k-nearest neighbor search and range search with kd-trees

Dominika Kubániová

Summary

- Data generation (uniform, normal, skewed normal, exponential, circular), 2D – 4D, max 10^7 points
- Build (sliding-midpoint) and search algorithms
- Visualization of search algorithms
- Time complexity graphs kd-tree vs. naive algorithm

Kd tree build algorithm

- struct kd_node {

  int is_leaf_split_dim;

  int index; // index to left child in array of nodes (internal node)
  // right child at index + 1
  // or index to array of data info (leaf node)

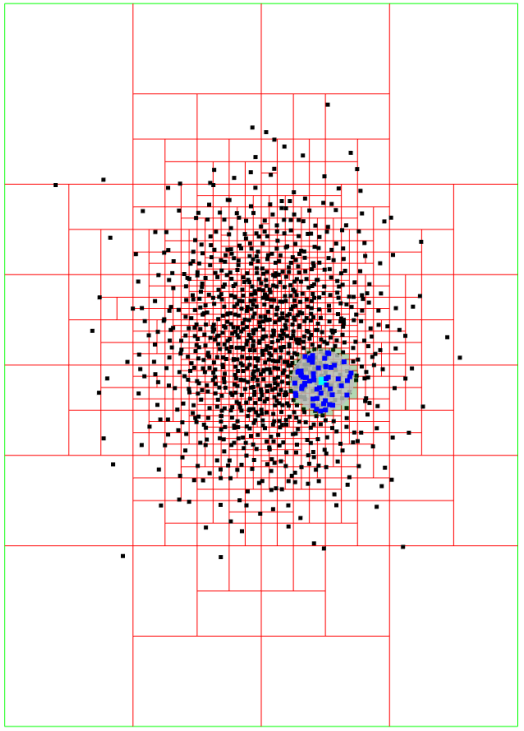
  float split_value;

  }

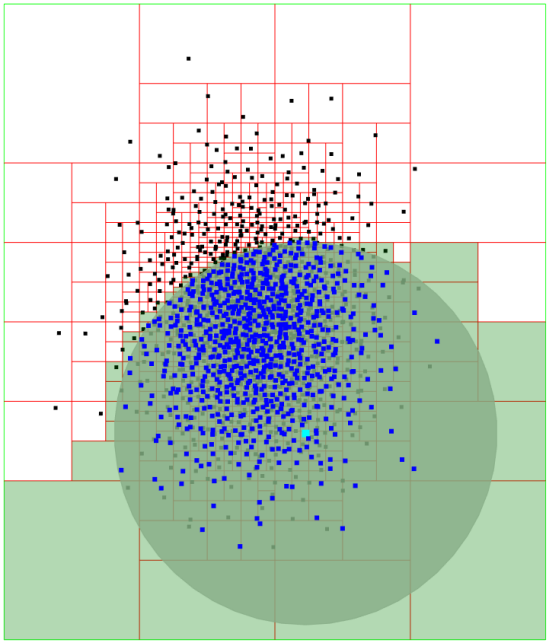
- In total 12 bytes

Build, kNN, spherical and rectangular search visualizations

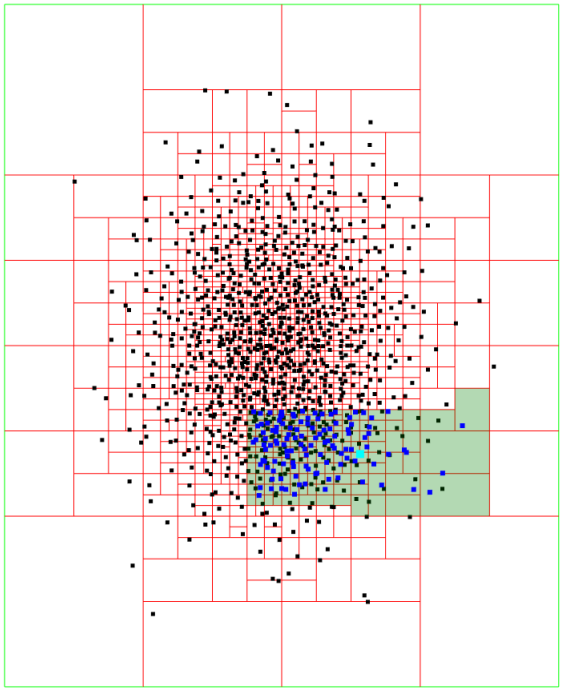
Normal distribution, 100 000 points



k=6000

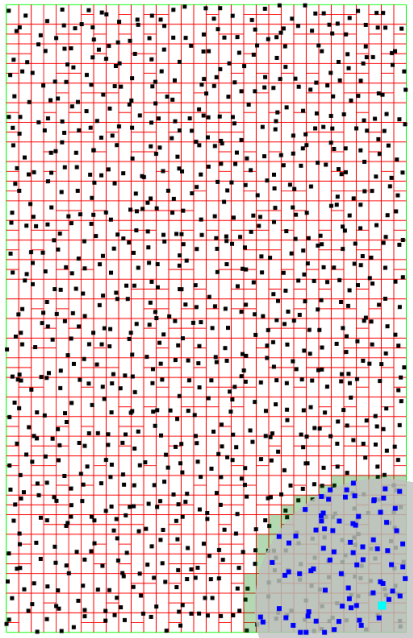


radius=10

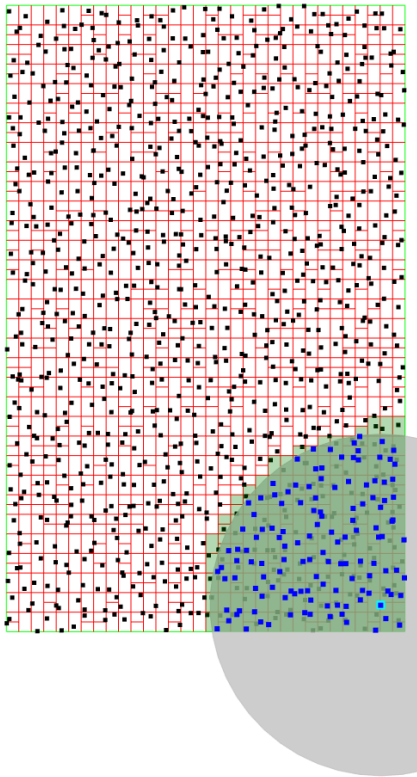


range=x: 5
y: 2

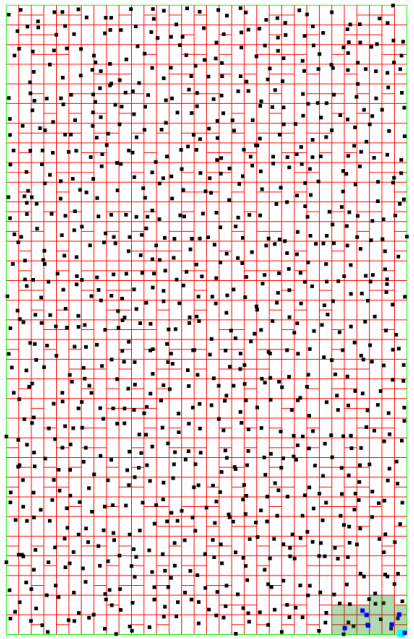
Uniform distribution, 100 000 points



k=8000

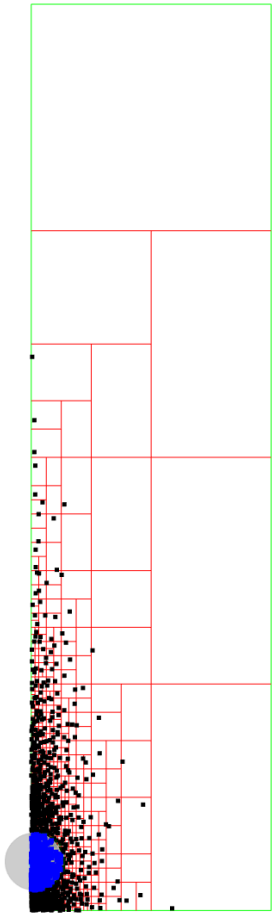


radius=15

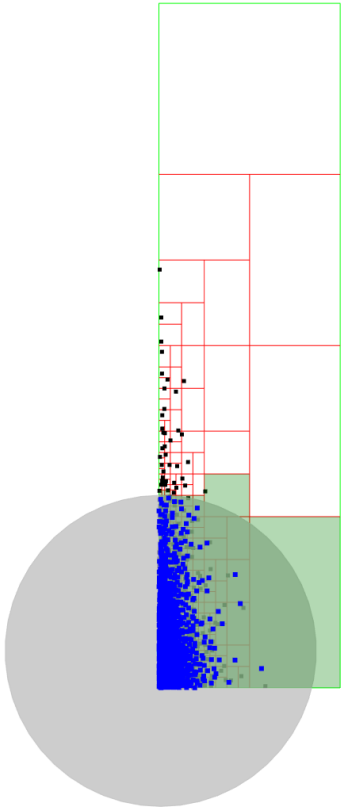


range=x: 5
y: 2

Exponential distribution, 100 000 points

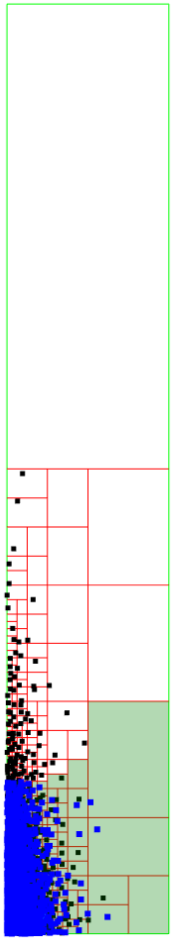


k=3000

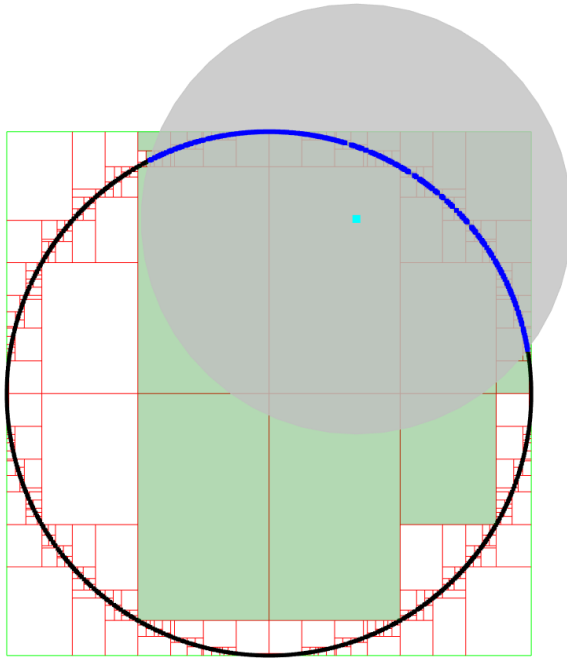


Radius=0.25

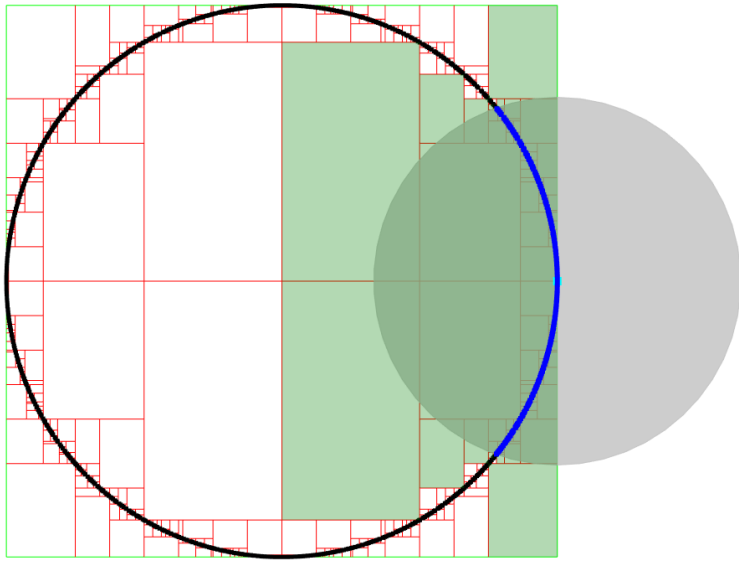
range=x: 0.2
Y: 0.2



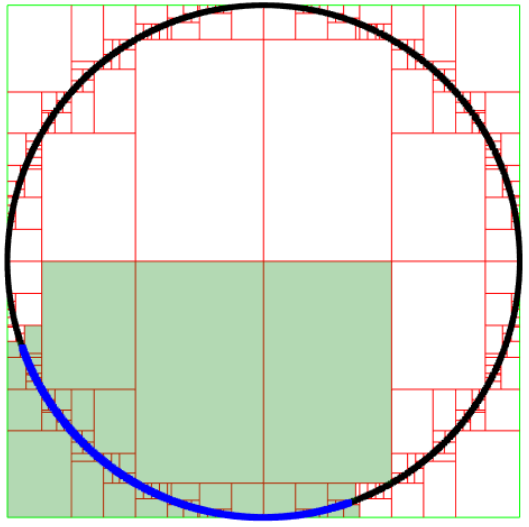
Circle distribution, 100 000 points



k=30000

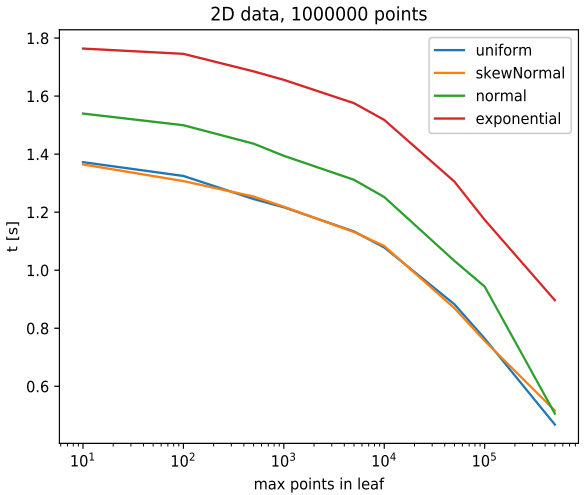


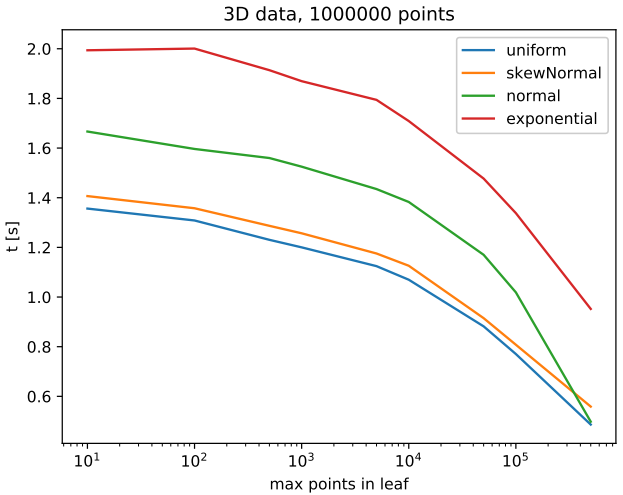
radius=2

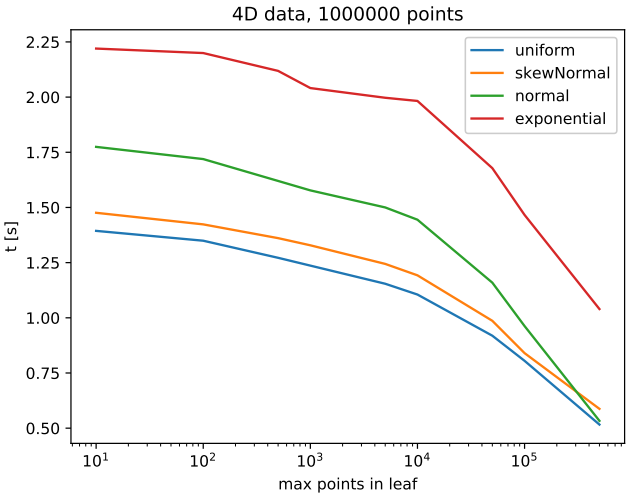


range=x: 5
y: 2

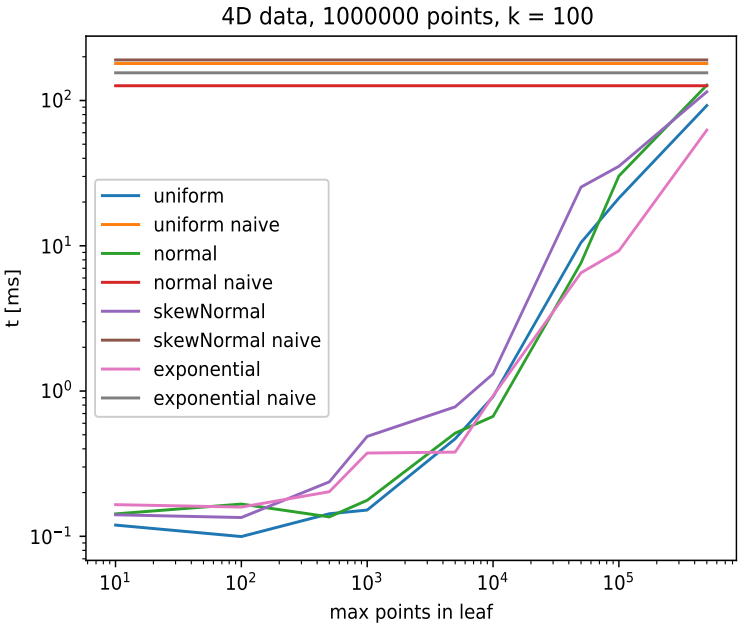
Build time complexity x number of point in a leaf

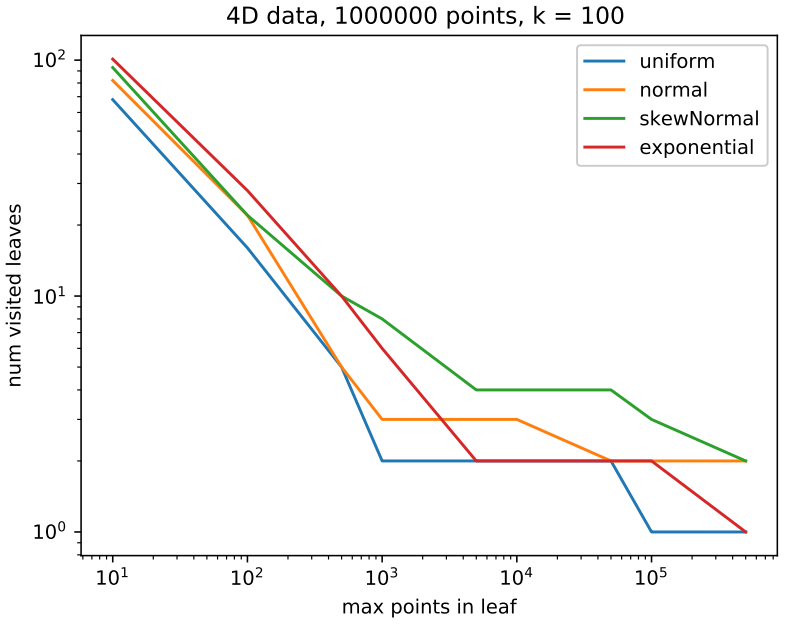




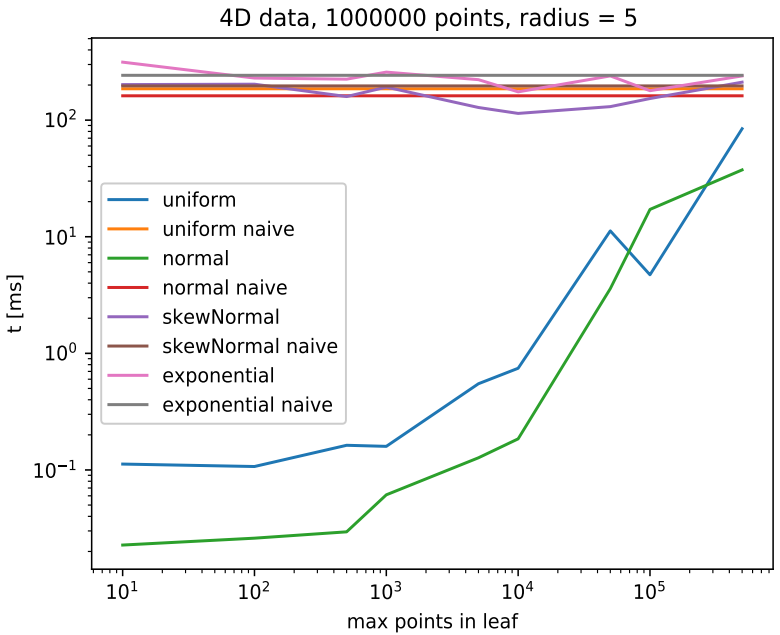


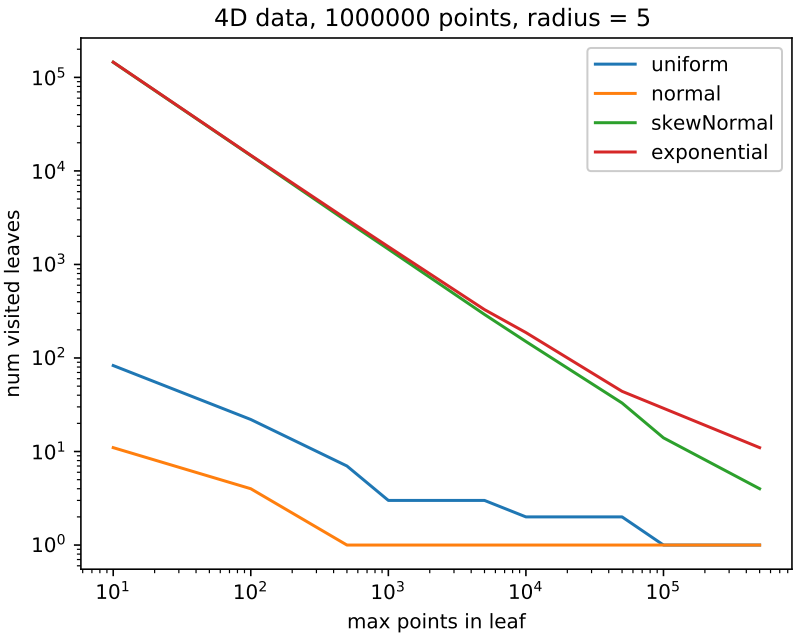
kNN search, kd vs. naive, priority queue



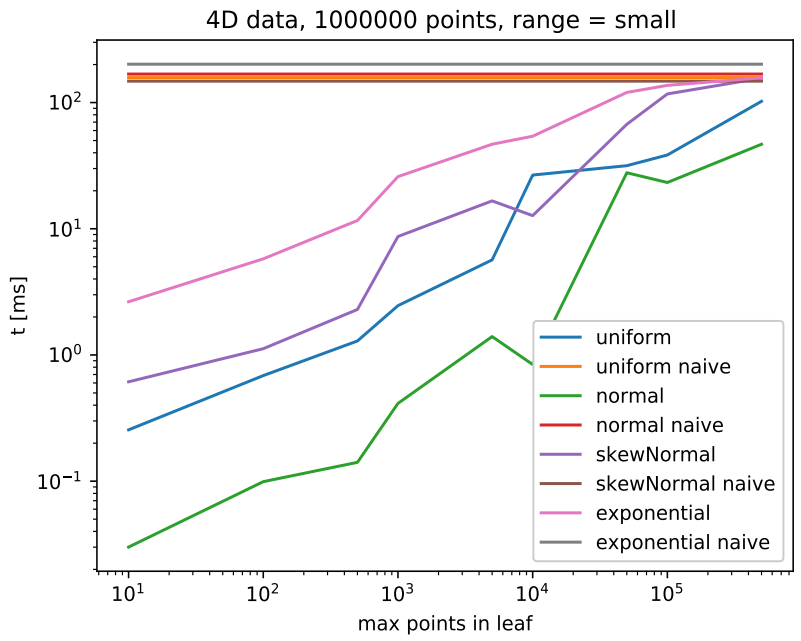


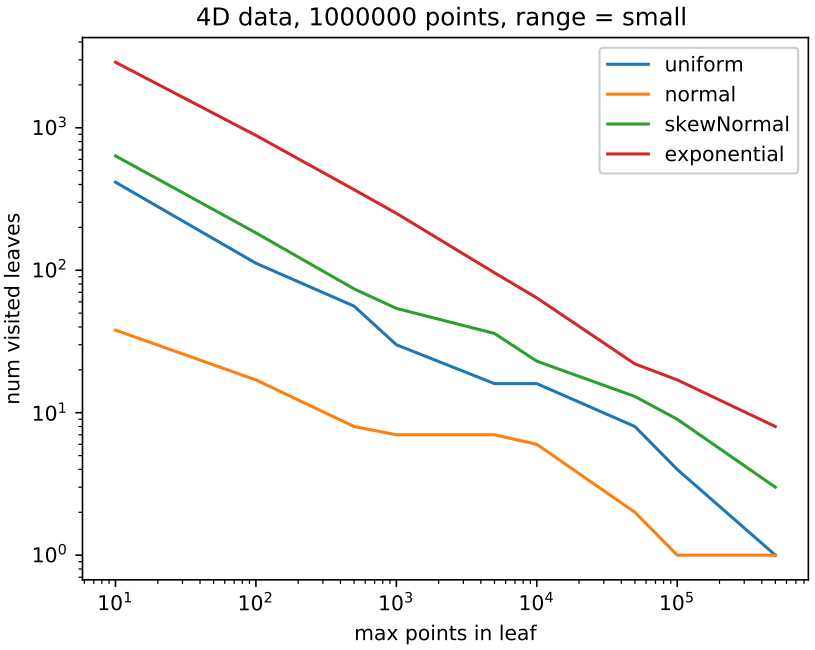
spherical search, kd vs. naive





rectangular search, kd vs. naive





Thank you for you attention