Path Query Data Structures in Practice

Meng He, *Serikzhan Kazi*

June 3, 2020

Dalhousie University

Plan

Motivation

Both theoretical and practical reasons:

- Proliferation of tree-structured data (think `xml/json` etc)
- Expected height of a tree is $\Theta(\sqrt{n})$
- Becoming first-class citizen in established domains such as e.g. RDBMS: see PostgreSQL's ltree module.
- graph databases

Query Types

- *Path Counting*: return $|\{z \in P_{x,y} \,|\, \mathbf{w}(z) \in Q\}|$.
- *Path Reporting*: enumerate $\{z \in P_{x,y} \,|\, \mathbf{w}(z) \in Q\}$.
- *Path Selection*: return the k^{th} $(0 \leq k < |P_{x,y}|)$ weight in the sorted list of weights on $P_{x,y}$; k is given at query time. In the special case of $k = \lfloor |P_{x,y}|/2 \rfloor$, a path selection is a *path median query*.

Motivation (contd.)

Empirical studies:

- ✓ (traditional) orthogonal range searching
- ✓ navigation and queries in succinct trees
- ✗ queries in weighted trees

Method

Source	Space	Time
Patil et al. [PST12]	$6n + n\lg\sigma + \mathcal{o}(n\lg\sigma)$	$\mathcal{O}(\lg n \lg\sigma)$
He et al. [HMZ16]	$n(2 + \lg\sigma) + \mathcal{o}(n\lg\sigma)$	$\mathcal{O}(\lg\sigma / \lg\lg n)$

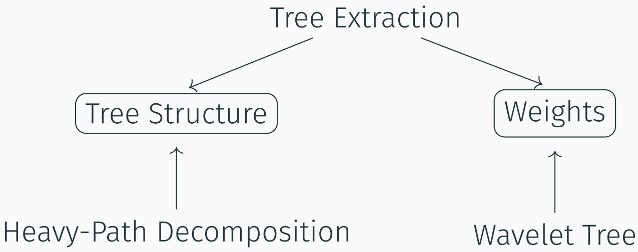
Datasets

	num nodes	diameter	σ	$\log \sigma$	H_0	Description
eu.mst.osm	27,024,535	109,251	121,270	16.89	9.52	An MST we constructed over map of Europe [Ope17]
eu.mst.dmcs	18,010,173	115,920	843,781	19.69	8.93	An MST we constructed over European road network [kit]
eu.emst.dem	50,000,000	175,518	5020	12.29	9.95	An Euclidean MST we constructed over DEM of Europe [srt]
mrs.emst.dem	30,000,000	164,482	29,367	14.84	13.23	An Euclidean MST we constructed over DEM of Mars [mar]

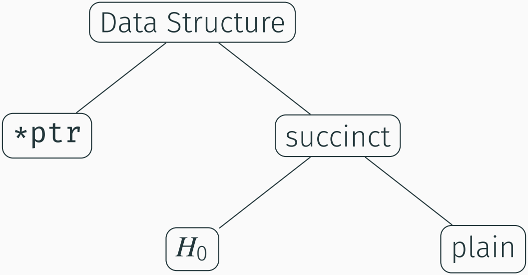
DEM – Digital Elevation Model; Euclidean MST – Euclidean Minimum Spanning Tree obtained using CGAL. Road networks are due to OpenStreetMap and KIT.

Plan

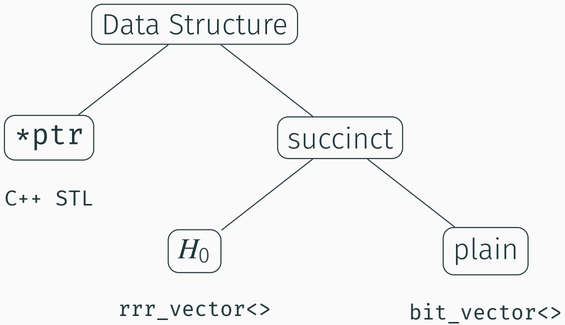
Analysis


Tree Extraction
Tree Structure
Weights
Heavy-Path Decomposition
Wavelet Tree

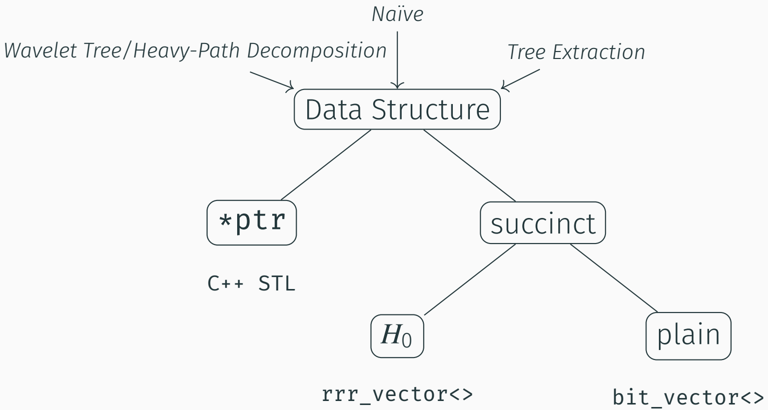
Implementation



Implementation



Implementation

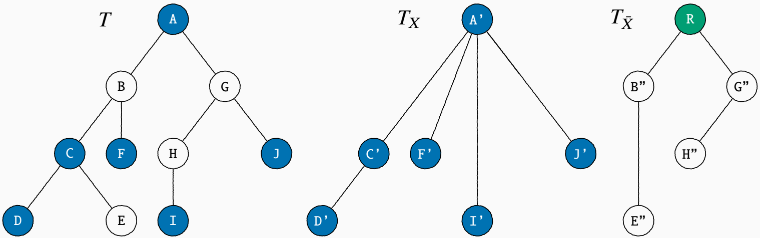


Notation

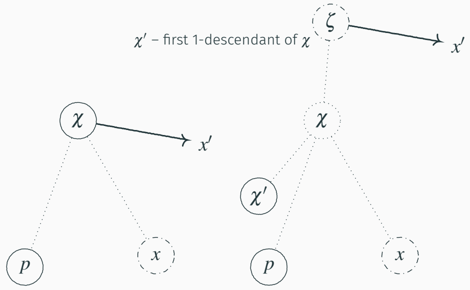
	Symbol	Description
pointer-based	nv	Naïve data structure
	$\mathtt{nv}^{\mathrm{L}}$	Naïve data structure, augmented with $\mathcal{O}(1)$ query-time LCA of [BFP+05]
	$\mathtt{ext}^{\dagger}$	A solution based on tree extraction [HMZ16]
	$\mathtt{whp}^{\dagger}$	A non-succinct version of the wavelet tree- and heavy-path decomposition-based solution of [PST12]
succinct	$\mathtt{nv}^{\mathrm{c}}$	Naïve data structure, using succinct data structures to represent the tree structure and weights
	$\mathtt{ext}^{\mathrm{c}}$	$3n\lg\sigma + o(n\lg\sigma)$-bits-of-space scheme for tree extraction, with compressed bitmaps
	$\mathtt{ext}^{\mathrm{P}}$	$3n\lg\sigma + o(n\lg\sigma)$-bits-of-space scheme for tree extraction, with uncompressed bitmaps
	$\mathtt{whp}^{\mathrm{c}}$	Succinct version of whp, with compressed bitmaps
	$\mathtt{whp}^{\mathrm{P}}$	Succinct version of whp, with uncompressed bitmaps

The implemented data structures and the abbreviations used to refer to them.

Tree Extraction



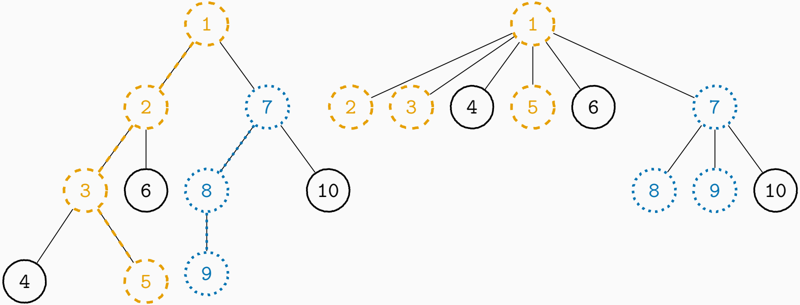
0/1-Parents



The 1-predecessor of x

Our implementation uses $3n \lg \sigma + \mathcal{O}(n \lg \sigma)$ bits, i.e. 3 times as much as optimal [HMZ16].

Heavy-Path Decomposition



$6n + \mathcal{O}(n)$-bit encoding of tree topology and its heavy-path decomposition due to Patil et al. [PST12].

Tools

framework: `sdsl-lite`

- `int_vector<>/bit_vector<>`
- `rrr_vector<>`
- `b[alanced]p[arentheses]_support`
- `rank/select`
- `wt_int<>`
- ...

timing: `google-benchmark`

memory: `malloc_count`

testing: `googletest`

datasets preparation: utilities and libraries:

- gdal
- cgal
- osm2po

Plan

Introduction

Methods

Results

Discussion

Path Median and Path Counting

	Dataset	nv	nv^L	$ext^\dagger$	$whp^\dagger$	nv^c	ext^c	ext^P	whp^c	whp^P	
median	eu.mst.osm	658	475	4.22	6.10	7078	85.3	51.1	111	51.2	
median	eu.mst.dmcs	566	412	5.16	6.28	6556	84.6	54.8	120	54.7	
median	eu.emst.dem	710	436	4.44	5.10	9404	106	81.9	96.7	54.9	
median	mrs.emst.dem	472	298	4.93	4.53	7018	124	97.0	88.3	49.5	
counting	eu.mst.osm	238	140	6.88	18.4	3553	247	167	139	56.9	large
counting	eu.mst.dmcs	204	121	7.31	19.7	3300	253	178	142	57.3	large
counting	eu.emst.dem	338	195	5.97	11.5	4835	215	168	105	55.9	large
counting	mrs.emst.dem	232	174	5.25	8.40	3614	206	164	91	49.3	large
counting	eu.mst.osm	244	143	5.47	17.8	3555	213	146	129	54.2	medium
counting	eu.mst.dmcs	209	124	6.94	18.4	3297	224	160	133	56.5	medium
counting	eu.emst.dem	339	195	4.55	10.0	4840	178	140	100	54.9	medium
counting	mrs.emst.dem	237	143	5.91	8.74	3613	199	154	89.7	48.9	medium
counting	eu.mst.osm	239	139	5.25	15.4	3551	190	132	119	53.9	small
counting	eu.mst.dmcs	209	123	5.25	18.9	3300	206	148	126	55.2	small
counting	eu.emst.dem	347	200	3.92	9.34	4832	154	124	94.9	53.2	small
counting	mrs.emst.dem	238	144	4.82	7.41	3615	178	133	84.2	47.6	small

Average time to answer a query, from a fixed set of 10^6 randomly generated path median and path counting queries, in microseconds. Path counting queries are given in `large`, `medium`, and `small` configurations.

Path Reporting

Dataset	κ	nv	nv^{L}	ext†	whp†	nv^{c}	ext^{c}	ext^{P}	whp^{c}	whp^{P}	
eu.mst.osm	9,840	356	256	184	70.7	3766					large
eu.mst.dmcs	9,163	309	224	147	66.8	3485					large
eu.emst.dem	14,211	389	241	140	77.5	4926					large
mrs.emst.dem	10,576	267	178	89.2	55.1	3668					large
eu.mst.osm	1,093	322	222	43.7	28.8	3706					medium
eu.mst.dmcs	1,090	277	196	34.0	29.7	3434					medium
eu.emst.dem	1,464	354	206	32.1	20.1	4880					medium
mrs.emst.dem	1,392	250	151	22.1	15.6	3639					medium
eu.mst.osm	182	311	212	13.8	19.0	3685	1965	485	795	226	small
eu.mst.dmcs	236	271	193	13.2	21.0	3529	2518	632	1043	292	small
eu.emst.dem	215	353	203	10.2	12.7	4873	1276	378	590	205	small
mrs.emst.dem	117	242	145	8.88	9.57	3632	881	278	475	162	small

Average time to answer a path reporting query, from a fixed set of 10^6 randomly generated path reporting queries, in microseconds. The queries are given in `large`, `medium`, and `small` configurations. Average output size for each group is given in column κ.

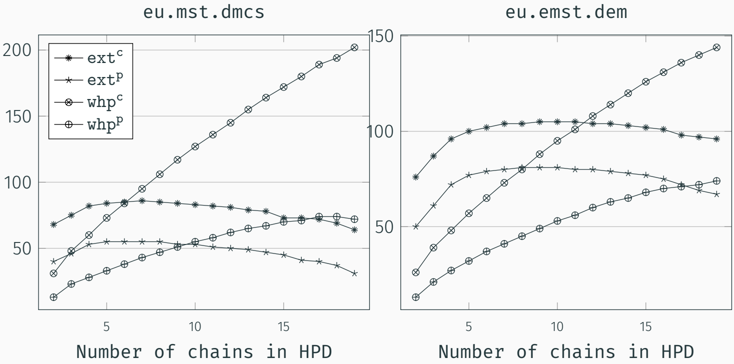
Space

	Dataset	nv	nv^L	$whp^†$	$ext^†$	nv^c	ext^c	ext^P	whp^c	whp^P
space	eu.mst.osm	406.3	972.1	3801	5943	21.71	59.85	75.74	21.71	34.42
	eu.mst.dmcs	406.4	974.0	4274	6768	34.46	82.16	106.0	29.69	48.77
	eu.emst.dem	394.1	988.5	3342	4613	19.64	45.41	59.15	19.64	31.66
	mrs.emst.dem	386.7	1005	3579	5383	17.35	51.71	66.02	17.35	28.80
peak/time	eu.mst.osm	491.0/1	987.9/5	3785/28	9586/47	21.71/1	295.0/23	295.0/23	1347/62	1347/61
	eu.mst.dmcs	439.8/1	1002/4	4403/19	12382/37	29.69/1	399.7/18	399.7/18	1360/42	1360/42
	eu.emst.dem	401.0/2	1021/10	3460/47	5286/67	19.64/1	287.6/32	287.6/32	1333/115	1333/115
	mrs.emst.dem	392.4/1	1016/5	3719/30	6027/46	17.35/1	269.3/22	269.3/22	1337/69	1337/69

(upper) Space occupancy of our data structures, in bits per node, when loaded into memory; (lower) peak memory usage (m in bits per node) during construction and construction time (t in seconds) shown as $\mathbf{m}/t$.

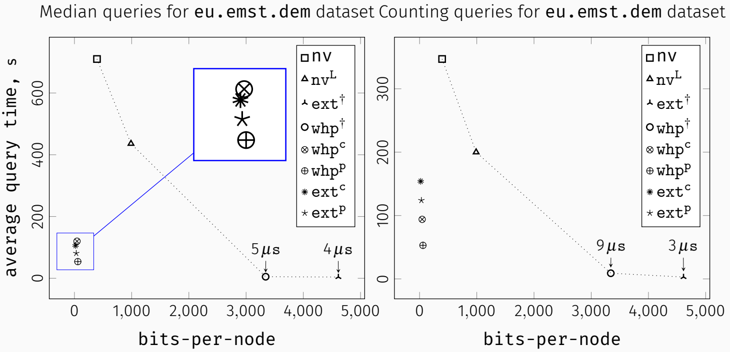
Comparison of ext and whp

From the full version:



Average time to answer a path median query, controlled for the number of segments in heavy-path decomposition, in microseconds. Random fixed query set of size 10^6.

Overall Evaluation



Visualization of some of the entries in Section 3. Inner rectangle magnifies the mutual configuration of the succinct data structures **whp**^{P},**whp**^{c},**ext**^{P}, and **ext**^{c}. The succinct naïve structure **nv**^{c} is not shown.

Conclusions

- Succinct data structures for path queries are **competitive** with more traditional approaches that are optimized either for speed or storage[1]
- `whp` is practical, overall average-case good choice
- When **worst-case** performance is important, `ext` should be preferred to `whp`

[1] except, possibly, for reporting queries

Plan

HPD/WT Solution

Wavelet tree search is launched **independently** over each of the heavy-path segments. But the segments themselves are **not independent** – a query node uniquely determines all the segments to be searched, and **whp** is “more powerful than needed” in that it does not take advantage of this.

Tree Extraction

Out $3n \lg \sigma + \mathcal{O}(n \lg \sigma)$-bit representation is 3 times worse than the optimal [HMZ16]. While time- and space-optimal solution needs **non-trivial word-RAM structures** and **lookup tables**, is better – time- or space-wise – alternative to our approach possible? This is an **interesting open problem** in algorithm engineering.

Acknowledgements

We acknowledge PTV `https://www.ptvgroup.com/` for providing data of the European road graph.

Thank you!

References i

- Michael A. Bender, Martin Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin, *Lowest common ancestors in trees and directed acyclic graphs*, J. Algorithms **57** (2005), no. 2, 75–94.
- Meng He, J. Ian Munro, and Gelin Zhou, *Data structures for path queries*, ACM Trans. Algorithms **12** (2016), no. 4, 53:1–53:32.
- *KIT roadgraphs*, `https://i11www.iti.kit.edu/information/roadgraphs`, Accessed: 07/12/2018.

References ii

- *MOLA Mars Orbiter Laser Altimeter data from NASA Mars Global Surveyor*, `https://planetarymaps.usgs.gov/mosaic/Mars_MGS_MOLA_DEM_mosaic_global_463m.tif`, Accessed: 10/01/2019.
- OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org* , `https://www.openstreetmap.org`, 2017.
- Manish Patil, Rahul Shah, and Sharma V. Thankachan, *Succinct representations of weighted trees supporting path queries*, J. Discrete Algorithms **17** (2012), 103–108.

SRTM Shuttle Radar Topography Mission, http://srtm.csi.cgiar.org/srtmdata/, Accessed: 10/01/2019.