An edge quadtree for external memory

Herman Haverkort[1] Mark McGranaghan[2] Laura Toma[2]

[1]Eindhoven University of Technology, the Netherlands
[2]Bowdoin College, USA

Symposium of Experimental Algorithms
June 2013, Rome

Outline

1. The problem
2. Preliminaries
3. Our algorithm
4. Empirical evaluation
    - Application: Map overlay

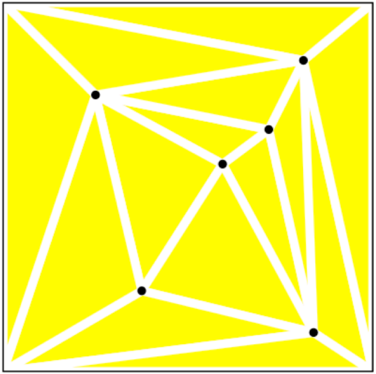
The problem

Build a quadtree subdivision for a set of non-intersecting edges in the plane.

Goal: Scalable to very large data (IO-efficient)

Quadtree: divide unit square into quadrants, refine until data per cell is “small”

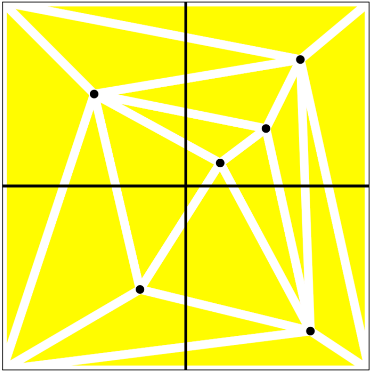
(for e.g., until every cell has at most one vertex)



figures thanks to H. Haverkort

Quadtree: divide unit square into quadrants, refine until data per cell is “scriptsize”

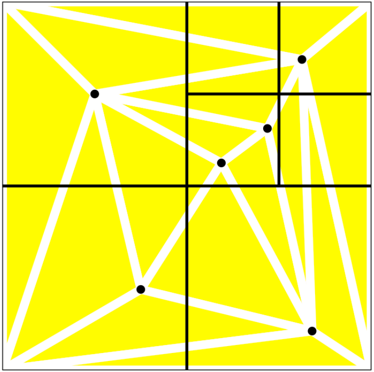
(for e.g., until every cell has at most one vertex)



figures thanks to H. Haverkort

Quadtree: divide unit square into quadrants, refine until data per cell is "scriptsize"

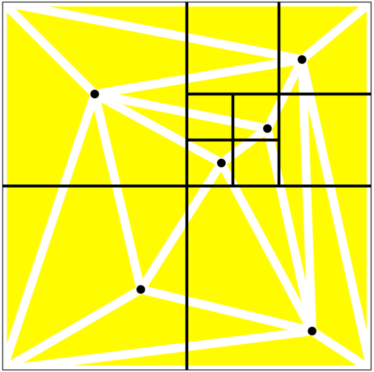
(for e.g., until every cell has at most one vertex)



figures thanks to H. Haverkort

Quadtree: divide unit square into quadrants, refine until data per cell is “scriptsize”

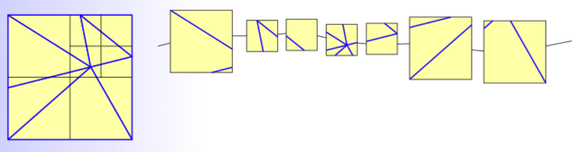
(for e.g., until every cell has at most one vertex)



figures thanks to H. Haverkort

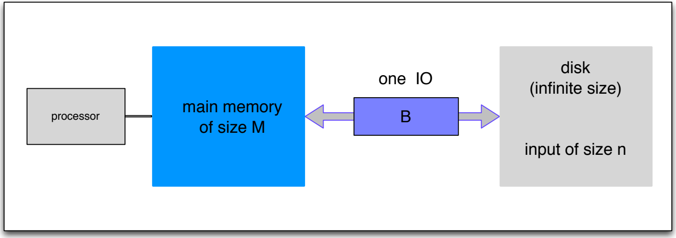
Linear edge quadtree

Linear (edge) quadtree: set of leaf cells, each cell storing its data.



- space efficient

The IO-Model [Agarwal & Vitter, 1988]



- IO-complexity: number of IOs
- Fundamental bounds
  - scan: $\texttt{scan}(n) = \frac{n}{B}$ IOs
  - sort: $\texttt{sort}(n) = \Theta(\frac{n}{B} \log_{M/B} \frac{n}{M})$ IOs

Quadtrees: Related work

Point quadtree

- compressed: $O(n)$ cells, $O(1)$ points each

Edge quadtree

- Build quadtree induced by endpoints and distribute edges
  - $O(n)$ cells, $O(1)$ points per cell, $I = O(n^2)$ edge-cell intersections
- Specific stopping criteria
  - E.g. split a region until it intersects a single edge (unbounded size)

Quadtrees: Related work

- Samet et al ('85,'86,'89,'92,'97,'99,'02)
  - PM quadtree (PM1, PM2, PM3)
  - segment quadtree
  - PMR quadtree
- Agarwal et al. 2006
  - build point qdt with $O(k)$ points per cell in $O(\frac{n}{B}\frac{h}{\log M/B}$ IOs
  - this is $O(\text{sort}(n))$ IOs when points are nicely distributed
- De Berg et al. 2010
  - star-qdt, guard-qdt
  - $O(1)$ point per cell, $O(\text{sort}(n+l))$ IOs
  - exploit fatness/low density: $O(1)$ edges per cell, $l = O(n)$

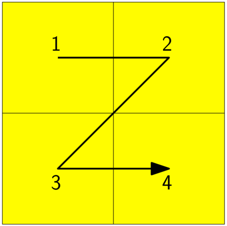
Our contributions

Let $\mathcal{E}$ be a set of n non-intersecting segments in the plane.
Let $k (k \geq 1)$ be a user-defined parameter.

- K-quadtree
  - Build a compressed, linear quadtree on the endpoints of E with $O(n/k)$ cells and $O(k)$ points in each cell in $O(\text{sort}(n))$ IOs.
  - Compute the intersections between the edges and the quadtree subdivision in $O(\text{sort}(n + I))$ IOs.
- Empirical evaluation
  - triangulated terrains, TIGER data

Quadtrees and Z-order

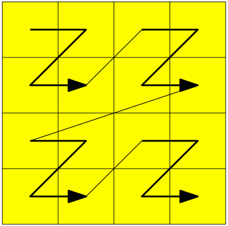
Z-order space filling curve: visits quadrants recursively in order NW, NE, SW, SE



figures thanks to H. Haverkort

Quadtrees and Z-order

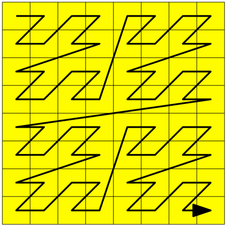
Z-order space filling curve: visits quadrants recursively in order NW, NE, SW, SE



figures thanks to H. Haverkort

Quadtrees and Z-order

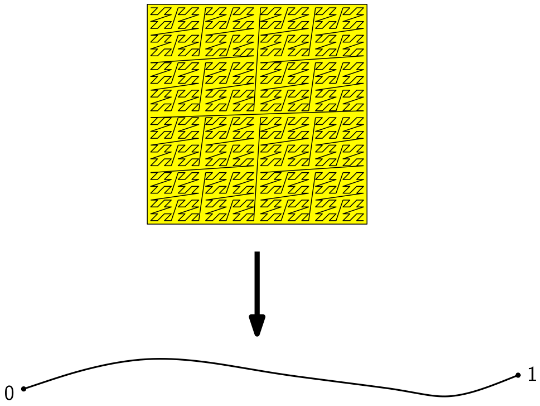
Z-order space filling curve: visits quadrants recursively in order NW, NE, SW, SE



figures thanks to H. Haverkort

Quadtrees and Z-order

Z-order space filling curve: visits quadrants recursively in order NW, NE, SW, SE

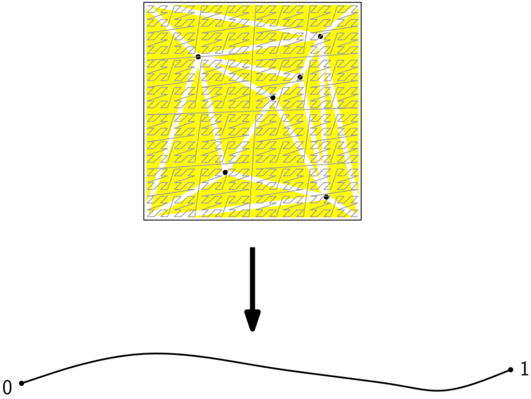


figures thanks to H. Haverkort

Quadtrees and Z-order

Quadtree cell = interval on Z-order curve

Quadtree subdivision = subdivision of Z-order curve

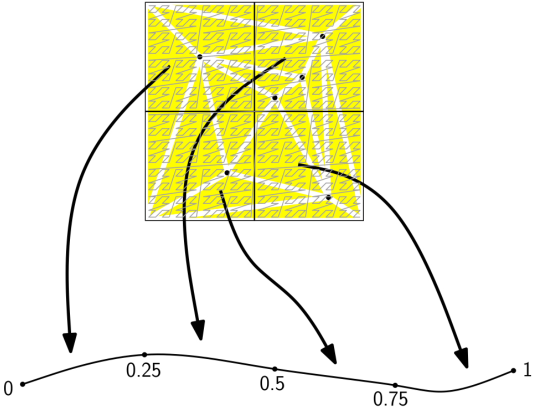


figures thanks to H. Haverkort

Quadtrees and Z-order

Quadtree cell = interval on Z-order curve

Quadtree subdivision = subdivision of Z-order curve

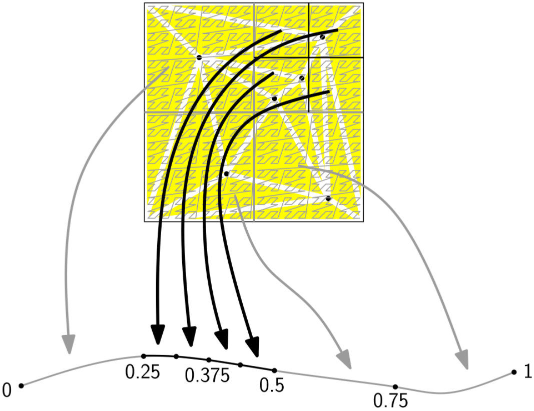


figures thanks to H. Haverkort

Quadtrees and Z-order

Quadtree cell = interval on Z-order curve

Quadtree subdivision = subdivision of Z-order curve

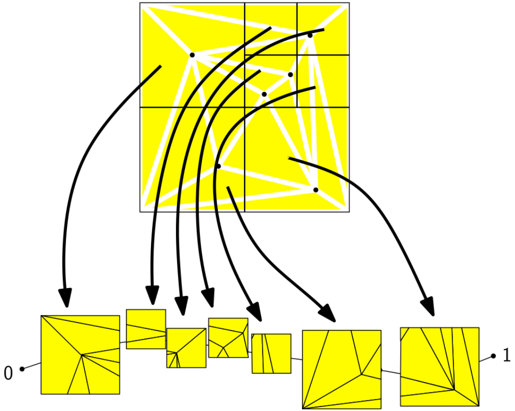


figures thanks to H. Haverkort

Quadtrees and Z-order

Quadtree cell = interval on Z-order curve

Quadtree subdivision = subdivision of Z-order curve

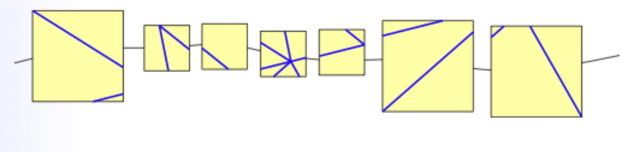


figures thanks to H. Haverkort

Our algorithm

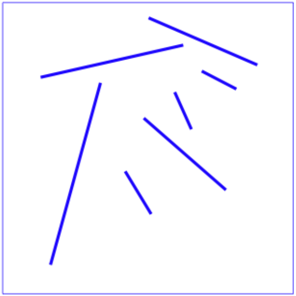
Input: Set of non-intersecting segments $\mathcal{E}$ and a value $k \geq 1$.

1. Construct the subdivision of the endpoints of $\mathcal{E}$
$\implies Q = \{[z_1 = 0, z_2], [z_2, z_3], ...\}$ a subdivision of $[0, 1]$
2. Compute the edge-cell intersections
$\implies Q = \{...(z_1, e), ...\}$



Our algorithm: Constructing the subdivision

Input: Set of non-intersecting segments $\mathcal{E}$ and value k.

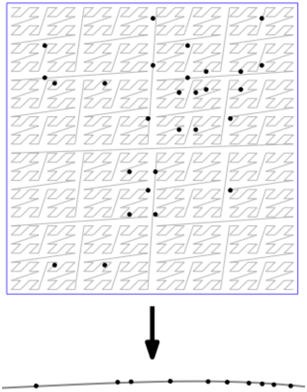


Our algorithm: Constructing the subdivision

Input: Set of non-intersecting segments $\mathcal{E}$ and value k.

Algorithm:

1. Find the endpoints of the segments and sort them in Z-order.

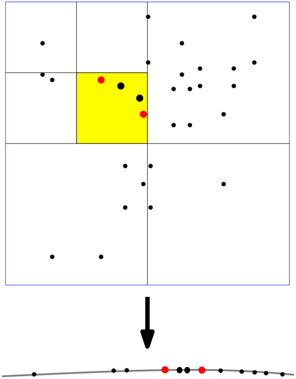


Our algorithm: Constructing the subdivision

Input: Set of non-intersecting segments $\mathcal{E}$ and value k.

Algorithm:

1. Find the endpoints of the segments and sort them in Z-order.
2. Let $P_k = \{p_0, p_k, p_{2k}, ...\}$ the set of every kth point. For every two consecutive points p and p' in P_k:
   - find smallest cell Q that contains p and p'

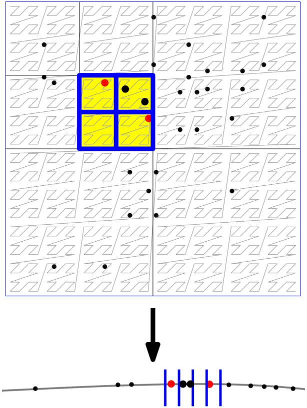


Our algorithm: Constructing the subdivision

Input: Set of non-intersecting segments $\mathcal{E}$ and value k.

Algorithm:

1. Find the endpoints of the segments and sort them in Z-order.
2. Let $P_k = \{p_0, p_k, p_{2k}, ...\}$ the set of every kth point. For every two consecutive points p and p' in P_k:
   - find smallest cell Q that contains p and p'
   - output cell boundaries of Q and its quadrants



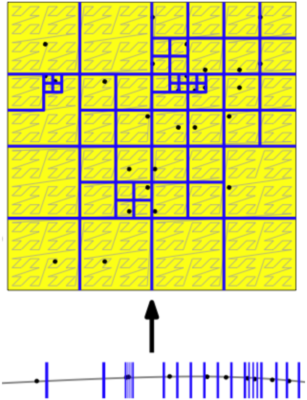
Our algorithm: Constructing the subdivision

Input: Set of non-intersecting segments $\mathcal{E}$ and value k.

Algorithm:

1. Find the endpoints of the segments and sort them in Z-order.
2. Let $P_k = \{p_0, p_k, p_{2k}, ...\}$ the set of every kth point. For every two consecutive points p and p' in P_k:
   - find smallest cell Q that contains p and p'
   - output cell boundaries of Q and its quadrants

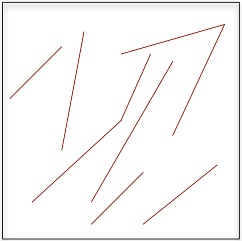
Lemma: Subdivision with $O(n/k)$ cells and $O(k)$ vertices per cell.



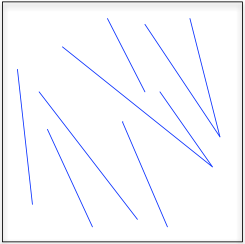
Our algorithm: Edge-cell intersection

Goal: For every $I_j = [z_j, z_{j+1}] \in Q$, compute the edges that intersect σ_j.

- Let E^+ (E^-) be the edges of positive (negative) slope



E+

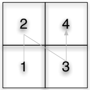


E-

Idea: Process E^+ and E^- separately.

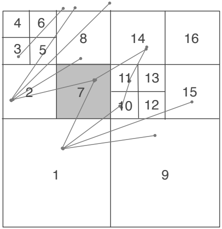
Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+



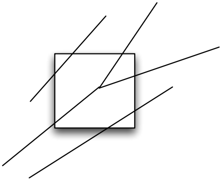
Algorithm:

- For each interval $I_j = [z_j, z_{j+1}]$ in Q, find all edges in E^+ that intersect I_j



Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+



Algorithm:

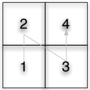
- For each interval $I_j = [z_j, z_{j+1}]$ in Q, find all edges in E^+ that intersect I_j

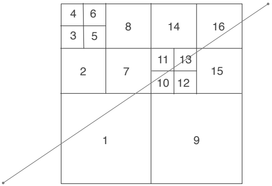
The edges that intersect I_j either:

- start in I_j or,
- start in an interval outside I_j

Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+





Lemma

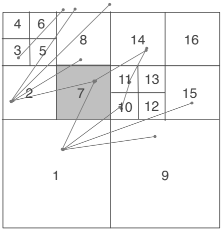
An edge of positive slope intersects the cells in Q in Z-order.

The edges that intersect I_j either:

- start in I_j, or,
- start in an interval before I_j

Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+



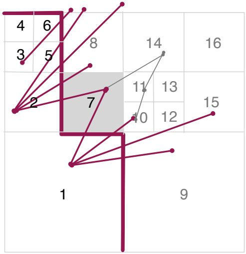
Algorithm:

- Sort E^+ by Z-index of first endpoint.
- For each interval $I_j = [z_j, z_{j+1}]$ in Q:
    - Find all edges in E^+ that originate in I_j: read all $e = (p, q)$ from E^+ s. th. $z(p) \in I_j$
    - Find all edges in E^+ that originate in an interval before I_j: HOW?

Finding the intersections of E^+ and Q ($k = 1$)

B_j: boundary between $\cup_{i<j}\sigma_i$ and $\cup_{i\geq j}\sigma_i$

BL_j: the edges that intersect B_j, in order.



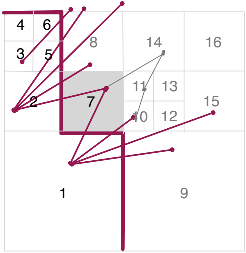
BL6

Finding the intersections of E^+ and Q ($k = 1$)

B_j: boundary between $\cup_{i<j}\sigma_i$ and $\cup_{i\geq j}\sigma_i$
BL_j: the edges that intersect B_j, in order.

Lemma

B_j is a monotone staircase and the intersection of σ_j and B_{j-1} covers a connected part of B_{j-1}.



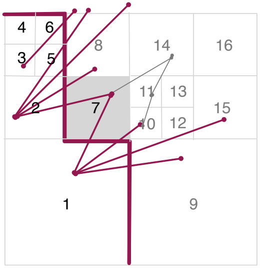
BL6

Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+

Algorithm:

- Sort E^+ by z-index of first endpoint.
- For each interval $I_j = [z_j, z_{j+1}]$ in Q:
  - Find all edges in E^+ that start in I_j: read all $e = (p, q)$ from E^+ s. th. $z(p) \in I_j$
  - Use BL_{j-1} to find the edges that start before σ_j and intersect σ_j, and update BL_{j-1} to BL_j

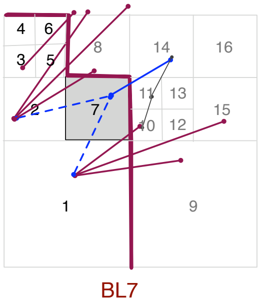


Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+

Algorithm:

- Sort E^+ by z-index of first endpoint.
- For each interval $I_j = [z_j, z_{j+1}]$ in Q:
  - Find all edges in E^+ that start in I_j: read all $e = (p, q)$ from E^+ s. th. $z(p) \in I_j$
  - Use BL_{j-1} to find the edges that start before σ_j and intersect σ_j, and update BL_{j-1} to BL_j

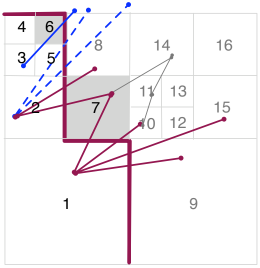


Finding the intersections of E^+ and Q ($k = 1$)

Input: $Q = \{[z_1, z_2], [z_2, z_3], ...\}$ in Z-order, E^+

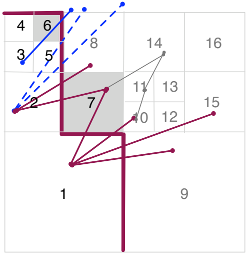
Algorithm:

- Sort E^+ by z-index of first endpoint.
- For each interval $I_j = [z_j, z_{j+1}]$ in Q:
  - Find all edges in E^+ that start in I_j: read all $e = (p, q)$ from E^+ s. th. $z(p) \in I_j$
  - Use BL_{j-1} to find the edges that start before σ_j and intersect σ_j,and update BL_{j-1} to BL_j

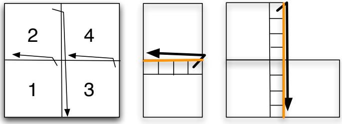


Search in BL_{j-1}: $\Omega(1)$ IOs per cell $\Longrightarrow \Omega(n)$ IOs
Idea: Start scanning BL_{j-1} from an edge that intersects σ_{j-1}

Finding the intersections of E^+ and Q ($k = 1$)



BL6



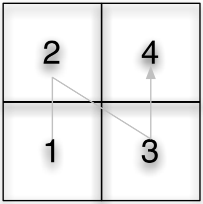
Lemma

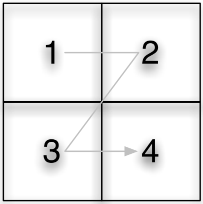
The number of edges traversed and skipped is $O(l)$, where l is the number of edge-cell intersections.

The intersections of E^+ and Q can be found in $O(\mathrm{scan}(n + l))$ IOs, once Q and E^+ are sorted.

Our algorithm

The algorithm generalizes to $k > 1$ and to E^-.





Empirical evaluation

`QDT-K`

- our algorithm for constructing a k-quadtree

Platform

- C
- Intel 2.83 GHz, 5400 rpm SATA drive (HP blade servers)
- 512 MB RAM

Datasets:

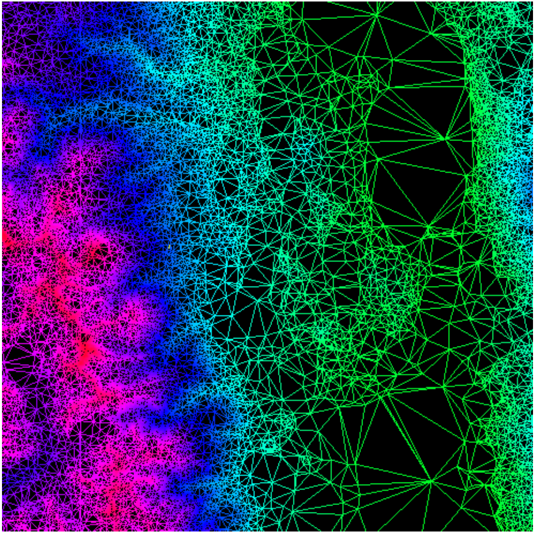
- triangulated terrains (TINs)
- TIGER data

Datasets: TINs

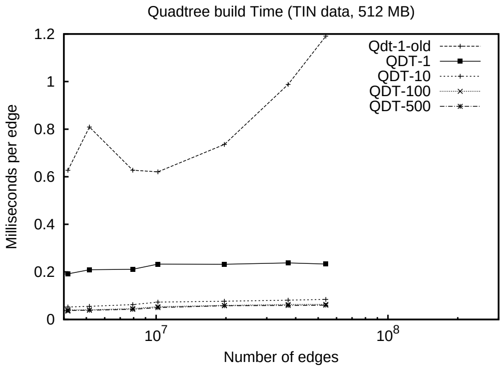
- We ignored the elevation.
- Delaunay triangulation
- Lots of small angles on the boundary

Dataset	e	Max inc.	Min $\angle$
Kaweah	$1.2 \cdot 10^6$	31	.0704
Puerto Rico	$4.1 \cdot 10^6$	291	.0010
Cumberlands	$5.1 \cdot 10^6$	44	.0016
Sierra	$7.9 \cdot 10^6$	75	.0137
Central App.	$10.1 \cdot 10^6$	62	.0013
Hawaii	$19.7 \cdot 10^6$	356	.0007
Haldem	$37.1 \cdot 10^6$	78	.0097
Lower NE	$53.9 \cdot 10^6$	168	.0021

Datasets: TINs



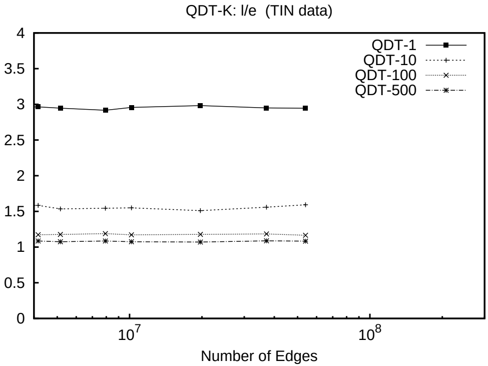
Results: TIN data



- `Qdt-1-old` from De Berg et al. 2010
- As *k* increases: construction time of `QDT-K` decreases

Results: TIN data

QDT−k total size



- As k increases: c decreases, and I/e decreases (fewer cells → fewer edge-cell intersections)

Results: TIN data

Sizes and build time on LowerNE ($e = 53.9 \cdot 10^6$)

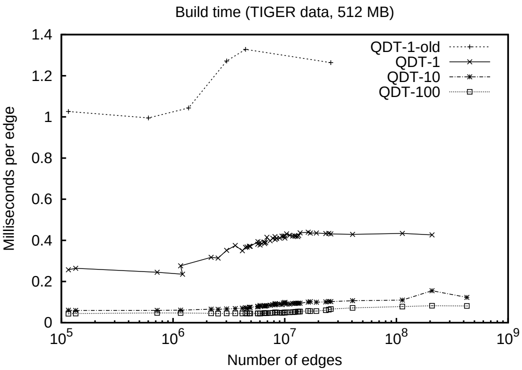
	c	l	l/c	build (min)
QDT-1-OLD	$32.5 \cdot 10^6$	$158.8 \cdot 10^6$	4.8	1,071
QDT-1	$32.5 \cdot 10^6$	$158.8 \cdot 10^6$	4.8	210
QDT-100	$.24 \cdot 10^6$	$62.8 \cdot 10^6$	257.4	57
QDT-500	$.06 \cdot 10^6$	$58.4 \cdot 10^6$	957.4	53
QDT-1000	$.02 \cdot 10^6$	$57.5 \cdot 10^6$	2456.5	54

Datasets: TIGER data

- Available at `http://www.census.gov/geo/www/tiger/`
- 50 sets, one set for each state, containing roads, hydrography, railways and boundaries
- Largest set: TX ($e = 40.4 \cdot 10^6$)
- We assembled larger bundles.

Dataset	e
`New England`	$25.8 \cdot 10^6$
`East Coast`	$113.0 \cdot 10^6$
`Eastern Half`	$208.3 \cdot 10^6$
`All USA`	$427.7 \cdot 10^6$

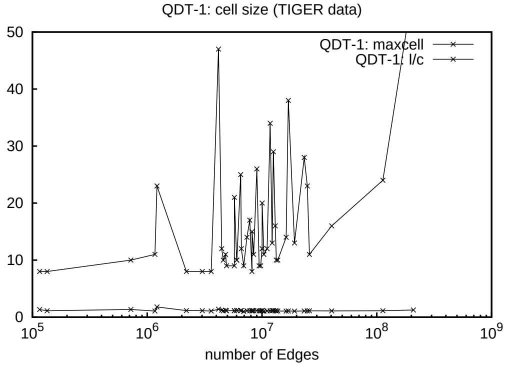
Results: TIGER data



- QDT−k gets faster up to $k = 100$ and then levels
- QDT−100 on AllUSA in 9.7 hours, 70% CPU.
- Bottleneck is finding edge-cell intersections

Results: TIGER data

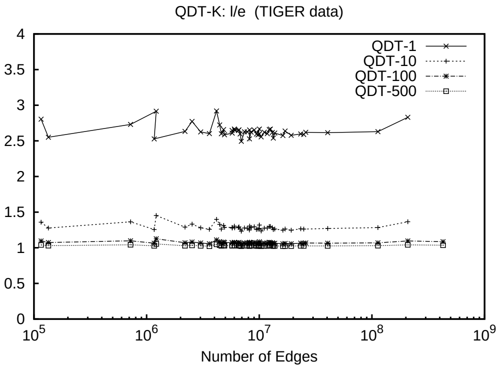
`QDT-1` cell size



- Varies widely from state to state
  - Easthalf: max cell intersects 58 edges
  - ME: max cell intersects 8 edges

Results: TIGER data

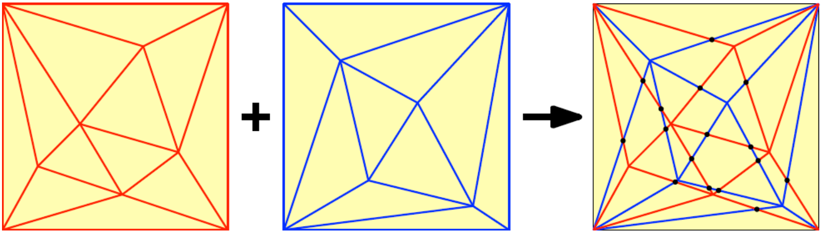
`QDT-k` total size



- As k increases: number of cells decreases, average size of a cell increases, and overall quadtree size decreases

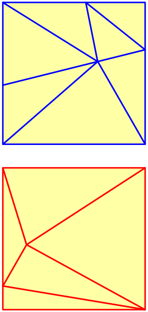
Application: Map overlay

Finding pairwise intersections between two sets of edges

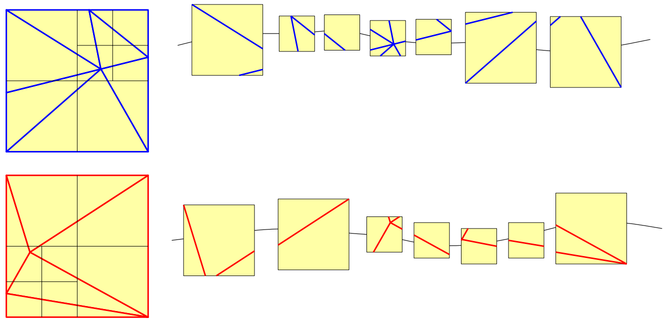


figures thanks to H. Haverkort

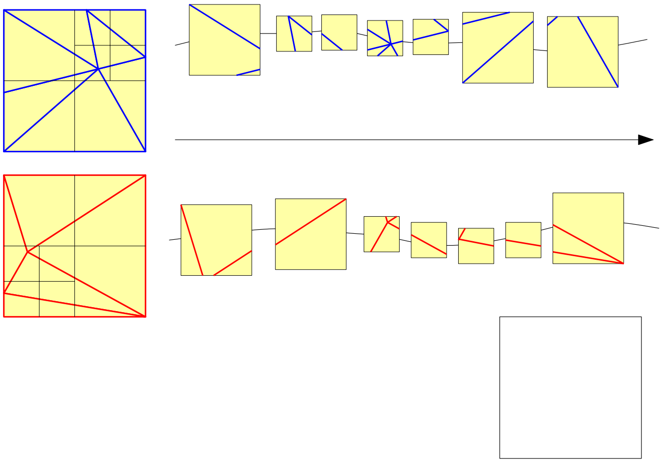
Map overlay with quadtrees in Z-order



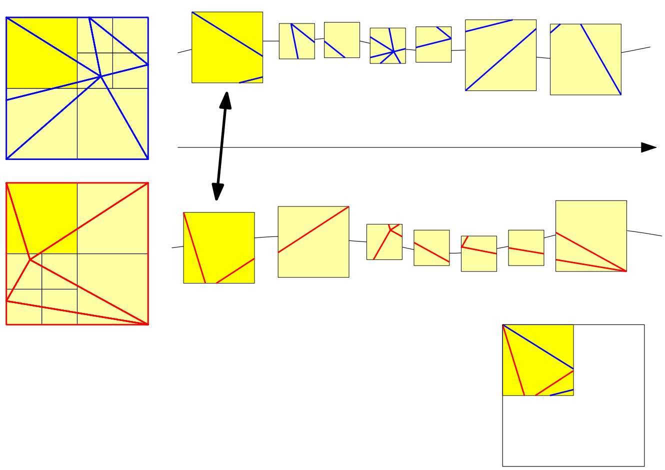
Map overlay with quadtrees in Z-order



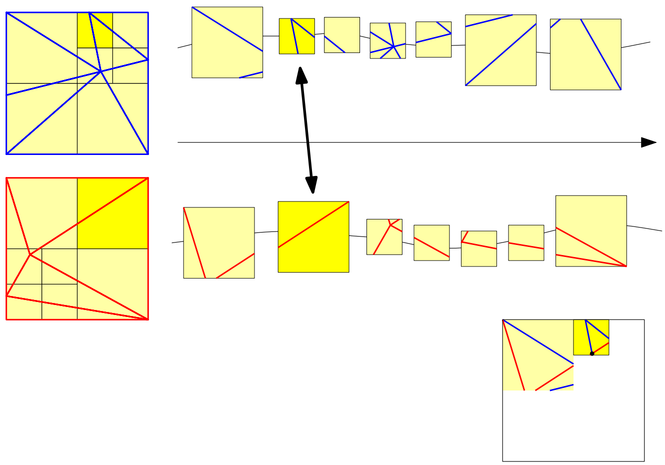
Map overlay with quadtrees in Z-order



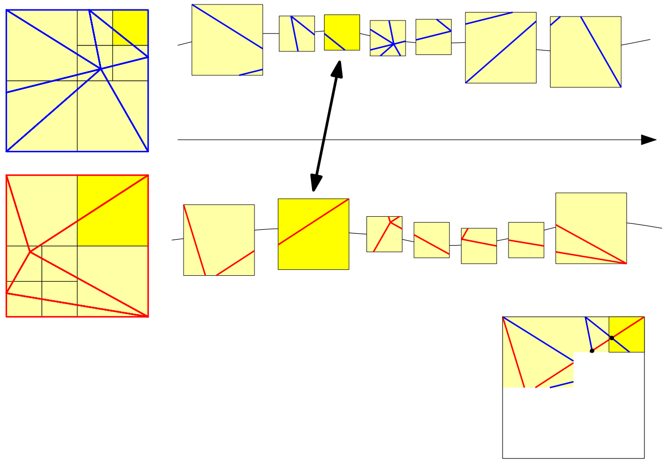
Map overlay with quadtrees in Z-order



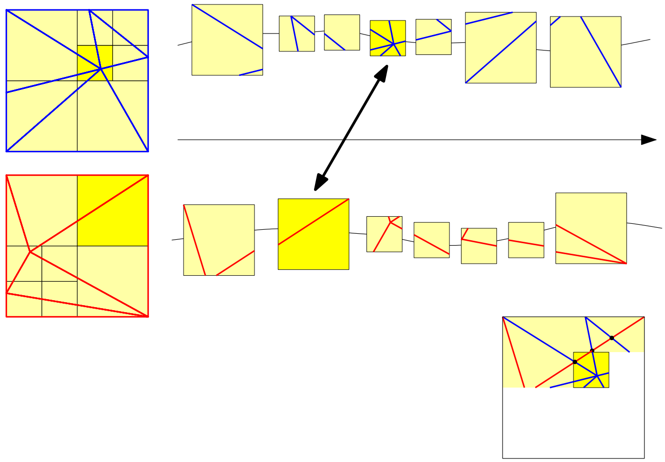
Map overlay with quadtrees in Z-order



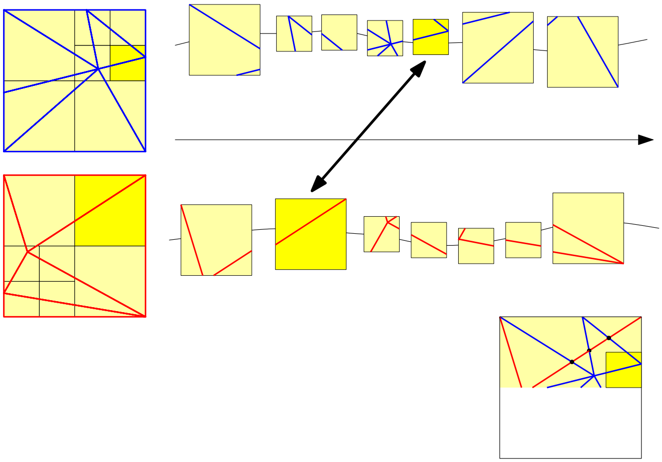
Map overlay with quadtrees in Z-order



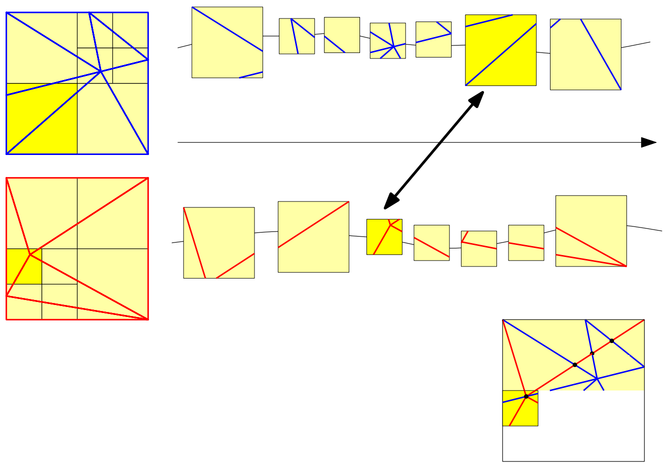
Map overlay with quadtrees in Z-order



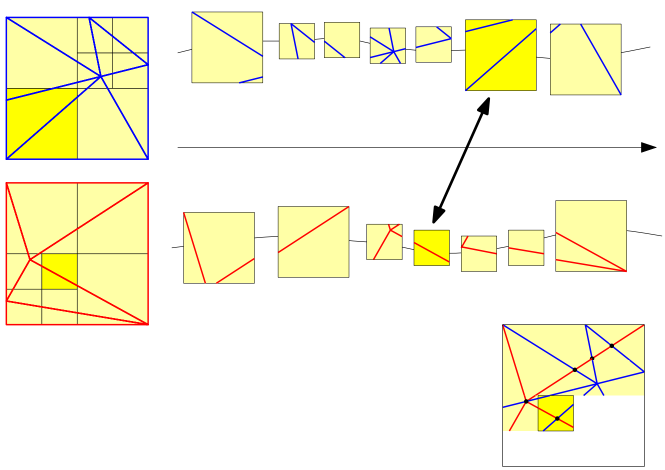
Map overlay with quadtrees in Z-order



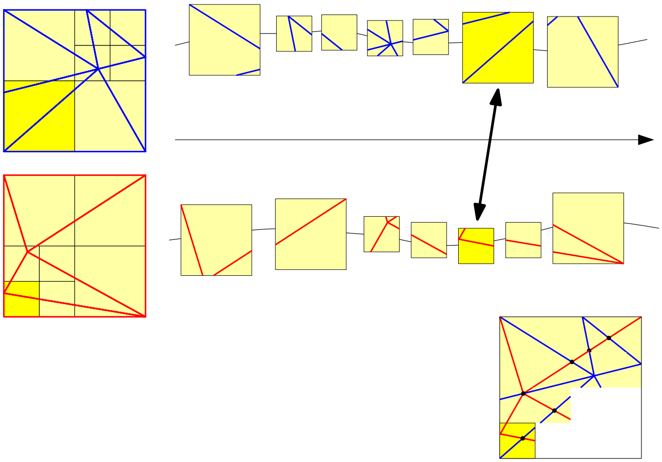
Map overlay with quadtrees in Z-order



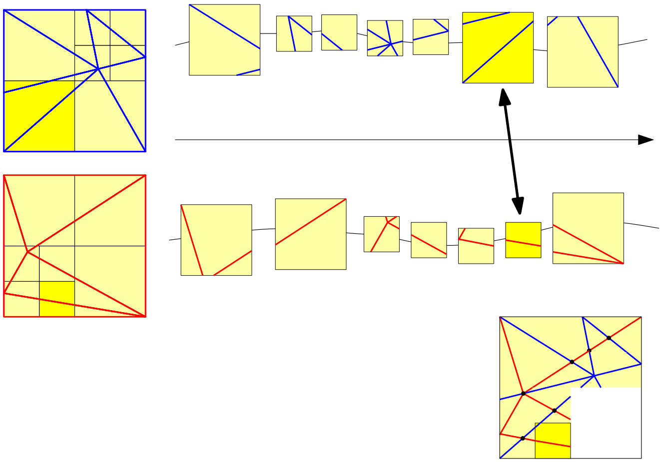
Map overlay with quadtrees in Z-order



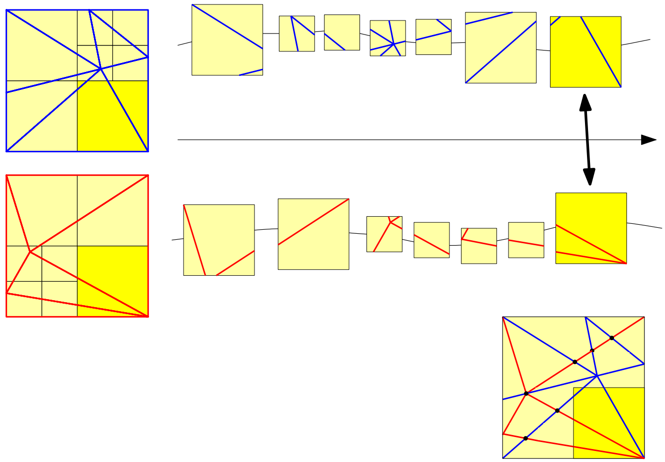
Map overlay with quadtrees in Z-order



Map overlay with quadtrees in Z-order



Map overlay with quadtrees in Z-order



Results: Map overlay with Quadtrees

- We overlayed a TIN stored as `QDT-1` with TIGER data stored as `QDT-k`
  - All data scaled to unit square
  - Artificial results
- Competing effects
  - size of a cell, and time for cell-cell intersection both increase with k
  - nb. of cells decreases with k
- Optimal k: $k \in [100, 500]$
- Overlay 262 mill. edge in 1.8 hours, if quadtrees are given

Conclusions

- New, IO-efficient algorithm for computing K-quadtrees with $O(k)$ vertices per cell, $O(n/k)$ cells, in $O(\text{sort}(n+I))$ IOs.
- Viable for two types of data used in practice
- As k increases, K-quadtrees are faster to compute and have smaller size.
- Optimal value k depends on application

Thank you!