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 method
- Vantage point trees
- Accelerating ray tracing
- K-means clustering
- Frequent item set mining
  - Han, 2000
- Yianilos, 1993
- Alsabti, 1997

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

ISPASS2017
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

- Tree algorithm
- Ontology
- Optimization

Generalize
Categorize
Get associated optimizations
Categorize
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](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 = $\{1, 2, \ldots, N\} \in \mathbb{R}^K$

*With kd-trees: $O(N\log N)$*

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

Treeology 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:* \(O(N\times M^2)\)

**ATGA#ATGTA\$**

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

Path to a node: substring of string 1 or string 2 or both (vertex number)

Generalized suffix tree

**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)

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

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

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\textsuperscript{[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

\textsuperscript{[15]} Goldfarb et al., SC’13
## Classification

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

What we have seen so far...

Tree algorithm -> Ontology -> Optimization

Categorize -> Determine optimizations
## Optimizations

- Optimizations are effective only when certain properties hold

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Structural properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile driven scheduling</td>
<td>Top-down</td>
</tr>
<tr>
<td>Tiling</td>
<td>Top-down, bottom-up</td>
</tr>
<tr>
<td>Vectorization</td>
<td>Unguided</td>
</tr>
<tr>
<td>Data representation</td>
<td>Vp trees for NN, prefix trees for FIM, suffix trees for LCS.</td>
</tr>
<tr>
<td>Communication overhead</td>
<td>Top-down</td>
</tr>
</tbody>
</table>

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

Number of processes
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...

Tree algorithm <-> Ontology <-> Optimization

Generalize <-> Categorize <-> Map optimizations
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