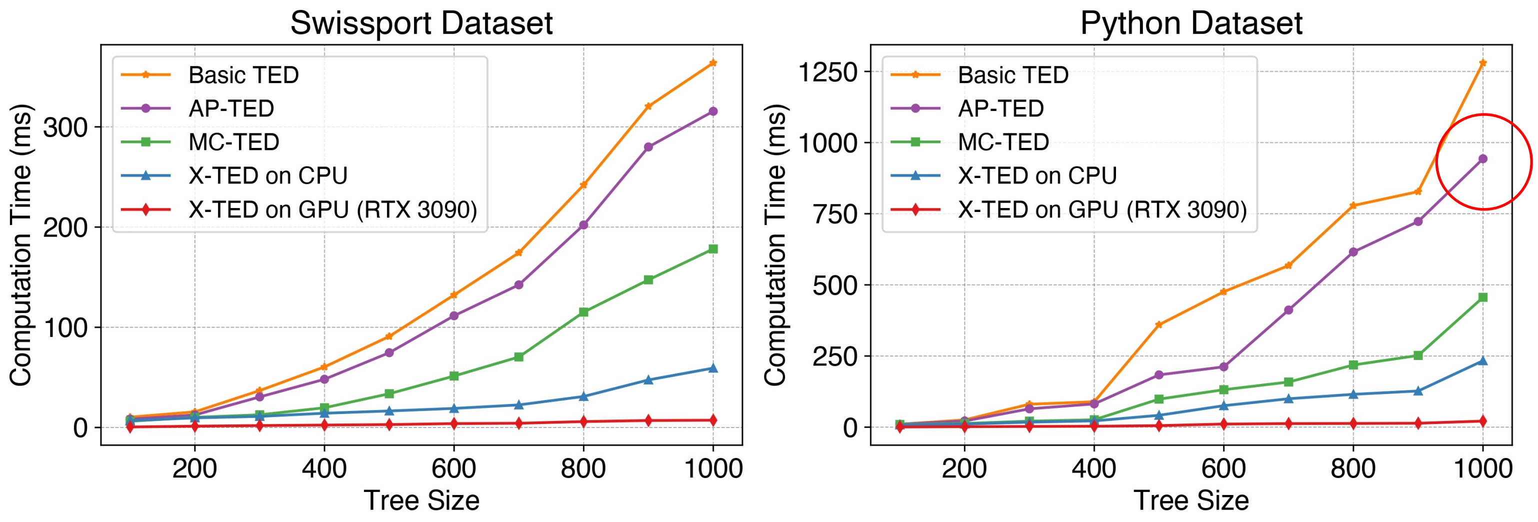
High-Performance Results of X-TED

- 3 baselines:
  - Basic TED (1989)
  - State-of-the-art sequential solution (AP-TED, 2016)
  - State-of-the-art multi-core solution (MC-TED, 2020)



High-Performance Results of X-TED

- 3 baselines:
  - Basic TED (1989)
  - State-of-the-art sequential solution (AP-TED, 2016)
  - State-of-the-art multi-core solution (MC-TED, 2020)



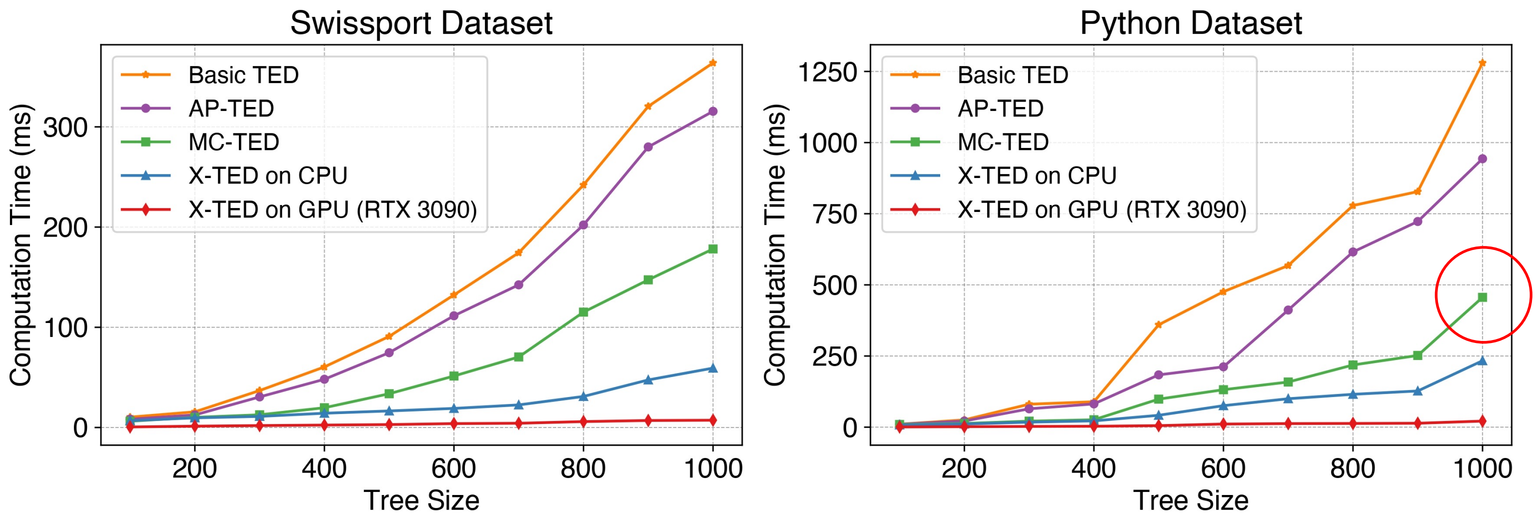
- Speedup over AP-TED

  X-TED (CPU): 4.8x

  X-TED (GPU): 42x

High-Performance Results of X-TED

- 3 baselines:
    - Basic TED (1989)
    - State-of-the-art sequential solution (AP-TED, 2016)
    - State-of-the-art multi-core solution (MC-TED, 2020)



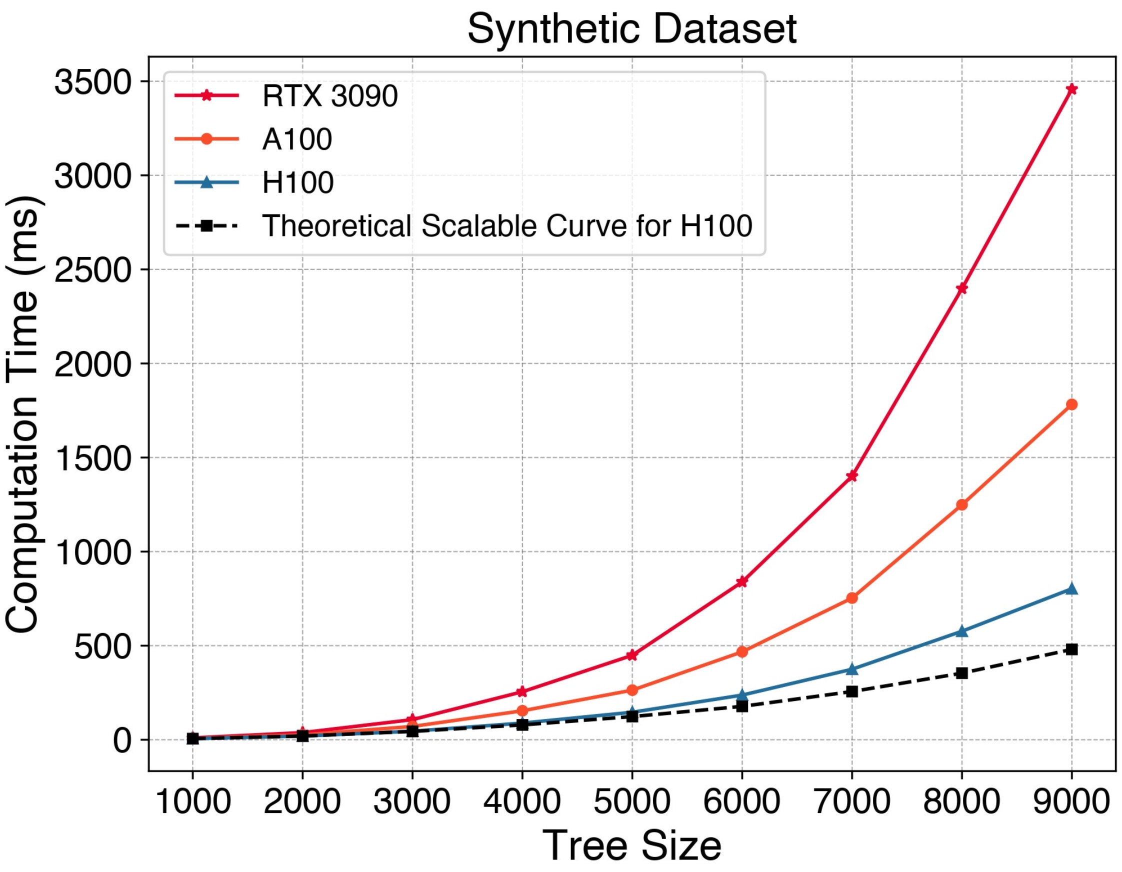
- Speedup over MC-TED

  X-TED (CPU): 3.8x

  X-TED (GPU): 31x

High Scalability of X-TED

- Tree size: 1000 nodes → 9000 nodes



Conclusion

Conclusion

- X-TED

  - A massive parallel computation framework for TED

  The ***best*** parallel TED solution so far

Conclusion

- X-TED

  - A massive parallel computation framework for TED

  The ***best*** parallel TED solution so far

- Preprocessing ***enables*** massive TED parallel processing

Conclusion

- X-TED
  - A massive parallel computation framework for TED

  The ***best*** parallel TED solution so far

- Preprocessing ***enables*** massive TED parallel processing

- Dynamic parallel strategy ***adaptively*** utilizes GPU resources

Conclusion

- X-TED

  - A massive parallel computation framework for TED

  The ***best*** parallel TED solution so far

- Preprocessing ***enables*** massive TED parallel processing

- Dynamic parallel strategy ***adaptively*** utilizes GPU resources

Thank You!

Email: fan.1090@osu.edu

* Project Website and Open-Source Code: https://github.com/Davis-Fan/X-TED

Image Reference

- Image in Page 13: Mohebbi, M., Razavi, S.N. & Balafar, M.A. Computing semantic similarity of texts based on deep graph learning with ability to use semantic role label information. Sci Rep 12, 14777 (2022). https://doi.org/10.1038/s41598-022-19259-5

- Image in Page 14: Chartier M, Chénard T, Barker J, Najmanovich R. Kinome Render: a stand-alone and web-accessible tool to annotate the human protein kinome tree. PeerJ. 2013 Aug 8;1:e126. doi: 10.7717/peerj.126. PMID: 23940838; PMCID: PMC3740139.

- Image in Page 15: Mate Kukri. 2022. Syntax searching C/C++ with Clang AST. Retrieved December 22, 2022 from https://blog.trailofbits.com/2022/12/22/syntax-searching-c-c-clang-ast/.

- Image in Page 16: Xutao Li, Yunming Ye, Mark Junjie Li, Michael K. Ng, On cluster tree for nested and multi-density data clustering, Pattern Recognition, Volume 43, Issue 9, 2010, Pages 3130-3143, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2010.03.020.