Treelogy: A Benchmark Suite for Tree Traversals

Nikhil Hegde, Jianqiao Liu, Kirshanthan Sundararajah, and Milind Kulkarni

School of Electrical and Computer Engineering

Purdue University

Tree algorithms

- Tree algorithms are important
  - Data mining, statistics, scientific computing, graphics, bioinformatics etc.

- Application-specific optimizations and tree algorithms have been developed over the years

Tree algorithms and Optimizations

Tree algorithms

Barnes,1986
Barnes-Hut
Fast multipole m
Rokhlin,1985
Yianilos,1993
Vantage point trees
Accelerating ray tracing
Foley,2005
Alsabti,1997
K-means clustering
Frequent item set mining
Han,2000

Optimizations

Ghoting,2007
Zhang,1997
Locality
Hamada,2009
Communication
Warren,1992
Vectorization
Gray,2001
Scheduling
Makino,1990
Höhl,2002
Liu,2016

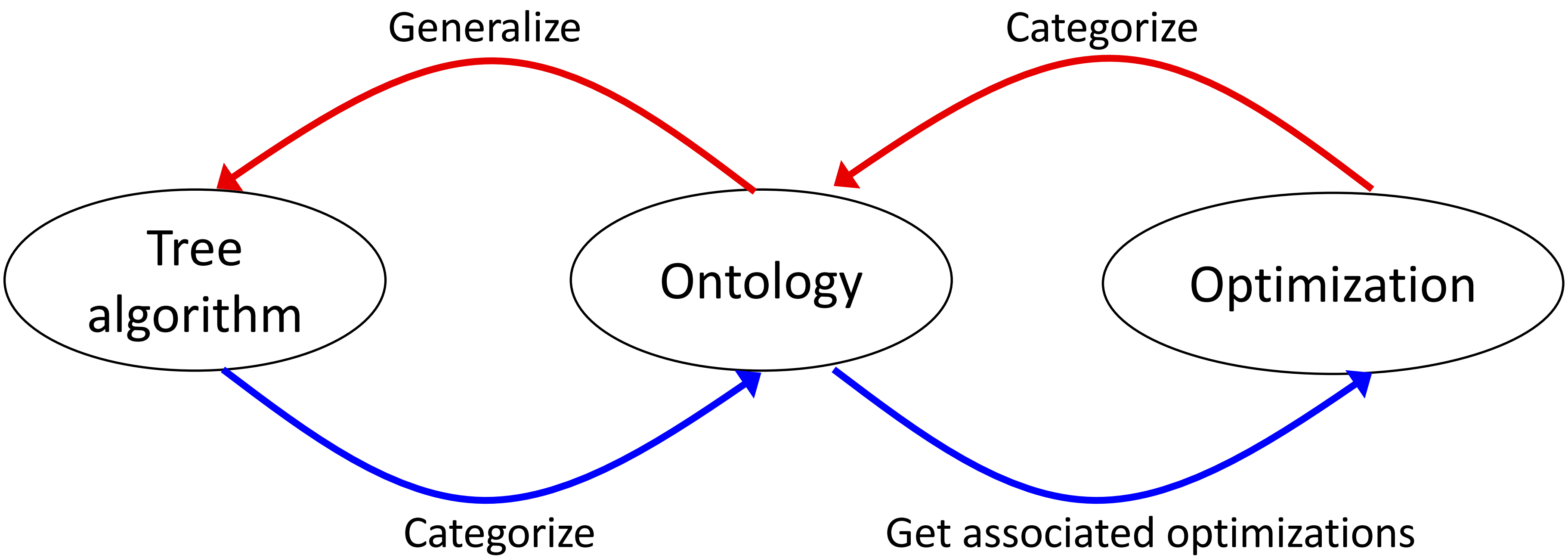
Tree algorithms and optimizations

1. Does the tree algorithm admit an existing optimization?

2. Can an optimization be generalized to other tree algorithms?

Treelogy helps to answer these questions.

Treelogy


Generalize
Categorize
Tree algorithm
Ontology
Optimization
Categorize
Get associated optimizations

Contributions

- Ontology for tree traversal algorithms
- Mapping of optimizations with structural properties of tree algorithms
- A suite of 9 tree traversal algorithms from multiple domains
- Evaluation with multiple tree types and hardware platforms (*GPUs*, *shared*- and *distributed-memory* systems)
- https://bitbucket.org/plcl/treelogy

Background

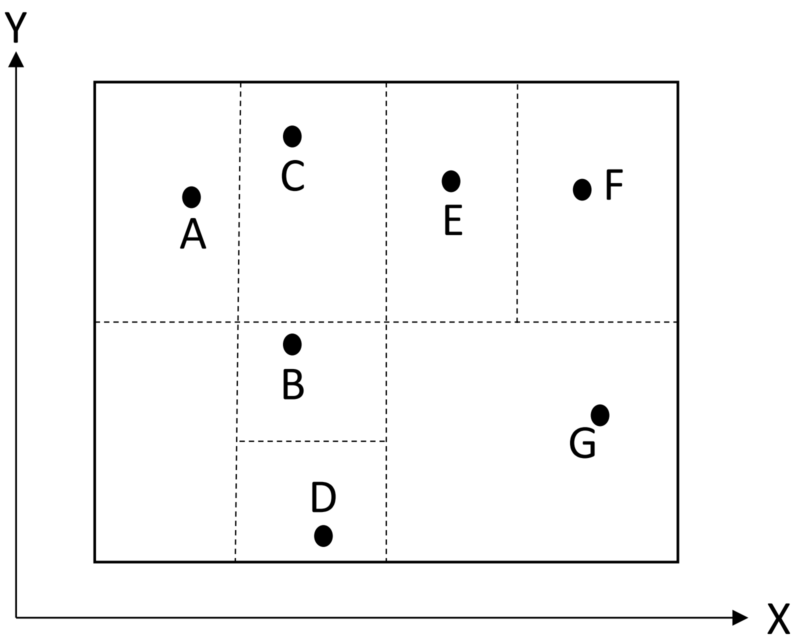
- Why trees and how?
  - Search space elimination and compact data representation
  - Often traversed repeatedly

- Metric trees and n-fix trees are the most common types

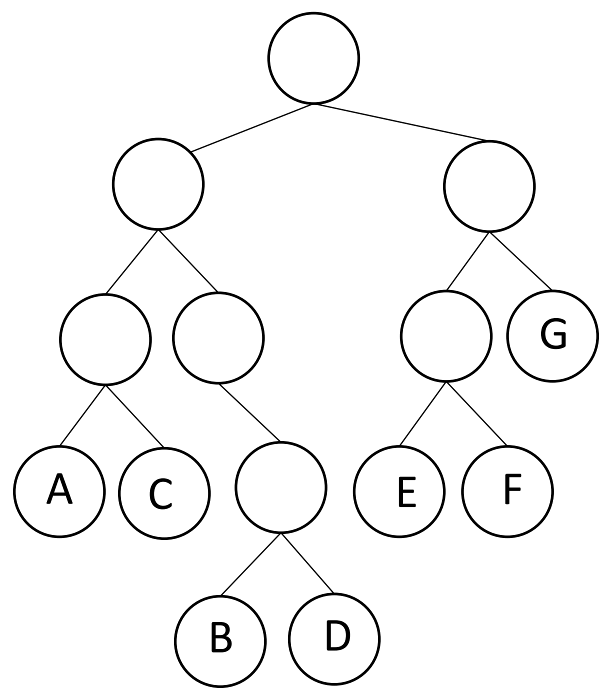
Examples – metric trees

e.g. K-dimensional (kd-), Vantage Point (vp-), quad-trees, octrees, ball-trees

2-dimensional space of points



Binary kd-tree, 1 point /leaf cell



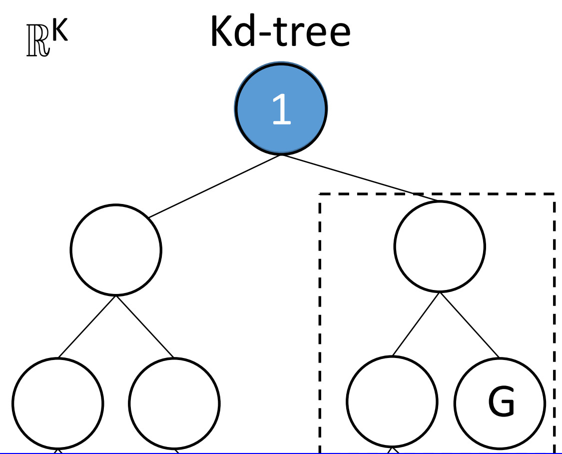
Kd-tree for two-point correlation

Goal: for every point, find the number of points that are located within a given distance R. *Naïve solution:* $O(N^2)$

Input points = {1, 2, … , N} $\in \mathbb{R}^K$

With kd-trees: O(NlogN)

Does the distance to any point within the cell < R ?



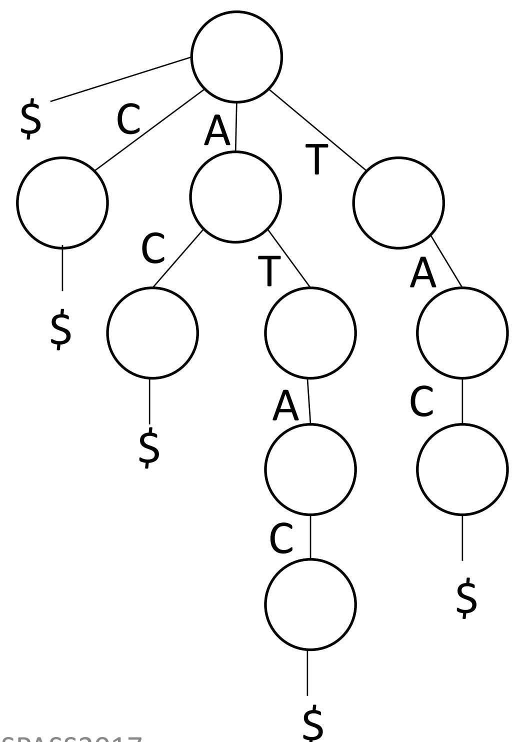
Treelogy kernels with metric trees:

1. Two-point correlation (PC)
2. Nearest Neighbor (NN)
3. K-Nearest Neighbor (K-NN)
4. Barnes-Hut (BH)
5. K-means clustering (KC)
6. Photon mapping (PM)
7. Fast multipole method (FMM)

Examples – n-fix tree

- We refer to prefix and suffix trees as n-fix trees
  - e.g. suffix tree (trie) for string ATAC$

Suffix set: {$} {C} {AC} {TAC} {ATAC}



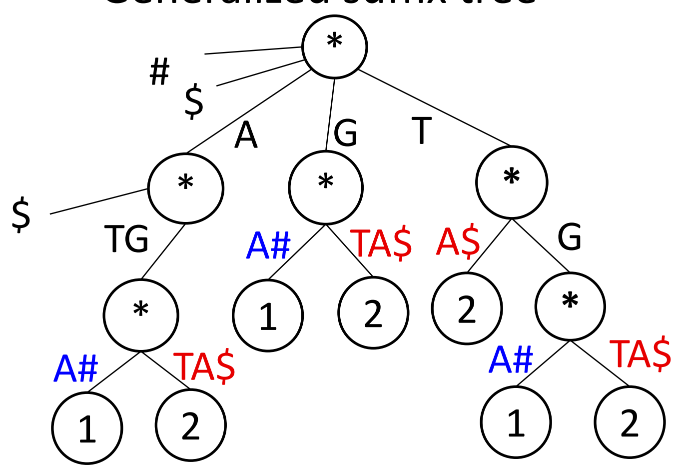
Generalized suffix trees for longest common substring

Goal: find the longest common substring of two strings: 1) ATGA and 2) ATGTA (*answer:* ATG) *Naïve solution:* O(N*M^2)

ATGA#ATGTA$

With suffix trees: O(N+M) in time and space

Generalized suffix tree



Path to ... ex number)

Longest

Treelogy kernels with n-fix trees:

1. Frequent item set mining (FIM)
2. Longest common substring (LCS)

Treelogy Kernels

- Two-point Correlation (PC)
- Nearest Neighbor (NN)
- K-Nearest Neighbor (KNN)
- Barnes-Hut (BH)
- Photon Mapping (PM)
- Frequent Item-set Mining (FIM)
- K-Means Clustering (KC)
- Longest Common Substring (LCS)
- Fast Multipole Method (FMM)

- Two-point Correlation (PC)
- Nearest Neighbor (NN)
- K-Nearest Neighbor (KNN)
- Barnes-Hut (BH)
- Photon Mapping (PM)
- Frequent Item-set Mining (FIM)
- K-Means Clustering (KC)
- Longest Common Substring (LCS)
- Fast Multipole Method (FMM)

- Traversals dominate computation
- Multiple Traversals
- Independent
- Do not modify the tree during traversal

- Traversals dominate computation
- Multiple Traversals
- Independent
- Do not modify the tree during traversal

- Top-down traversal, different tree type
- Bottom-up traversal, same tree type
- Iterative, modify tree or (and) traversals

- Two-point Correlation (PC)
- Photon Mapping (PM)
- Longest Common Substring (LCS)
- K-Means Clustering (KC)

- Barnes-Hut (BH)
- Fast Multipole Method (FMM)

T Bottom-up traversal, same tree type

- Barnes-Hut (BH)
- Frequent Item-set Mining (FIM)

Iterative, modify tree and (or) traversals

The Ontology

- Top-down vs. Bottom-up
- Type of tree
- Iterative with tree mutation
- Iterative with working-set mutation
- Guided vs. Unguided

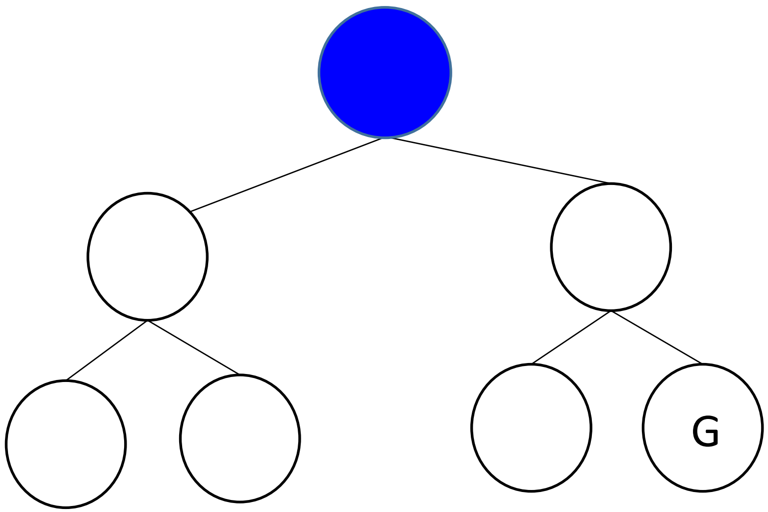
Guided vs. Unguided

1.Unguided traversal[15]

- Fixed order for every traversal
  (e.g. left child followed by right)

2.Guided traversal

- Data dependent traversal order
- Order depends on vertex-computation



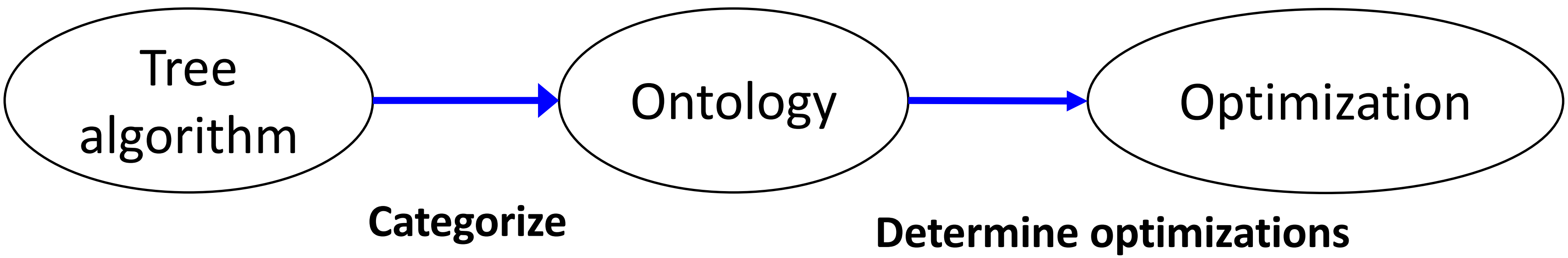
[15] Goldfarb et.al.,SC'13

Classification

Benchmark	Domain	Attributes	Tree Type
Two-Point Correlation	Astrophysics, Statistics	Top-down (preorder), guided (vp), unguided (kd)	Kd, vp
Nearest Neighbor	Data mining	Top-down (preorder), guided	Kd, vp
K-Nearest Neighbor	Data mining	Top-down (preorder), guided	Kd, Ball
Barnes-Hut	Astrophysics	Top-down (preorder), unguided, tree mutation	oct, Kd
Photon Mapping	Computer Graphics	Top-down (preorder), unguided, working-set mutation	Kd
Frequent item-set mining	Data mining	Bottom-up, unguided, tree mutation, working-set mutation	Prefix
K-Means Clustering	Data mining, Machine learning	Top-down (inorder), guided, tree mutation	Kd
Longest common substring	Bioinformatics	Top-down (postorder), unguided, tree mutation	Suffix
Fast Multipole Method	Scientific computing	Top-down (preorder) and bottom-up, unguided, tree mutation	Quad

Algorithm -> Ontology

What we have seen so far…



Optimizations

- Optimizations are effective only when certain properties hold

Optimization	Structural properties
Profile driven scheduling	Top-down
Tiling	Top-down, bottom-up
Vectorization	Unguided
Data representation	Vp trees for NN, prefix trees for FIM, suffix trees for LCS.
Communication overhead	Top-down

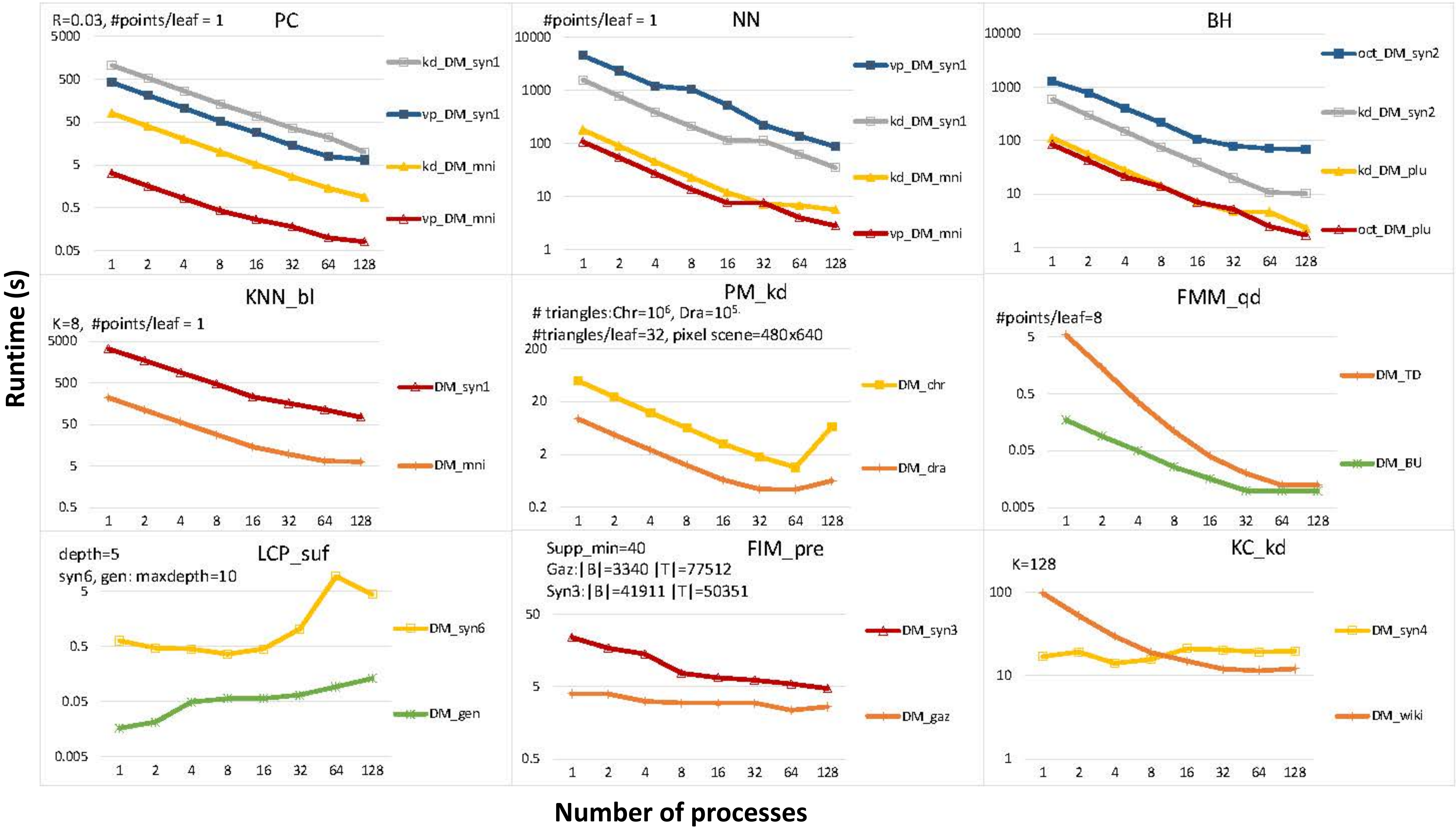
Liu,2016 Jo,2012 Hamada,2009 Makino,1990 Zhang,1997 Ghoting,1997 Jo,2011 Kumar,2008 Han,2000 Höhl,2002 Zhang,1997 Warren,1992

Evaluation Methodology

- Platforms:
  - **Shared-memory (SHM):** processors - 2 10-core Xeon E5 2660 V3, memory - 32 KB L1, 256KB L2, 25MB L3, 64GB RAM
  - **Distributed-memory (DM):** 10 nodes with high-speed Ethernet interconnect
  - **GPU:** nVidia Tesla K20C.

    host – 2 AMD 6164 HE processors, 32GB RAM
- Metrics:
  - Architecture-independent
    - Average traversal length, Load imbalance
  - Architecture-dependent
    - L3 Miss Rate, CPI
- All measurements consider traversal times only

Scalability



Scalability contd.

- Adding more cores results in better performance
  - DM plots show excellent scaling
  - SHM and GPU plots similar

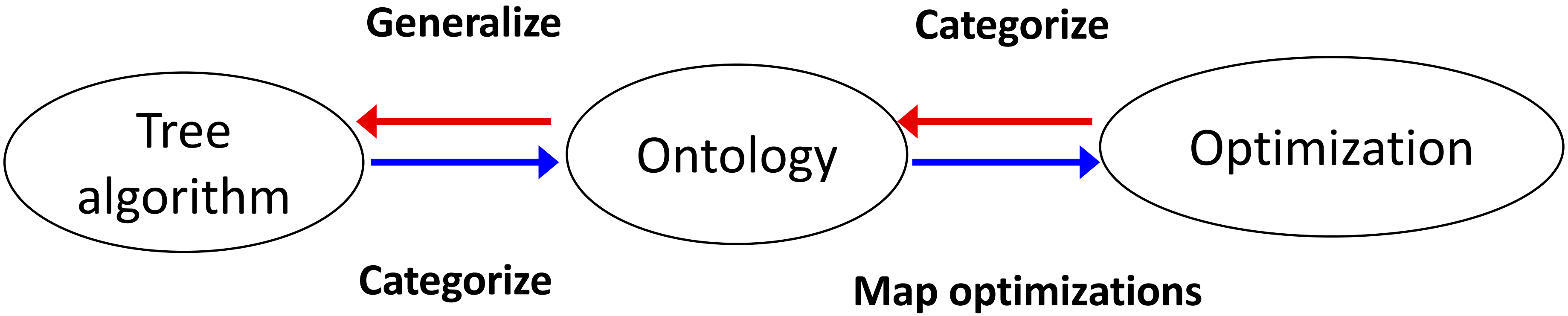
- KC and LCS are exceptions
  - Iterative tree mutation algorithms marked by heavy synchronization at the end of an iteration
  - LCS less available parallelism

Summary (scalability)

- Most kernels scale well while taking advantage of ontology-driven optimizations
- Point Correlation (PC) with vp-tree is better than kd-tree
- Barnes-Hut (BH) is sensitive to tree type and input distribution

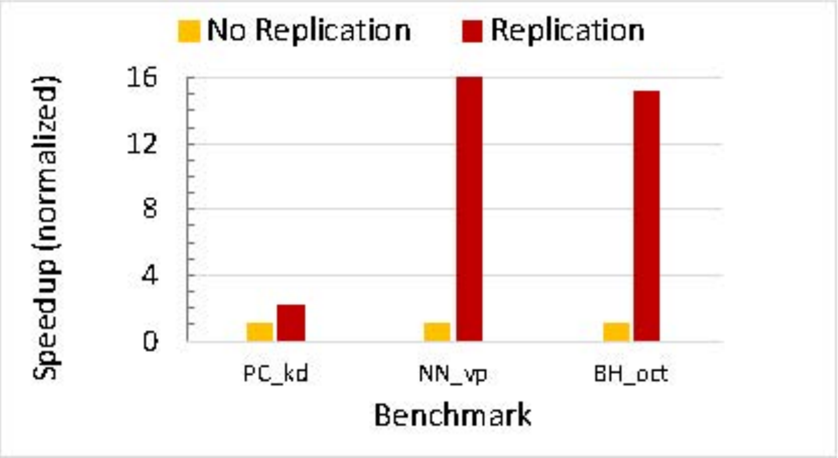
Algorithm <- Optimization

What we have seen so far…



Case study

- Generalizing locally essential trees (LET)
  - BH specific (distributed-memory)
  - Partial replication of tree structure



- Partial replication of only the top-subtree.
  - Improves load-imbalance and minimizes communication overhead

Conclusions

- Treelogy
  - Ontology
  - Mapping of optimizations to structural properties
  - A suite of 9 tree traversal kernels spanning ontology
  - Shared-memory, distributed-memory, and GPU implementations
    - Multiple tree types based on popularity and efficiency
- Evaluations showed that most kernels scale well
  - Two-point correlation (PC) with vp-trees better than standard tree used in literature

Thank you