A Linear-Time Algorithm for Optimal Tree Sibling Partitioning and Approximation Algorithms in Natix

VLDB 2006

Carl-Christian Kanne

University of Mannheim

kanne@db.informatik.uni-mannheim.de

Guido Moerkotte

University of Mannheim

moer@db.informatik.uni-mannheim.de

UNIVERSITÄT MANNHEIM

XML Document Tree Storage

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

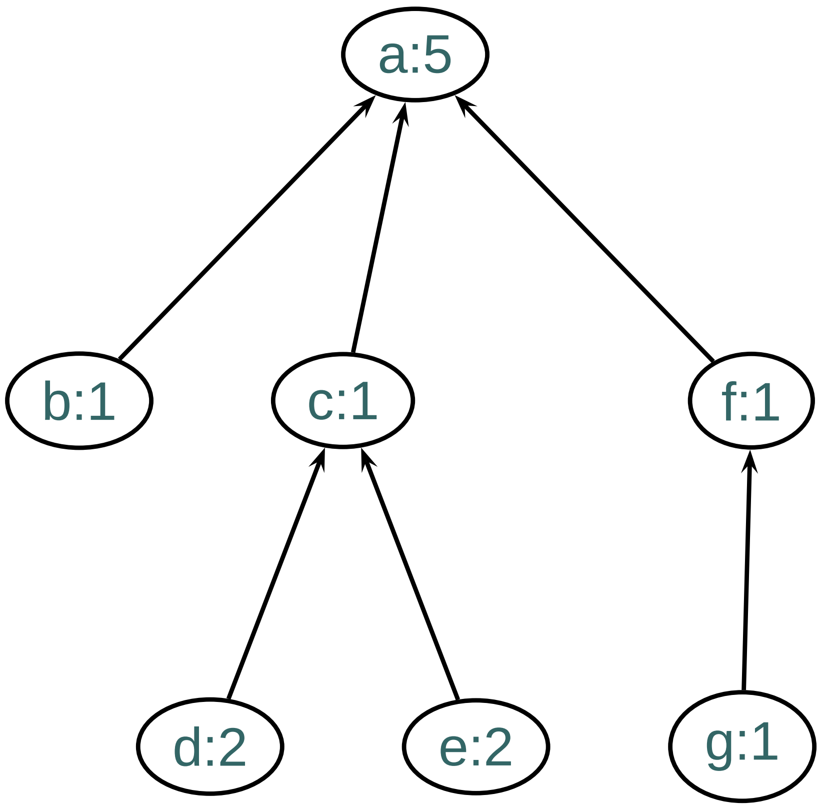
Flat Tree Partitioning

Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion



UNIVERSITÄT MANNHEIM

XML Document Tree Storage

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

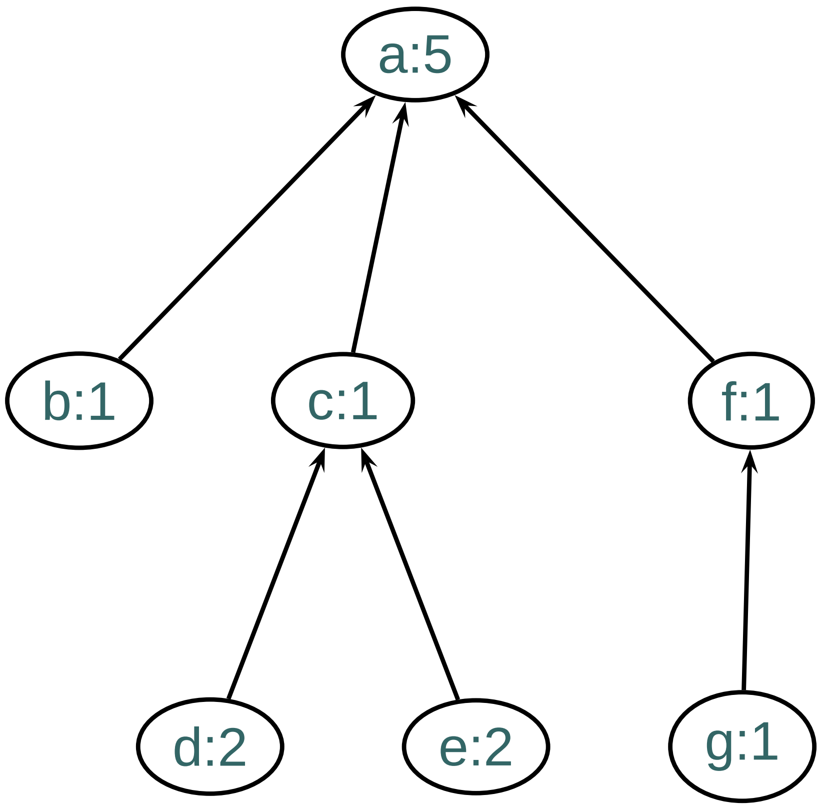
Flat Tree Partitioning

Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion



Weight Limit: 5

UNIVERSITÄT MANNHEIM

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

Flat Tree Partitioning

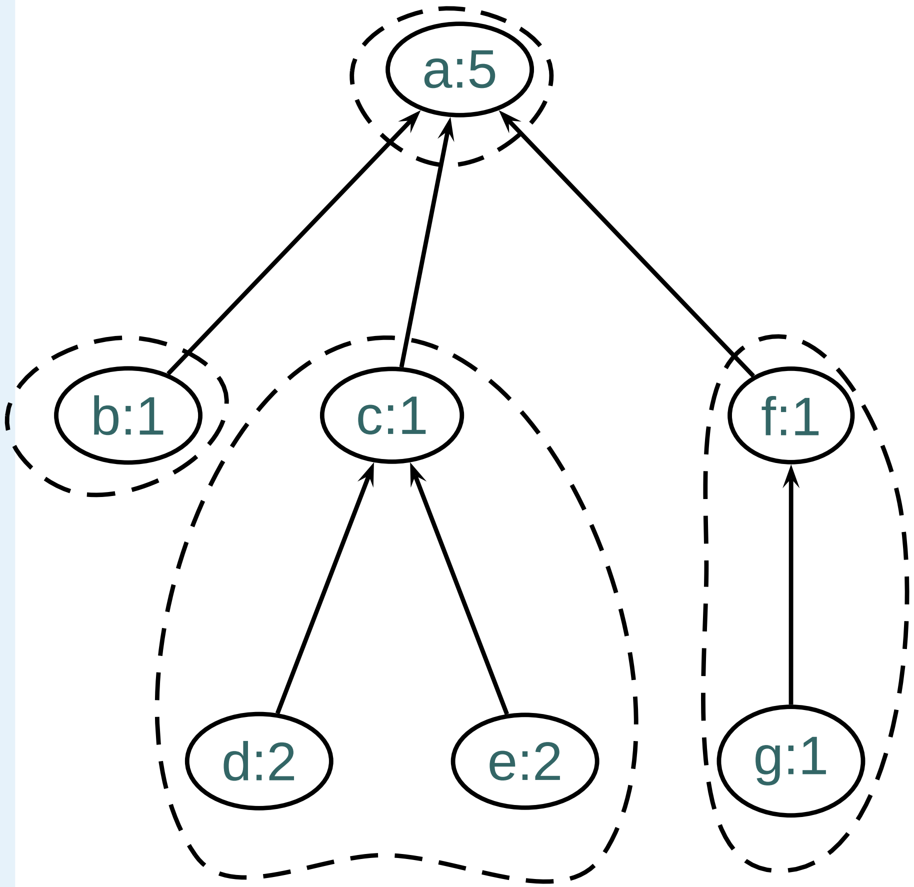
Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

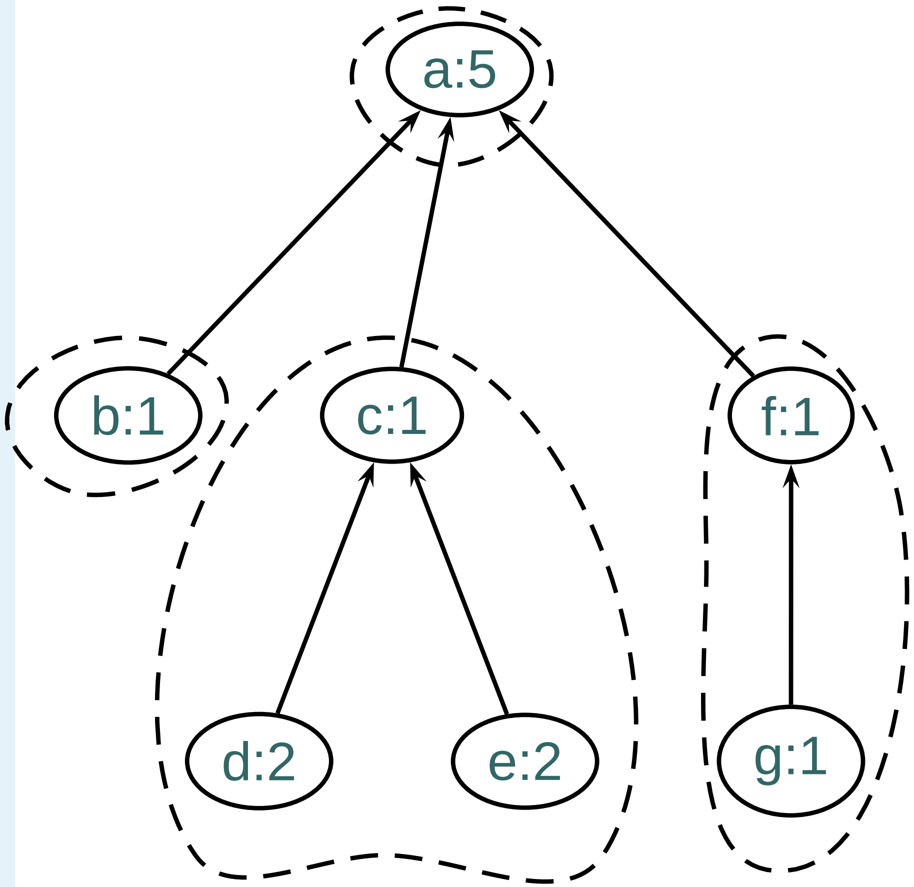
Tree Partitioning



Weight Limit: 5

XML Document Tree Storage

Minimal Tree Partitioning



Weight Limit: 5

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

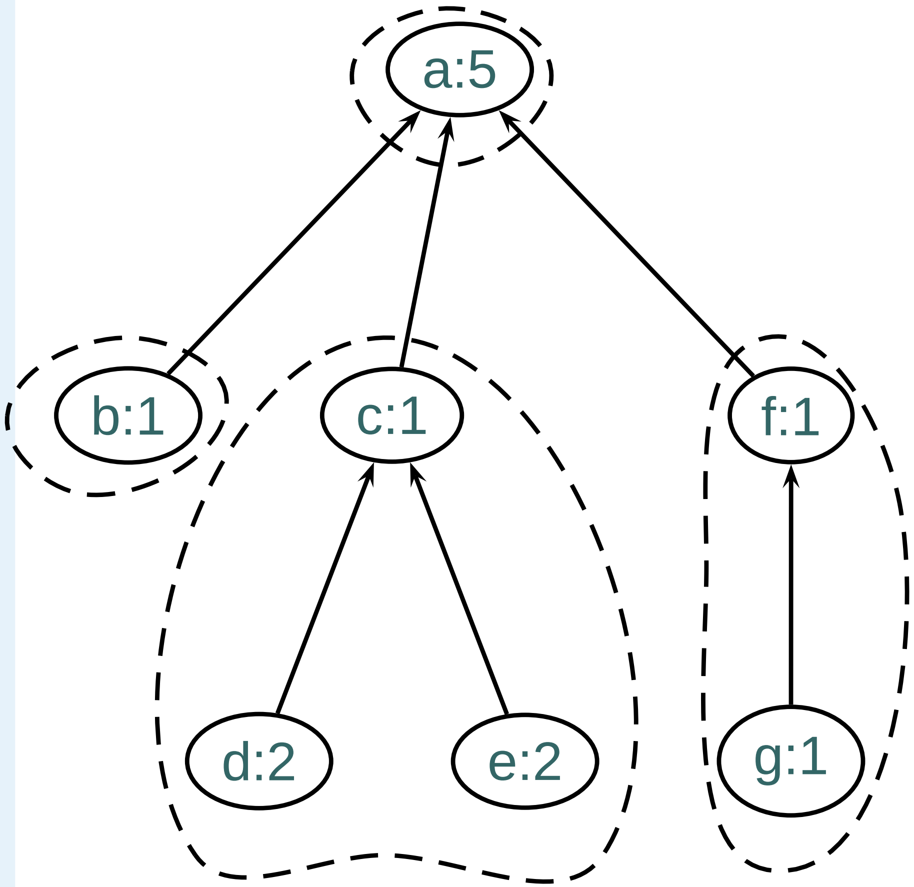
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

UNIVERSITÄT MANNHEIM

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

Minimal Tree Partitioning



Single-edge partitions: 4

Weight Limit: 5

XML Document Tree Storage

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

Flat Tree Partitioning

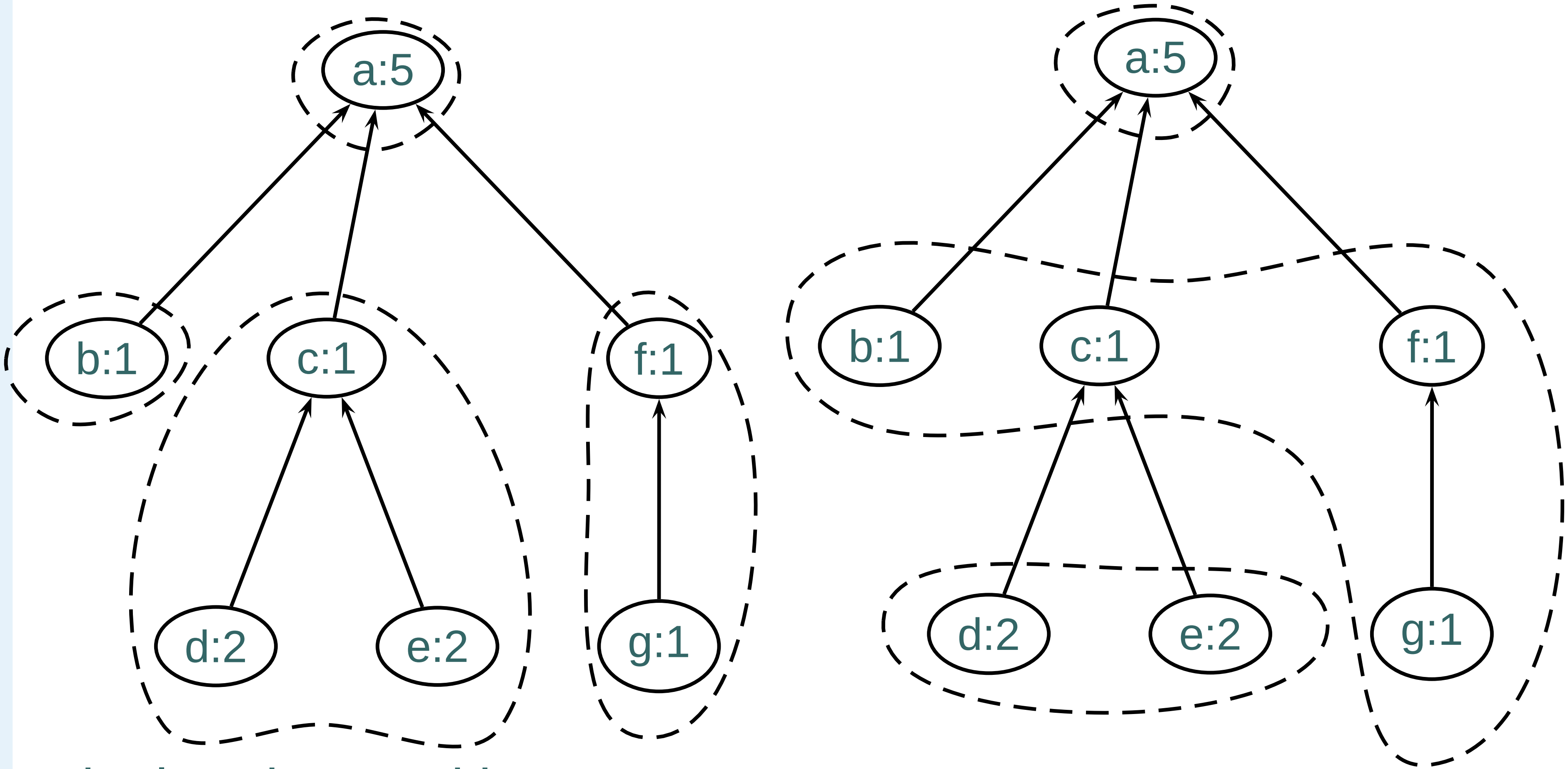
Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

Minimal Tree Partitioning



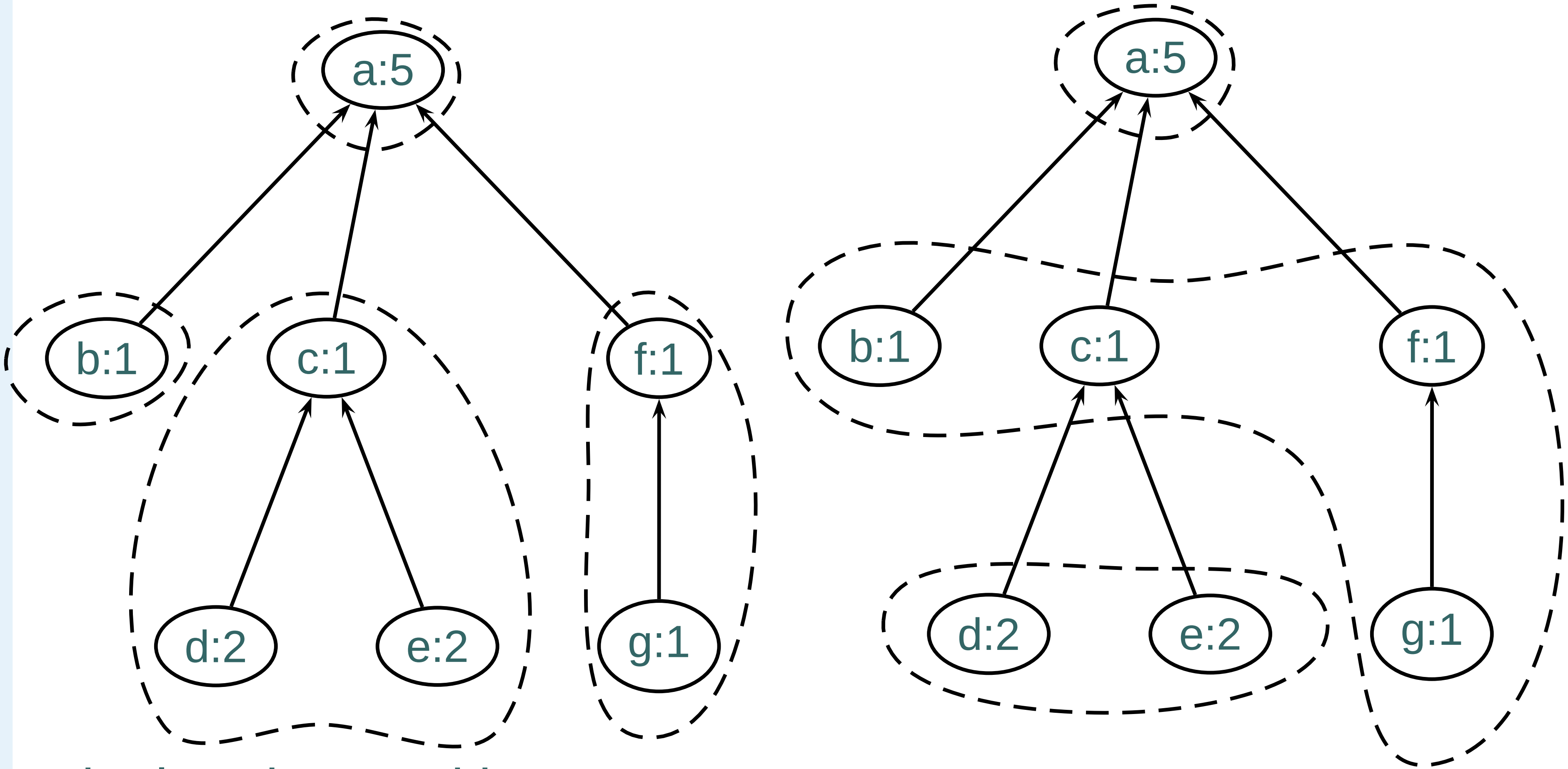
Single-edge partitions: 4

Weight Limit: 5

XML Document Tree Storage

Introduction
XML Document Tree Storage
Definition: Tree Sibling Partitioning
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

Minimal Tree Partitioning



Single-edge partitions: 4

Sibling partitions: 3

Weight Limit: 5

Definition: Tree Sibling Partitioning

Introduction
- XML Document Tree Storage
- Definition: Tree Sibling Partitioning

Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

- P set of sibling intervals (l, r)
  - 1 interval defines 1 partition
  - may cut more than one edge
- Single-edge partitioning: Intervals with $(l = r)$
- *Weight of partition*: Sum of node weights
- *Feasible partitioning*, given constant K
  All partition weights $\leq K$
- *Optimal* tree sibling partitioning P
  - Feasible
  - Minimal
  - (Lean)

UNIVERSITÄT MANNHEIM

Flat Tree Partitioning

Introduction

Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

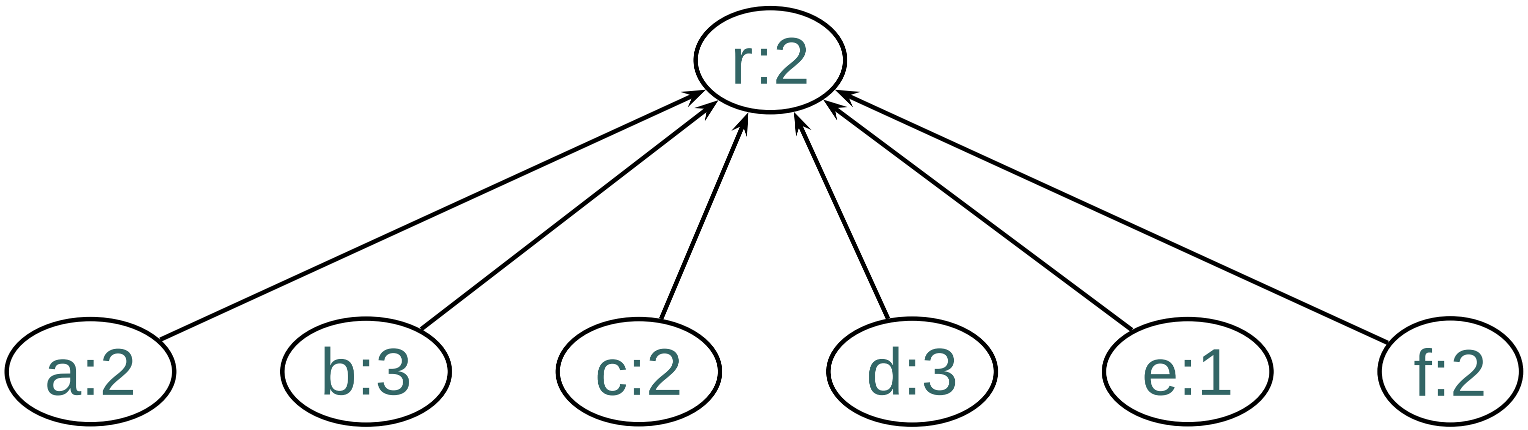
Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

Weight Limit $K = 5$:



UNIVERSITÄT MANNHEIM

Optimal Substructure (Flat Trees)

Introduction

Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

- Alternatives for last leaf v:
  1. Put into partition with parent
  2. Put into own partition (v, v)
  3. Put into partition with up to $K - 1$ preceding siblings

Example

Introduction

Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

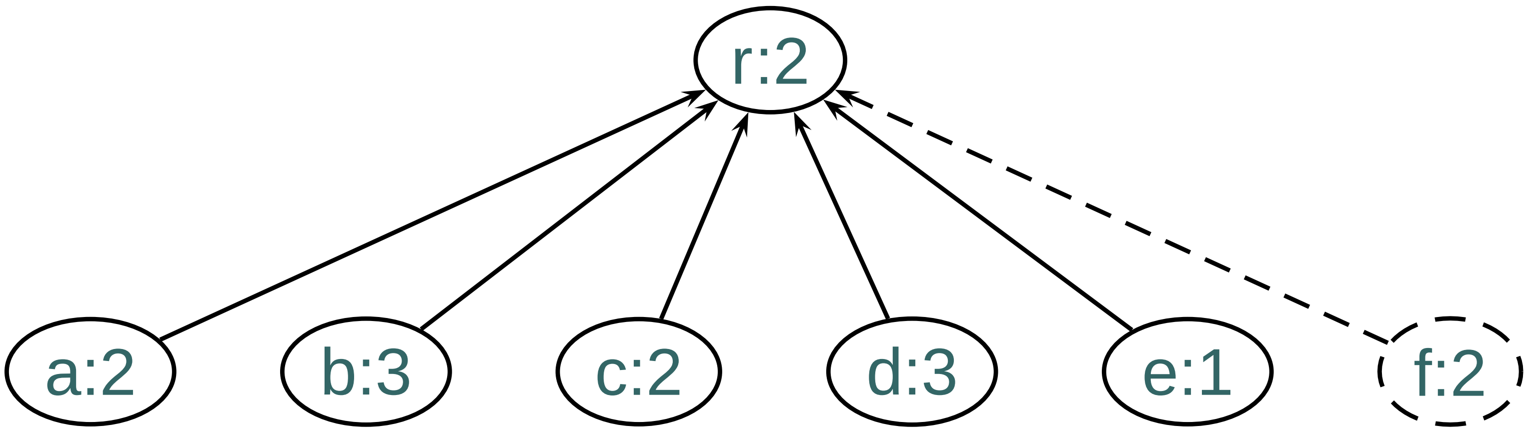
Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

Weight Limit $K = 5$:

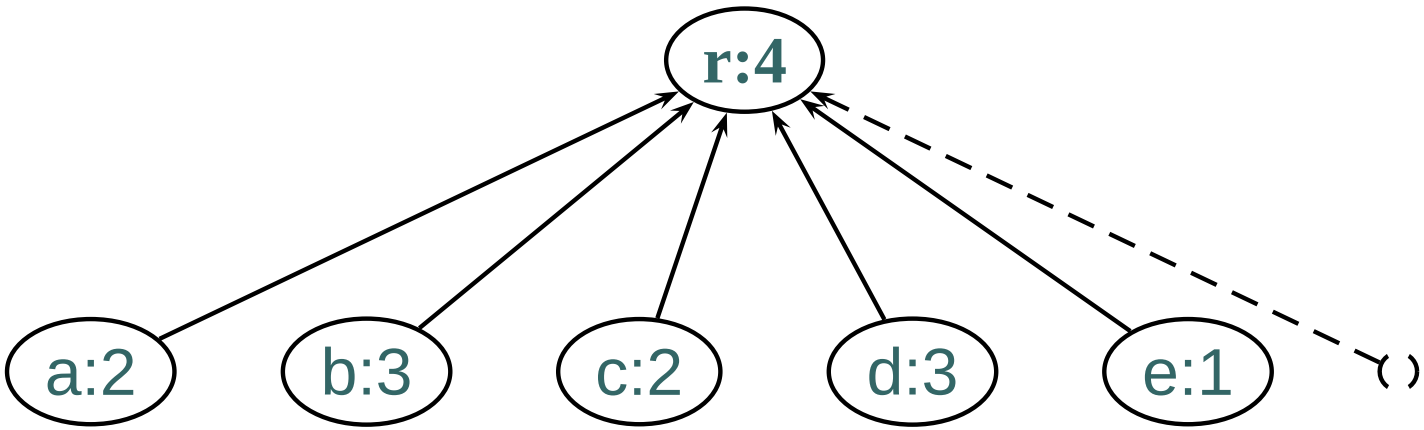


Parent Partition

Introduction
Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:

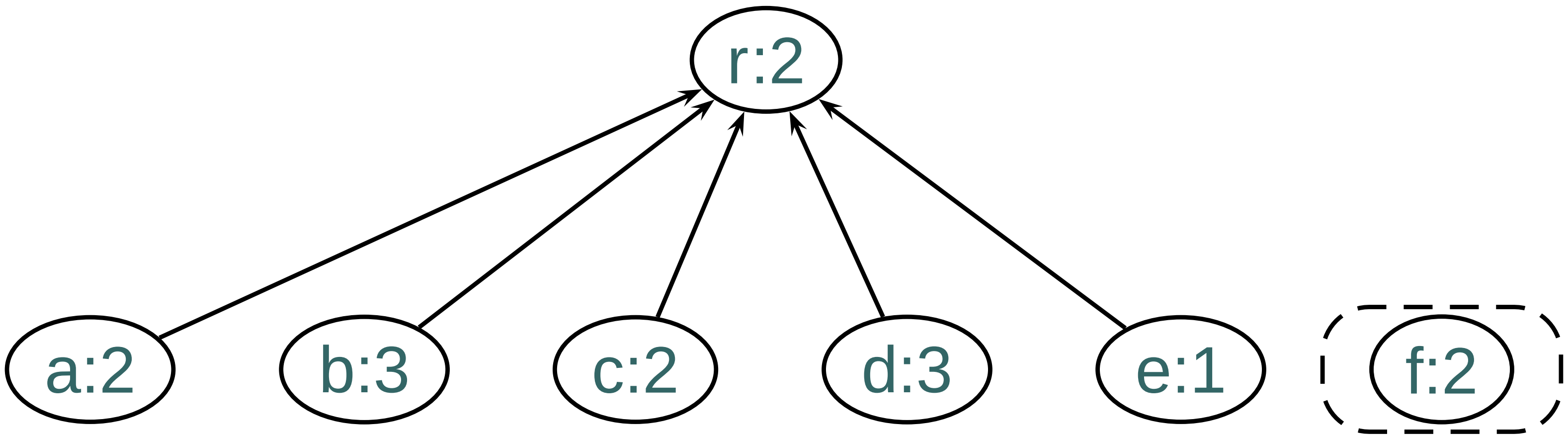


Own Partition

Introduction
Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



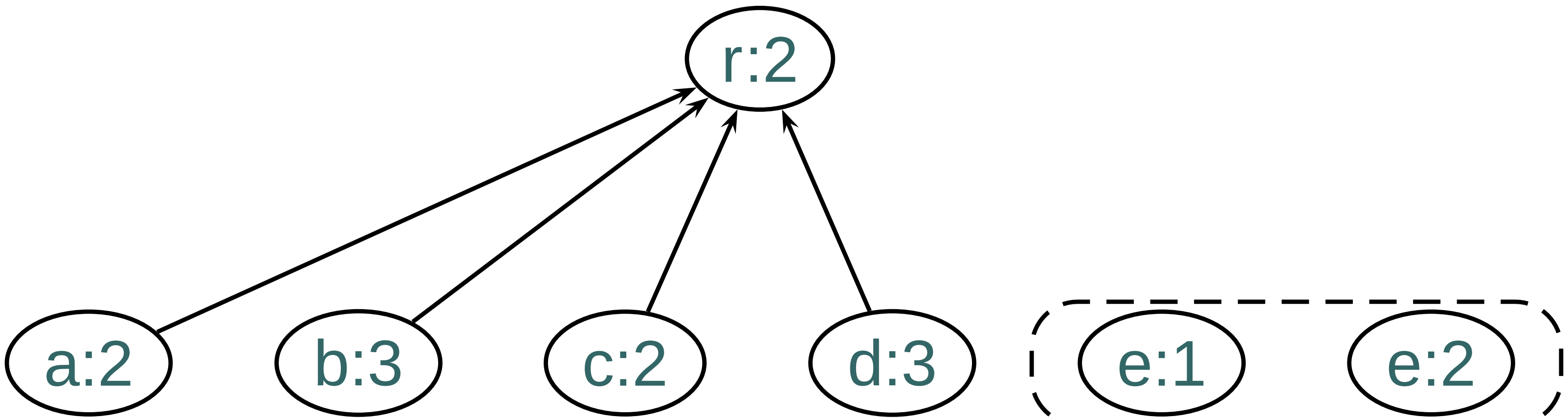
UNIVERSITÄT MANNHEIM

With Left Sibling

Introduction
Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



Algorithm FDW

Introduction

Flat Tree Partitioning
- Flat Tree Partitioning
- Optimal Substructure (Flat Trees)
- Example
- Parent Partition
- Own Partition
- With Left Sibling
- Algorithm FDW

Deep Tree Partitioning

Approximation Algorithms

Evaluation

Conclusion

- **F**lat trees, **D**ynamic programming, **W**idth
- Dynamic programming table
  D_j^s Optimal partitioning for j leaves, root weight s
- Overall solution is $D_n^{weight(root)}$
- Each entry D_j^s computed in $K+1$ steps
  - Based on entries $D_{j'}^{s'}$ with $j < j'$ and $s' \geq s$
- Overall run-time: $O(nK^2)$

UNIVERSITÄT MANNHEIM

Deep Trees: Bottom-Up Processing

Introduction

Flat Tree Partitioning

Deep Tree Partitioning

- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms

Evaluation

Conclusion

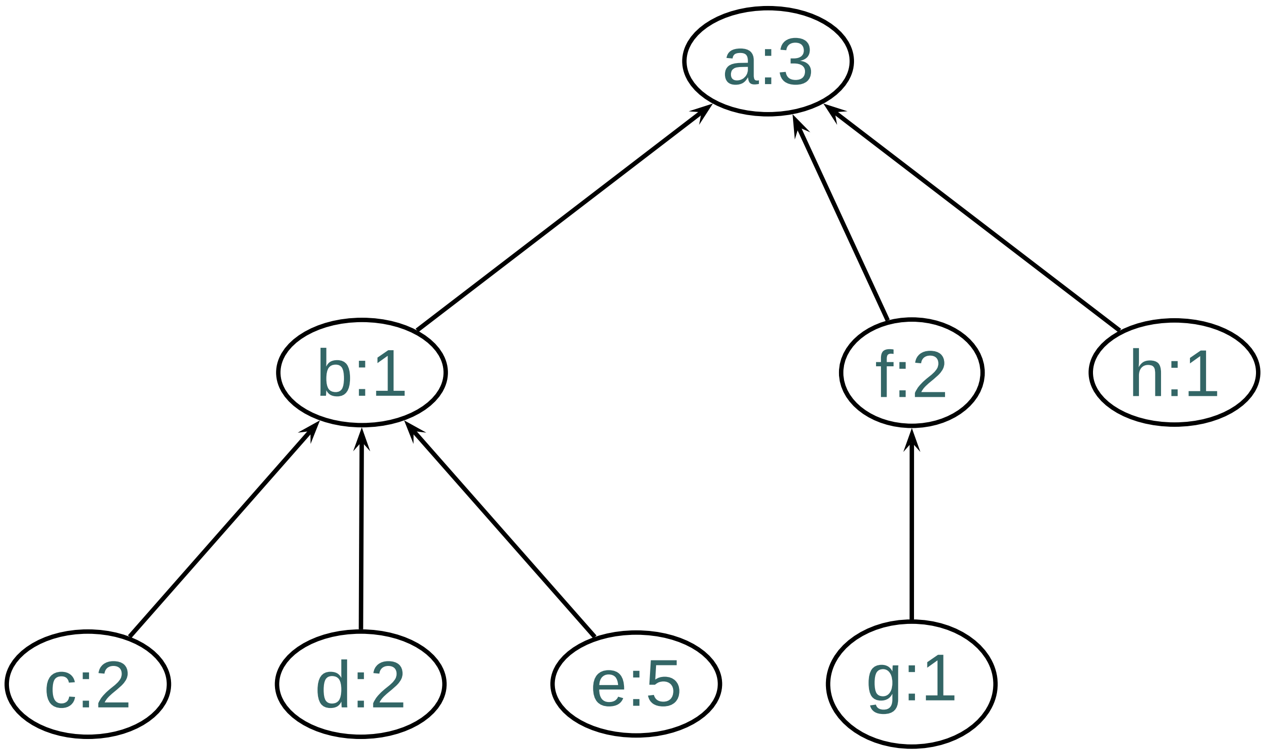
- Extend to deep trees
  - Process deep trees bottom-up
  - Process flat subtrees with FDW
  - Collapse processed trees into single node

$\Longrightarrow$ Algorithm GHDW
(**G**reedy **H**eight, **D**ynamic **W**idth)

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



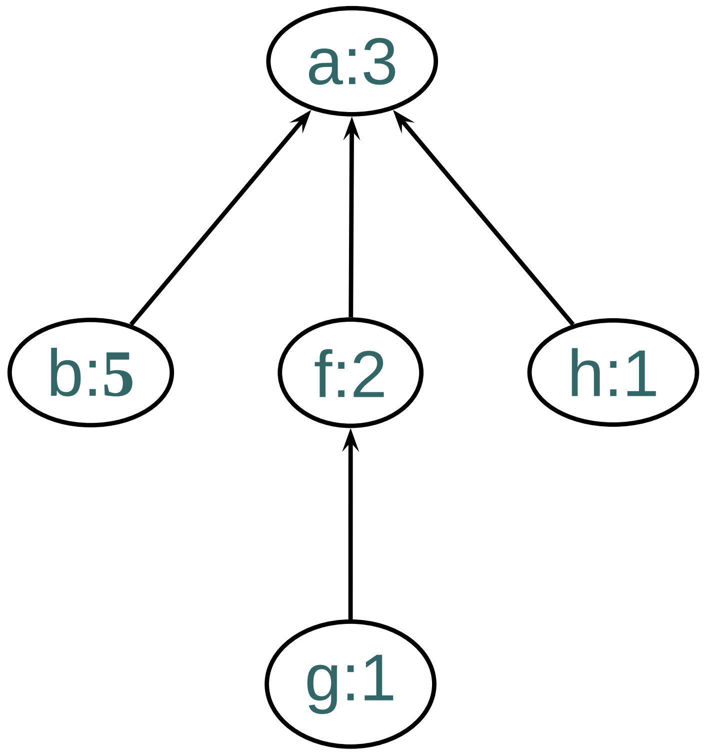
Collapse $\mathbf{b}$ usign partitioning: (e, e)

Collapsing the Tree

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



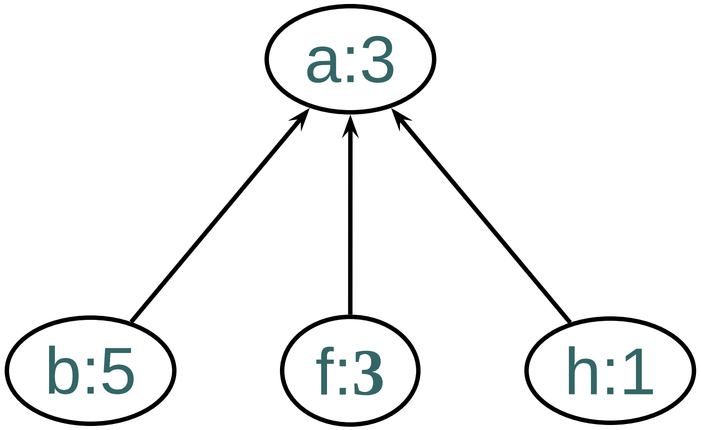
Collapse **f**, $\{(e, e)\}$

Flat Tree

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



Now use FDW on root

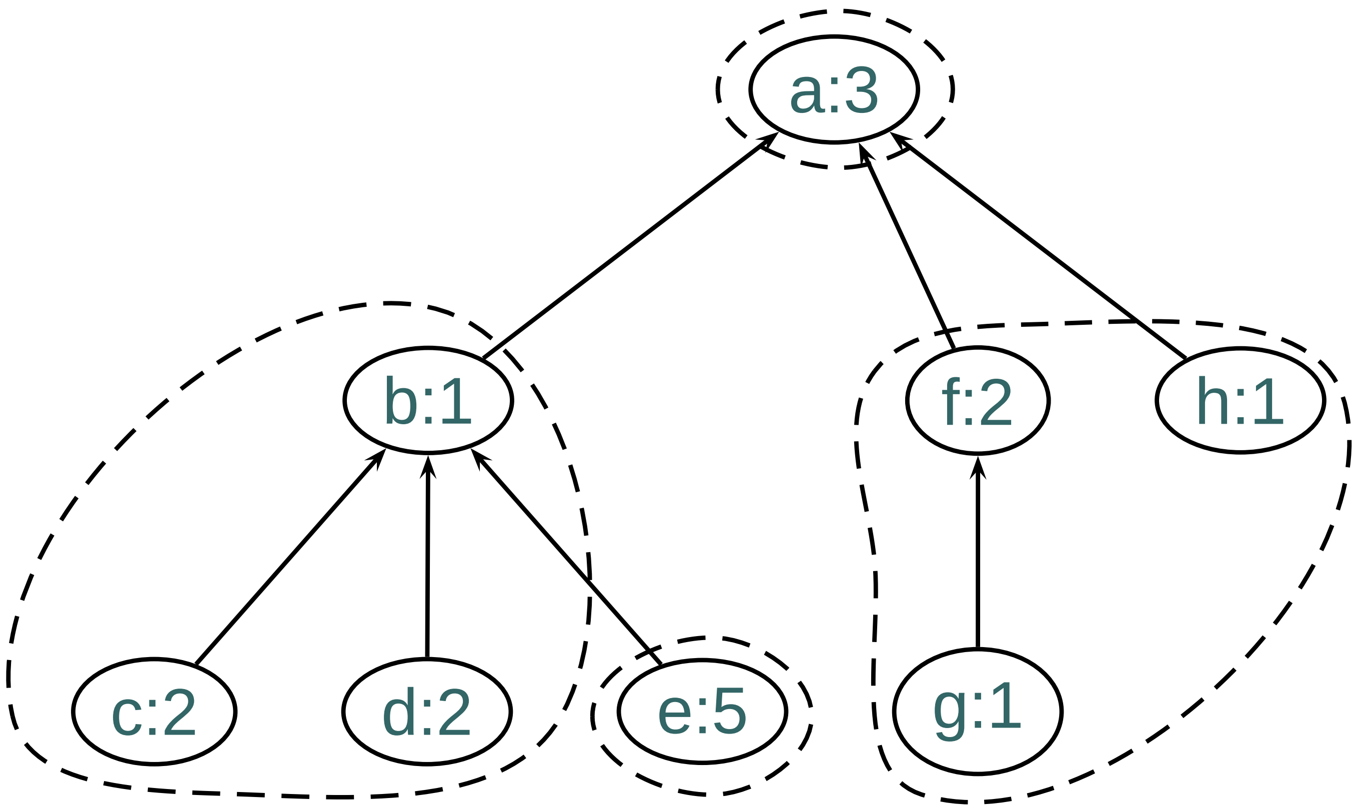
$$\{(e, e)\}$$

GHDW Result

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

Weight Limit $K = 5$:



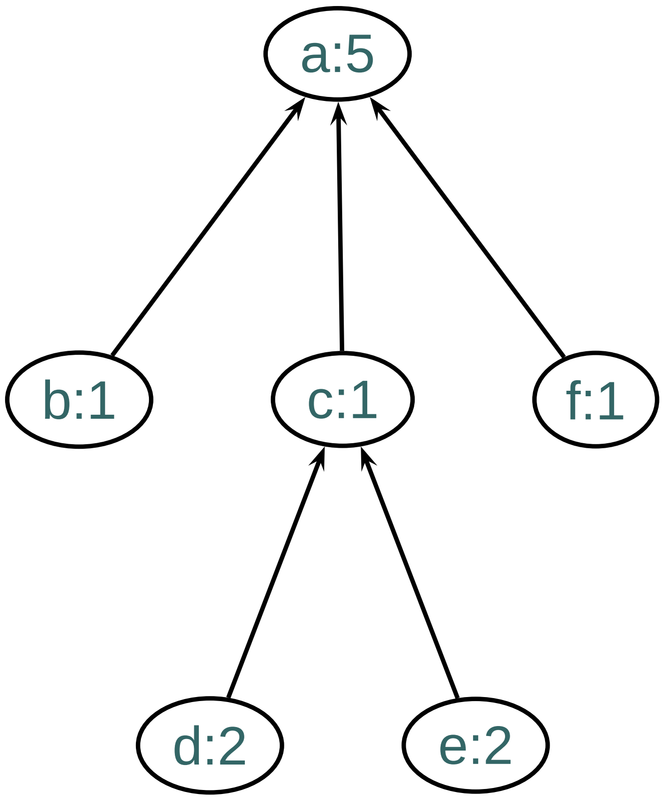
Final result of GHDW Algorithm:

$\{(e,e),(b,b),(f,h),(a,a)\}$

UNIVERSITÄT MANNHEIM

Problematic Case : GHDW

GHDW Result $K = 5$:



Problematic Case : GHDW

Introduction

Flat Tree Partitioning

Deep Tree Partitioning

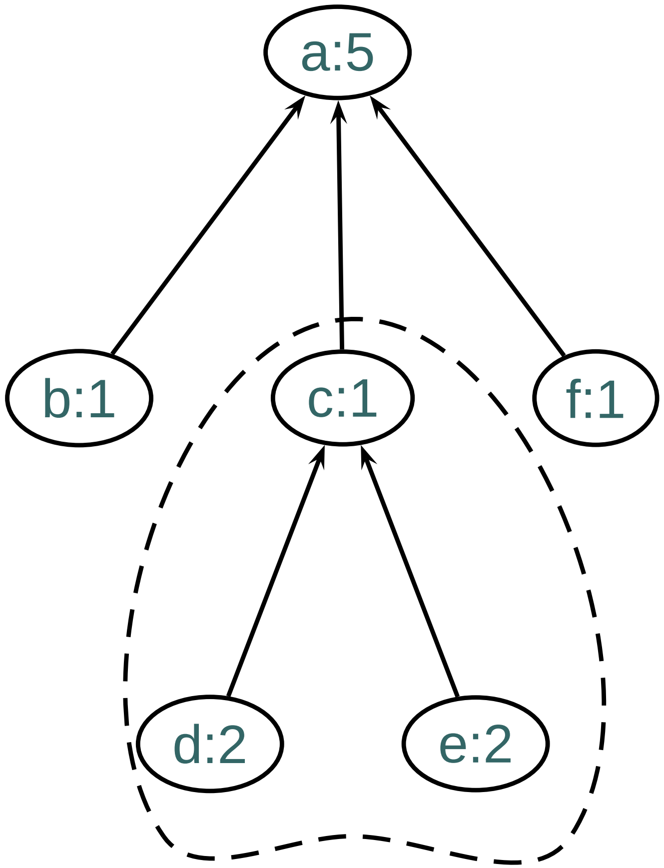
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms

Evaluation

Conclusion

GHDW Result $K = 5$:



UNIVERSITÄT MANNHEIM

Problematic Case : GHDW

Introduction

Flat Tree Partitioning

Deep Tree Partitioning

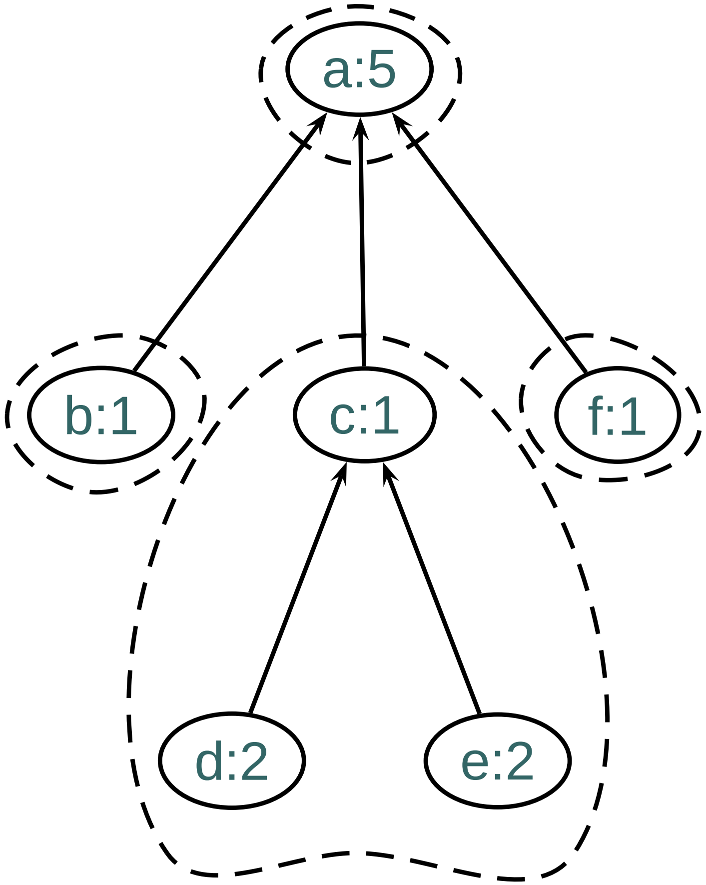
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- **Problematic Case : GHDW**
- Optimal Substructure (Deep Trees)

Approximation Algorithms

Evaluation

Conclusion

GHDW Result $K = 5$:
4 Partitions



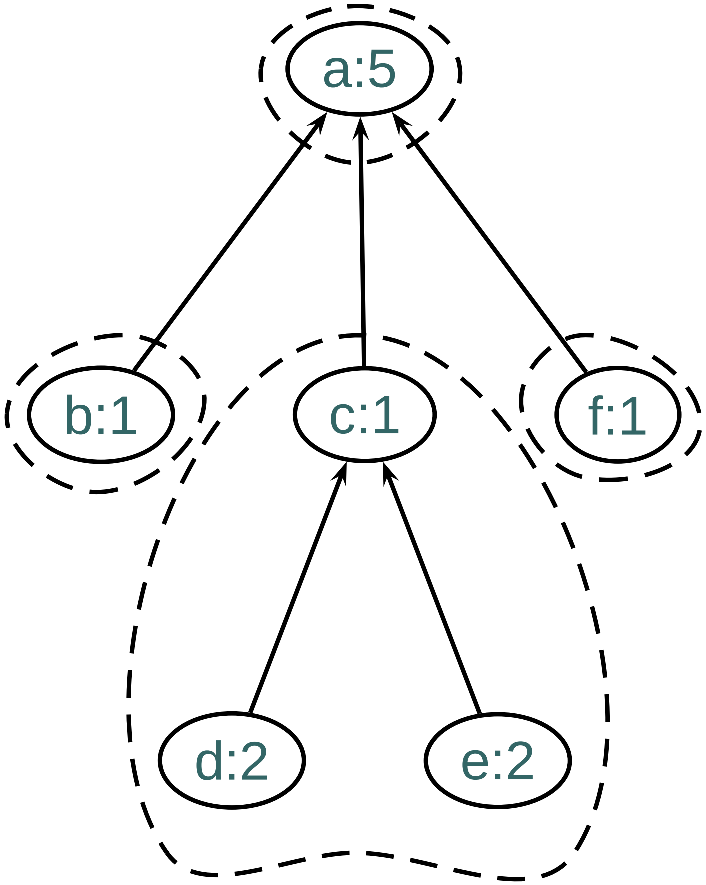
UNIVERSITÄT MANNHEIM

Problematic Case : GHDW

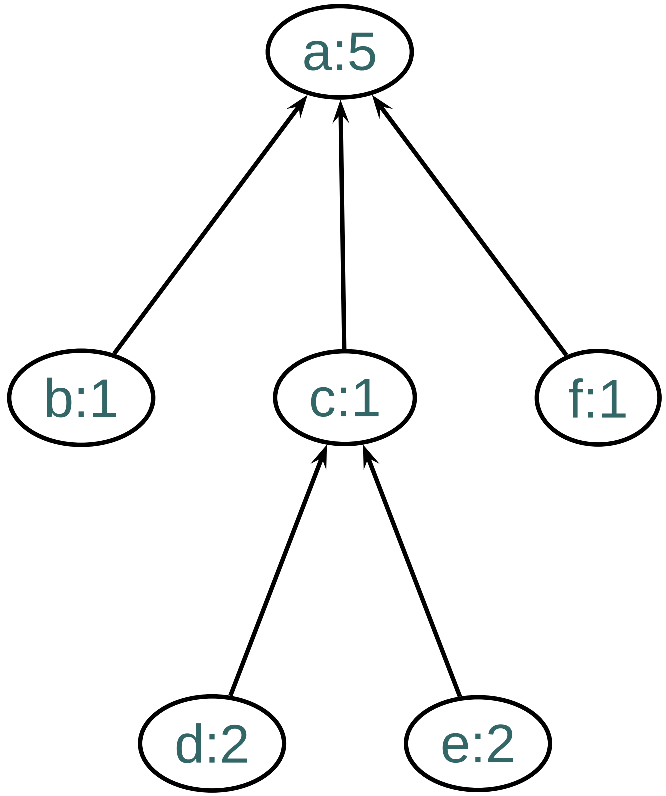
Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

GHDW Result $K = 5$: 4 Partitions



Optimal Result $K = 5$:



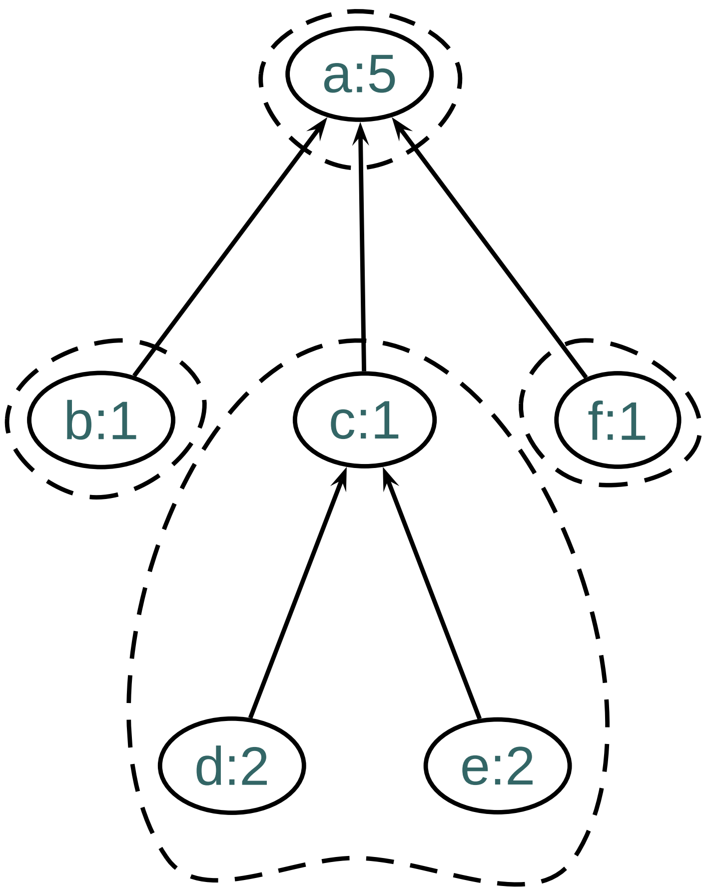
UNIVERSITÄT MANNHEIM

Problematic Case : GHDW

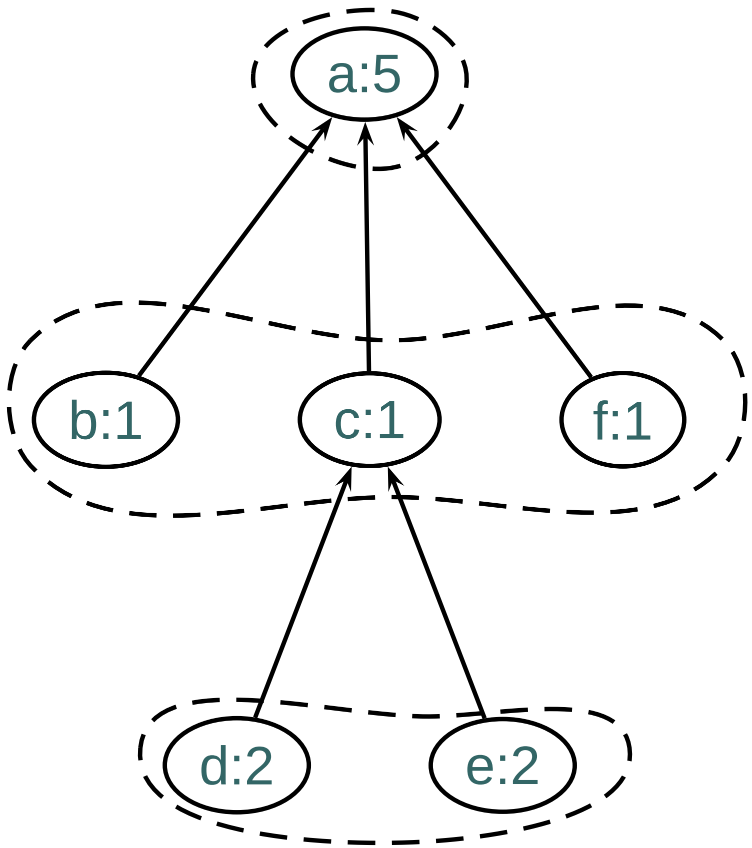
Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- **Problematic Case : GHDW**
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

GHDW Result $K = 5$:
4 Partitions



Optimal Result $K = 5$:
3 Partitions



Optimal Substructure (Deep Trees)

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
- Deep Trees: Bottom-Up Processing
- Bottom-Up Processing with GHDW
- Collapsing the Tree
- Flat Tree
- GHDW Result
- Problematic Case : GHDW
- Optimal Substructure (Deep Trees)

Approximation Algorithms
Evaluation
Conclusion

- For each subtree, global optimum contains *one of*
  - locally optimal solution
  - locally nearly optimal solution (+1 interval)
- Algorithm DHW (Dynamic Height&Width)
  - integrate choice into dynamic programming
  - $O(nK^3)$

Alternative Algorithms

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
- Alternative Algorithms
- Enhanced Kundu and Misra (EKM)

Evaluation
Conclusion

GHDW Always use optimal subtree partitionings

DFS Assign nodes to current partition in depth-first order
new partition if full or not connected

BFS As above, but with breadth-first search

KM Kundu and Misra (1977)
- While subtree weight $> K$:
  - Cut edge of heaviest son
- Optimal for single-edge partitions!

UNIVERSITÄT MANNHEIM

Enhanced Kundu and Misra (EKM)

Introduction

Flat Tree Partitioning

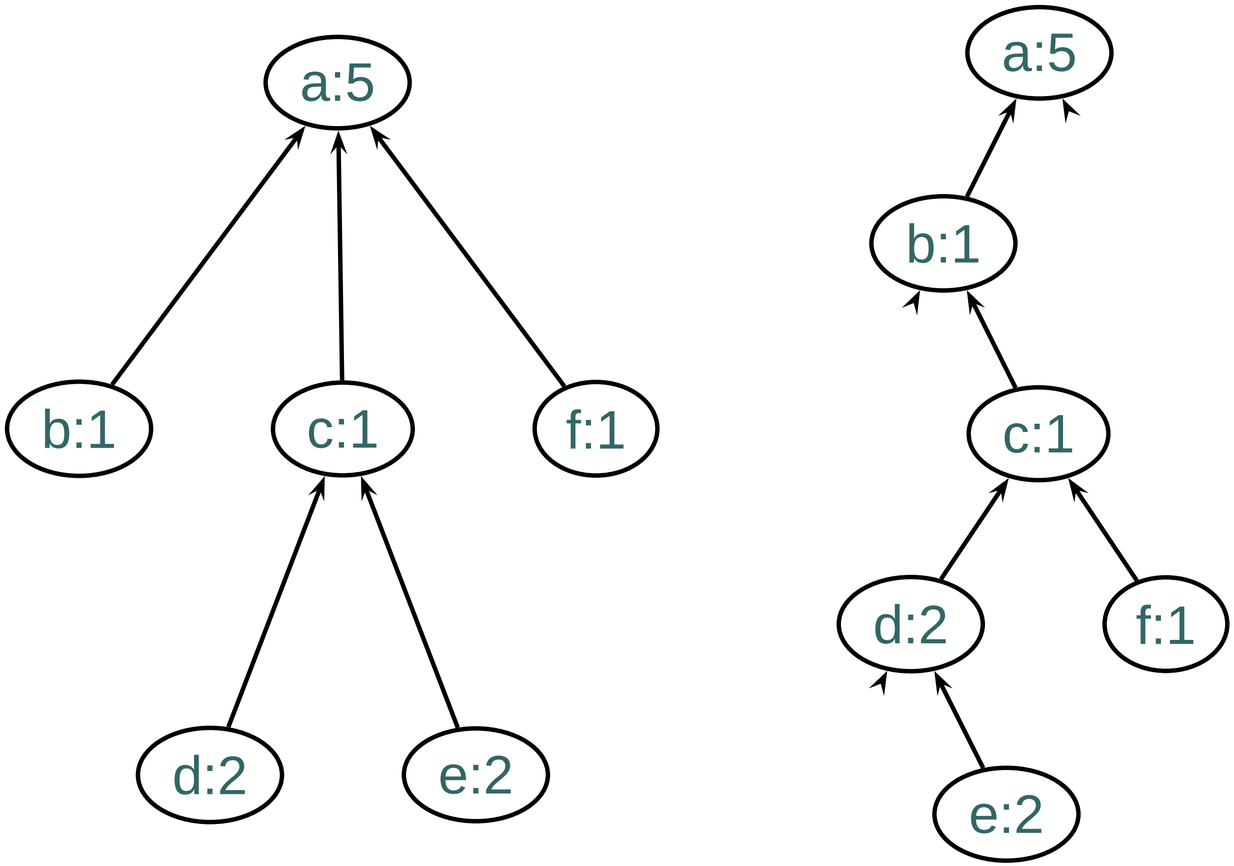
Deep Tree Partitioning

Approximation Algorithms
- Alternative Algorithms
- Enhanced Kundu and Misra (EKM)

Evaluation

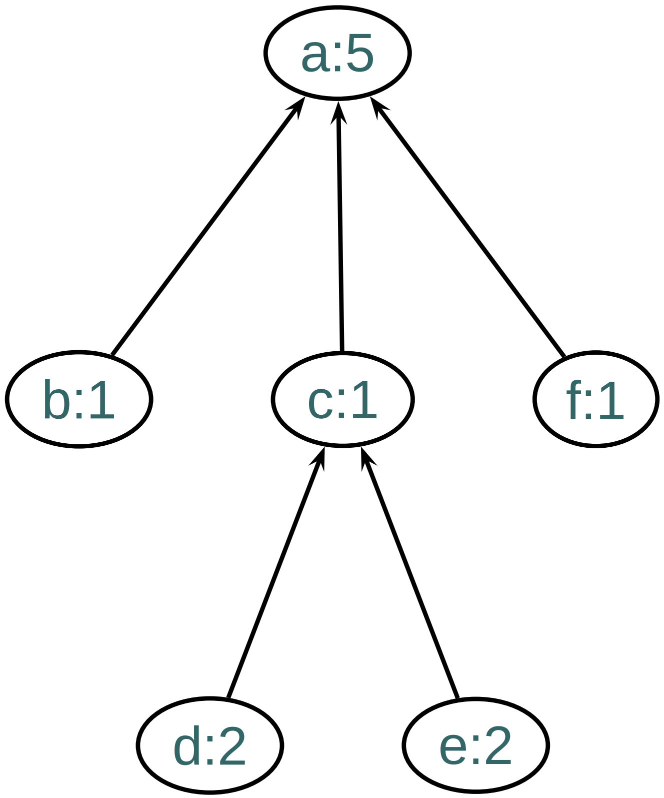
Conclusion

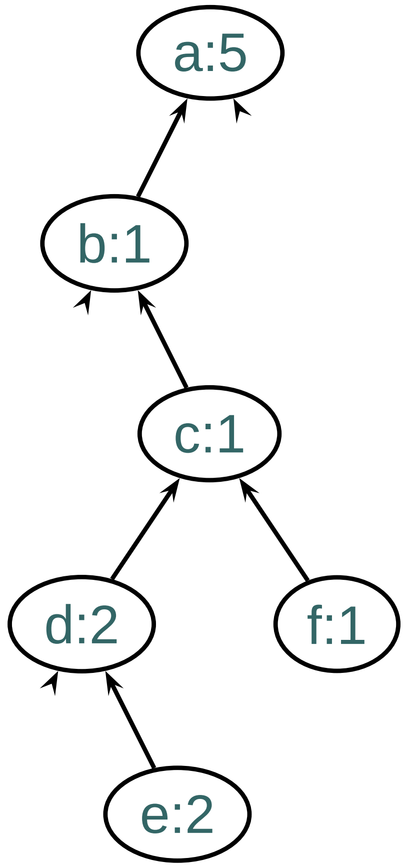
binary representation of n-ary tree



Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
- Alternative Algorithms
- Enhanced Kundu and Misra (EKM)
Evaluation
Conclusion

Run Kundu and Misra Algorithm on binary representation of n-ary tree!



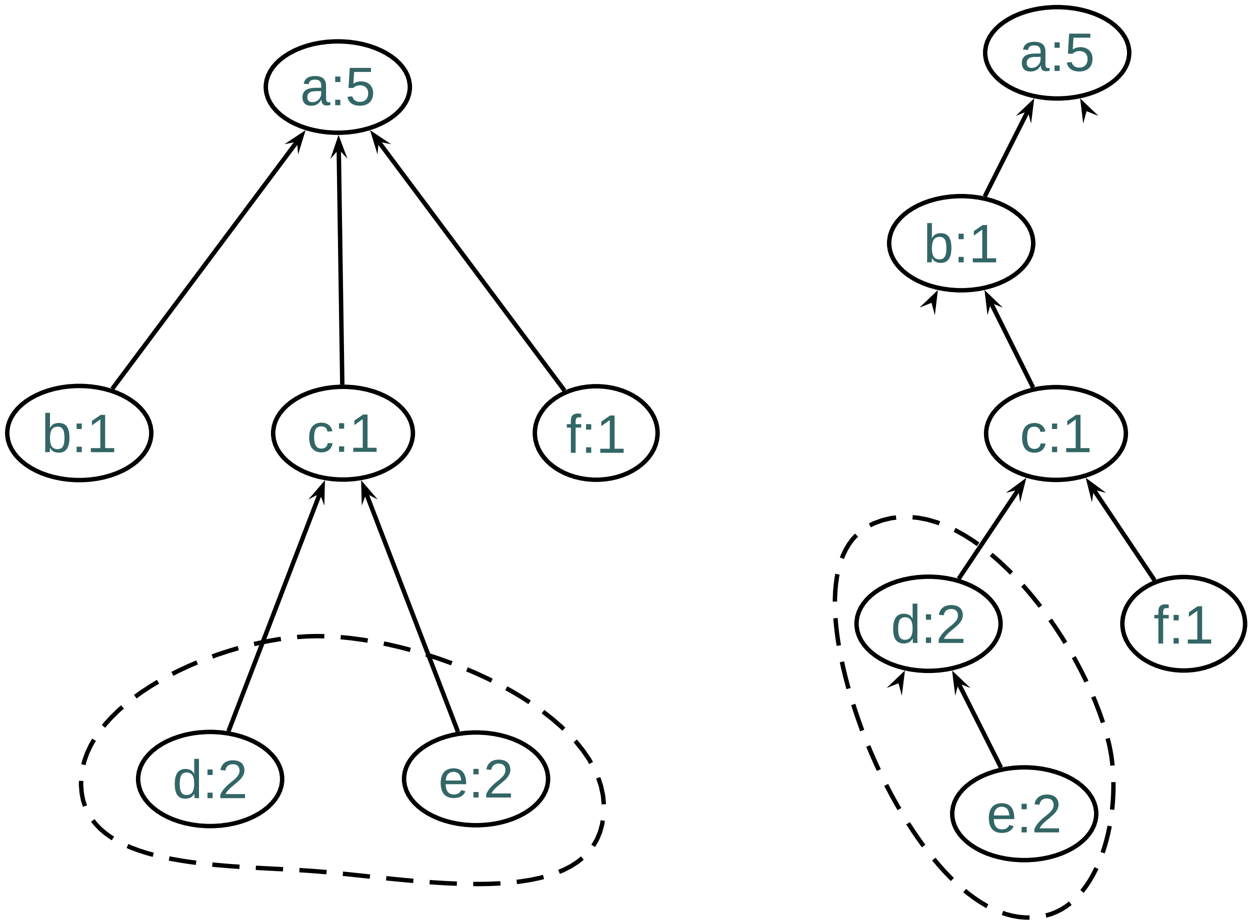


Enhanced Kundu and Misra (EKM)

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
- Alternative Algorithms
- Enhanced Kundu and Misra (EKM)

Evaluation
Conclusion

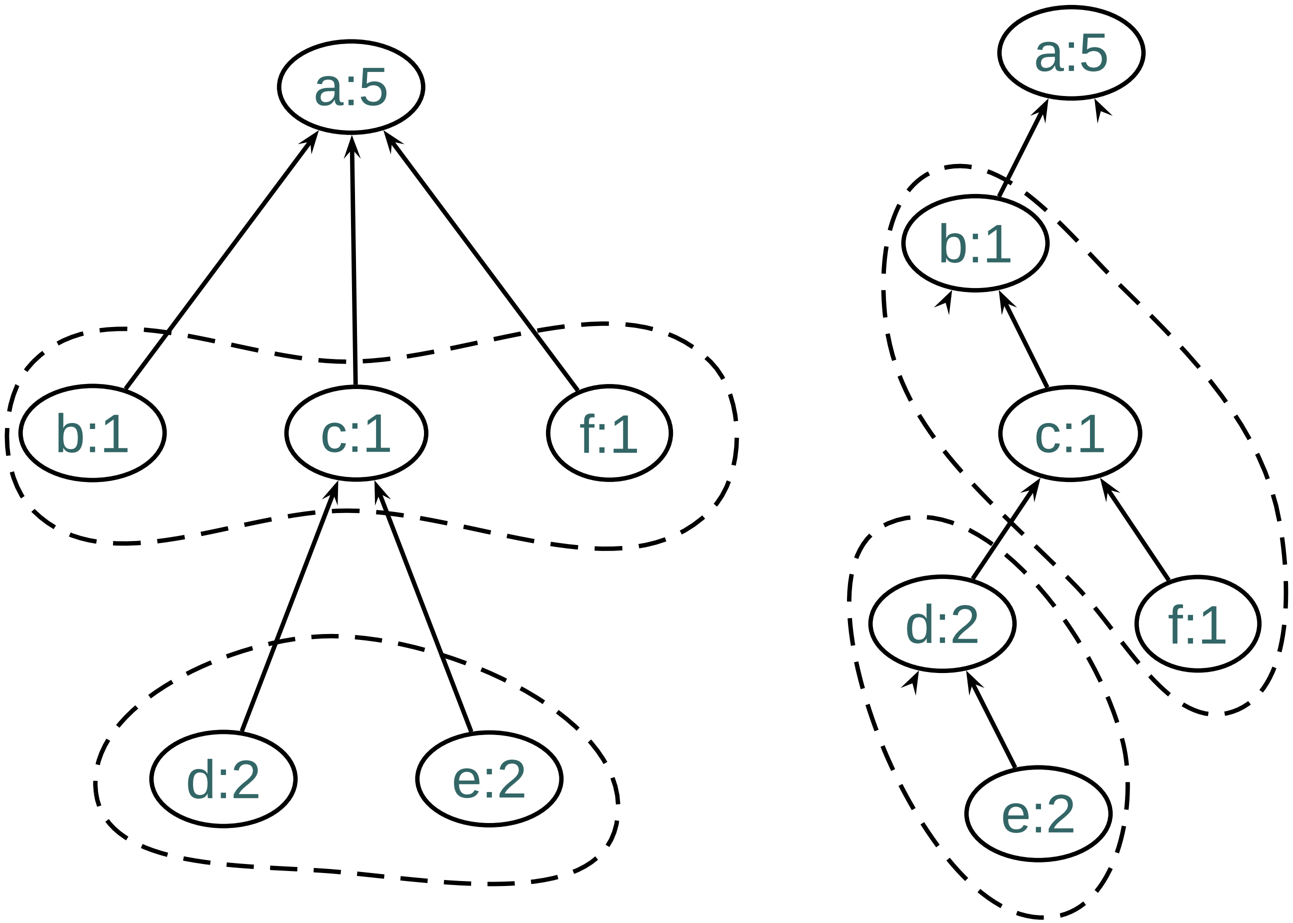
Run Kundu and Misra Algorithm on binary representation of n-ary tree!



Enhanced Kundu and Misra (EKM)

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
● Alternative Algorithms
● Enhanced Kundu and Misra (EKM)
Evaluation
Conclusion

Run Kundu and Misra Algorithm on binary representation of n-ary tree!



Enhanced Kundu and Misra (EKM)

Introduction

Flat Tree Partitioning

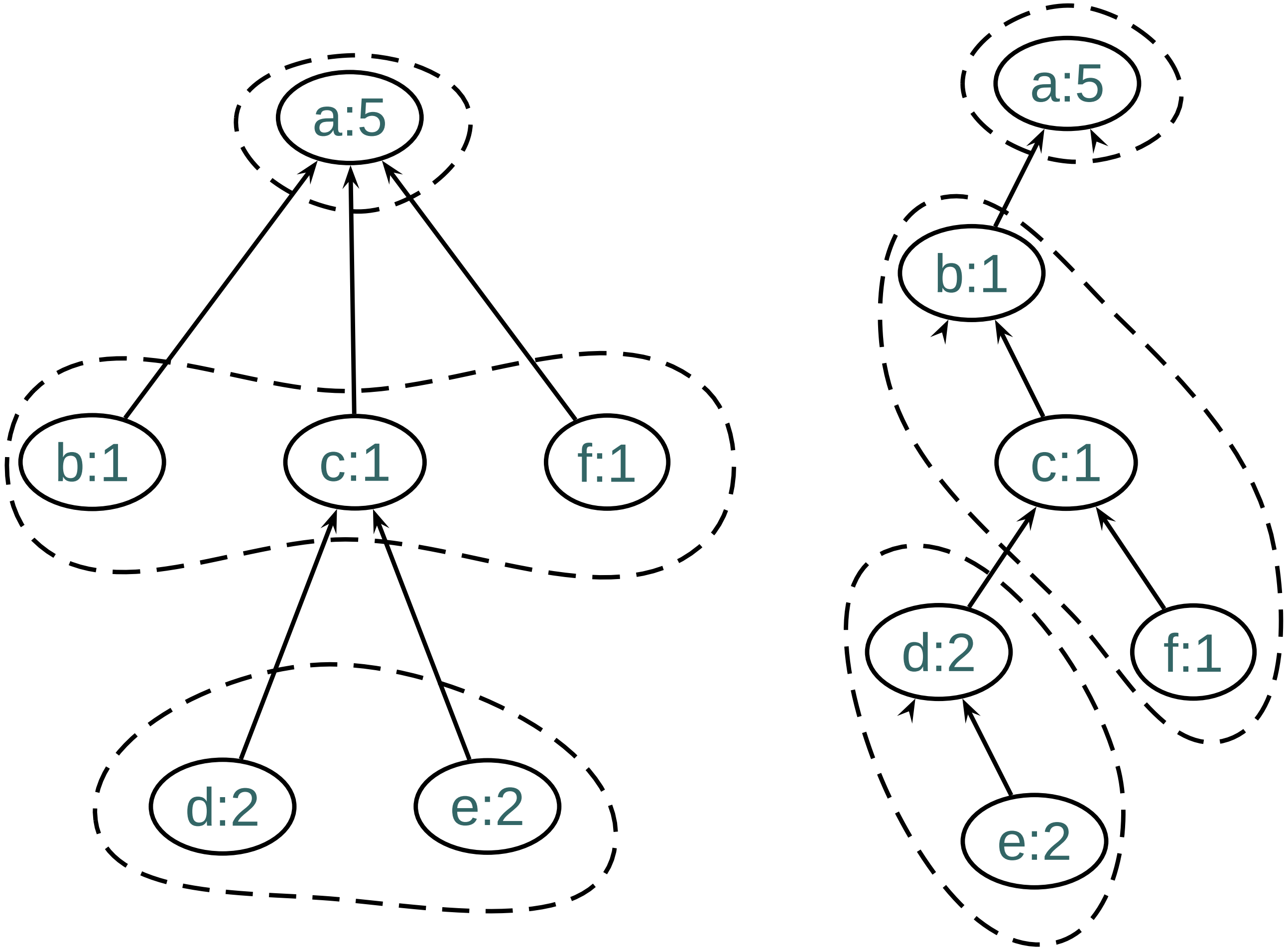
Deep Tree Partitioning

Approximation Algorithms
- Alternative Algorithms
- Enhanced Kundu and Misra (EKM)

Evaluation

Conclusion

Run Kundu and Misra Algorithm on binary representation of n-ary tree!



Number of Partitions

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
- Number of Partitions
- Partitioning Time
- Query Performance

Conclusion

	Document		
Method	DBLP	partsupp.xml	XMark
DHW (Opt)	382	1083	8603
GHDW	384	1083	8838
EKM	402	1091	8975
KM	1294	15876	20519
DFS	1153	2282	25046
BFS	2987	8192	42155

$K = 256$ slots (1 slot = 8 bytes)

UNIVERSITÄT MANNHEIM

Partitioning Time

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
- Number of Partitions
- Partitioning Time
- Query Performance

Conclusion

	Document		
Method	DBLP	partsupp.xml	XMark
DHW (Opt)	24.83	474.13	2041.18
GHDW	0.28	5.55	6.24
EKM	$<$0.01	$<$0.01	0.02
KM	0.05	0.16	0.63
DFS	$<$0.01	$<$0.01	$<$0.01
BFS	$<$0.01	0.02	0.11

Elapsed CPU time in seconds

UNIVERSITÄT MANNHEIM

Query Performance

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
- Number of Partitions
- Partitioning Time
- Query Performance
Conclusion

Query	KM	EKM	Speedup
Space	ca. 8192KB	ca. 8232KB	
Q1	0.065s	0.036s	1.81×
Q2	0.033s	0.023s	1.43×
Q3	0.770s	0.595s	1.29×
Q4	0.344s	0.262s	1.31×
Q5	0.150s	0.074s	2.03×
Q6	0.870s	0.650s	1.34×
Q7	0.854s	0.607s	1.41×

XPathMark on XMark scaling factor 0.1

UNIVERSITÄT MANNHEIM

Conclusion

Introduction
Flat Tree Partitioning
Deep Tree Partitioning
Approximation Algorithms
Evaluation
Conclusion
Conclusion

- Allow siblings to share a partition
  - 50%-90% fewer partitions
  - Query performance $\times 2$
- Optimal dynamic programming algorithm $O(nK^3)$
- Very good approximation algorithm
  - run Kundu and Misra on binary tree (EKM)

UNIVERSITÄT MANNHEIM