A well-separated pairs decomposition algorithm for kd-trees implemented on multi-core architectures

Raul H. C. Lopes
Ivan D. Reid
Peter R. Hobson

Particle Physics Group, School of Engineering and Design, Brunel University

October 17, 2013

Table of Contents

Motivation

- A problem
  - Given a set P of n points in R^d:
    - find the two closest points to each other belonging to P (e.g. Eppstein);
    - for each $q \in P$, find its closest neighbour in $P - q$ [6];
    - find all k nearest neighbours of each $q \in P$ [4].

Motivation

- A problem
  - Given a set P of n points in R^d:
    - find the two closest points to each other belonging to P (e.g. Eppstein);
    - for each $q \in P$, find its closest neighbour in $P - q$ [6];
    - find all k nearest neighbours of each $q \in P$ [4].
- Theoretical limits
  - all solvable in $O(n \log n)$ work **if** an $O(n \log n)$ work algorithm spatial indexing is available.
  - parallel algorithms using p processors and $O(n \log n)$ work theoretically available (Callahan in [4]).

Good for HEP and Physics?

- Multivariate analysis for TMVA in ROOT uses k nearest neighbours search. Repeated analysis might benefit of k-d-tree algorithm with $O(n \log n)$ work with decent scalability.
- Track reconstruction by joining compatible triplets has been approached in CMS using k-d-trees and cellular automata. Even the the simulation of celular automata might demand multidimensional data organisation when the number of dimensions increase.
- N-body computations based on Barnes-Hutt or Fast Multipole Method in general depend on tree methods.
- N-point correlation functions have been tackled in Astronomy by the use of search in k-d-trees [5].

Proposed solution

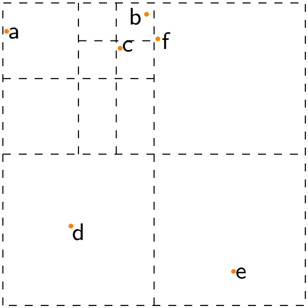
- K-d-tree based solution built on
  - fair split that will lead to *Well Separated Pair Decomposition*;
  - balanced partitioning giving $O(\log n)$ height independent of input;
  - work balanced partition and tree construction.

Proposed solution

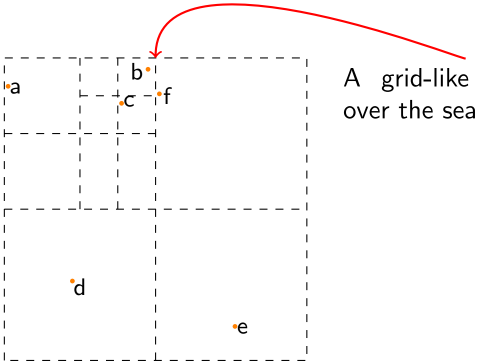
- K-d-tree based solution built on
  - fair split that will lead to *Well Separated Pair Decomposition*;
  - balanced partitioning giving $O(\log n)$ height independent of input;
  - work balanced partition and tree construction.
- Implementations in the paper
  - First implementation of a parallel **WSPDP** algorithm.
  - Possibly first implementation of a parallel k-d-tree (Lisp, C/OMP, C++/TBB.)
  - Parallel scalable implementation of Kth-selection algorithm

Table of Contents

Quadtree (for set P of n points in R^d)

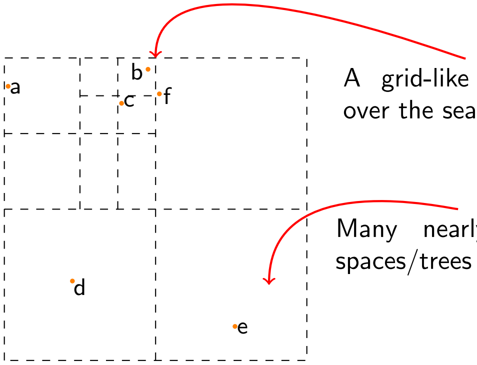


Quadtree (for set P of n points in R^d)



A grid-like structure over the search space

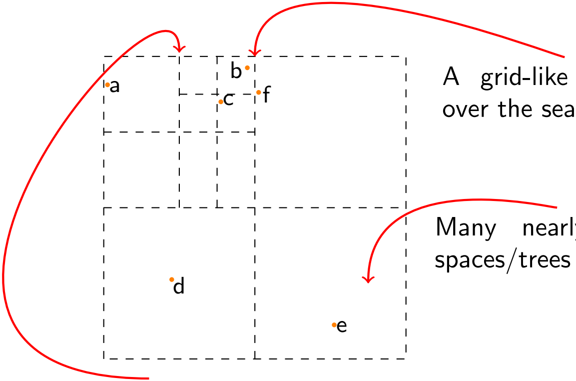
Quadtree (for set P of n points in R^d)



A grid-like structure over the search space

Many nearly empty spaces/trees

Quadtree (for set P of n points in R^d)

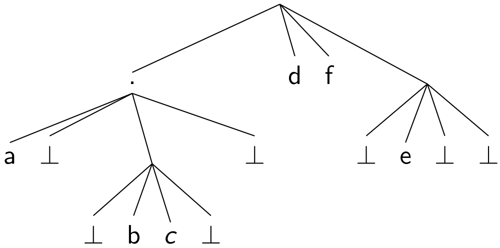


A grid-like structure over the search space

Many nearly empty spaces/trees

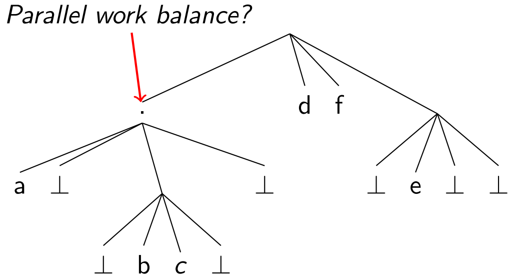
a difficult act to balance parallel work for unequal regions

Quadtrees: a choice for parallel spatial indexing?



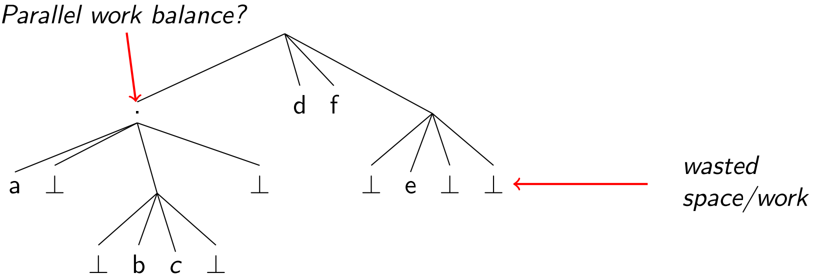
- each node partitions a set of points by all *d* attributes, each representing one dimension. partitions for the set in question
- Samet [8]: used by *all* published works on parallel spatial indexing.
- disadvantages:
  - curse of dimensions: number of empty (or nearly empty) partitions increasing fast with dimensions.
  - hard to balance the processors' work.

Quadtrees: a choice for parallel spatial indexing?



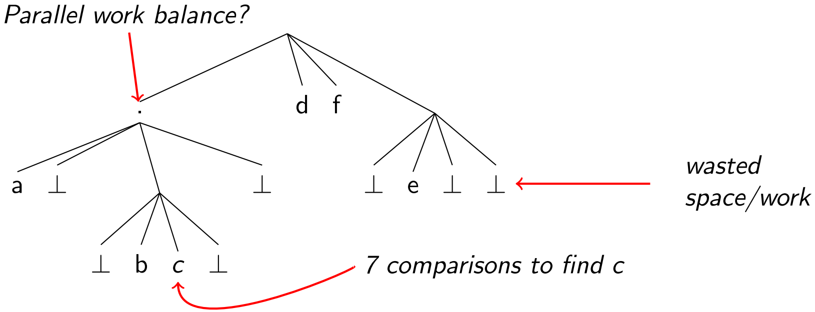
- each node partitions a set of points by all *d* attributes, each representing one dimension. partitions for the set in question
- Samet [8]: used by *all* published works on parallel spatial indexing.
- disadvantages:
  - curse of dimensions: number of empty (or nearly empty) partitions increasing fast with dimensions.
  - hard to balance the processors' work.

Quadtrees: a choice for parallel spatial indexing?



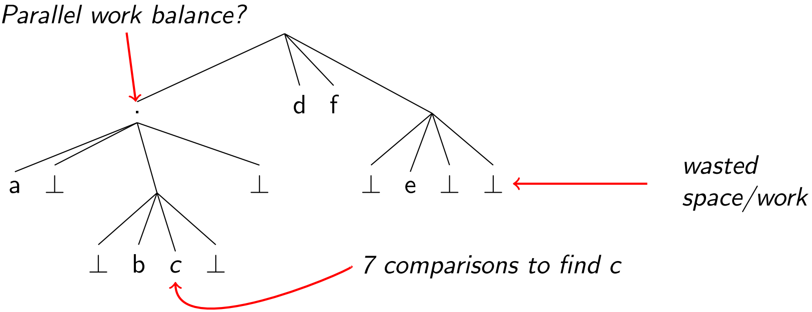
- each node partitions a set of points by all *d* attributes, each representing one dimension. partitions for the set in question
- Samet [8]: used by *all* published works on parallel spatial indexing.
- disadvantages:
  - curse of dimensions: number of empty (or nearly empty) partitions increasing fast with dimensions.
  - hard to balance the processors' work.

Quadtrees: a choice for parallel spatial indexing?



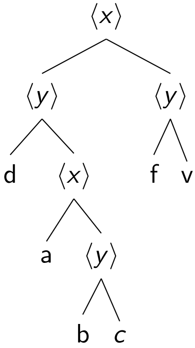
- each node partitions a set of points by all *d* attributes, each representing one dimension. partitions for the set in question
- Samet [8]: used by *all* published works on parallel spatial indexing.
- disadvantages:
  - curse of dimensions: number of empty (or nearly empty) partitions increasing fast with dimensions.
  - hard to balance the processors' work.

Quadtrees: a choice for parallel spatial indexing?

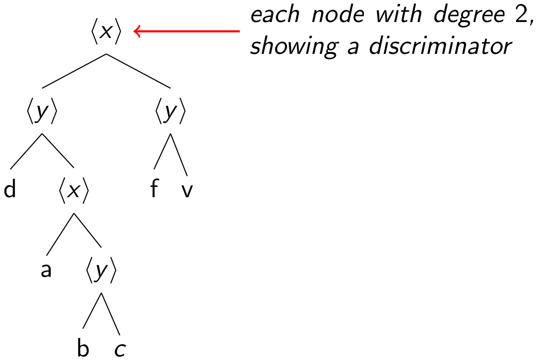


- each node partitions a set of points by all *d* attributes, each representing one dimension. partitions for the set in question
- Samet [8]: used by *all* published works on parallel spatial indexing.
- disadvantages:
  - curse of dimensions: number of empty (or nearly empty) partitions increasing fast with dimensions.
  - hard to balance the processors' work.

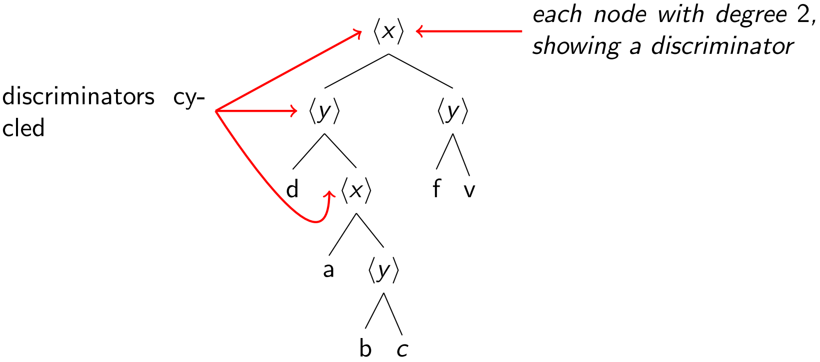
Bentley's K-d-trees



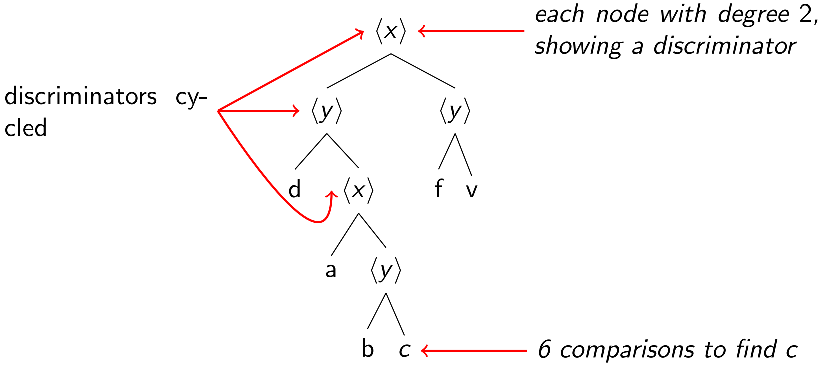
Bentley's K-d-trees



Bentley's K-d-trees



Bentley's K-d-trees



K-d-trees: questions

- K-d-tree by Jon Bentley [2]
    - Each nodes defines a discriminator (splitting dimension)
    - Each discriminator has an associated cut value: the dimension value of the points in the subset being partitioned
    - discriminators cycle through the *k* dimension on the path from root to leaf nodes

K-d-trees: questions

- K-d-tree by Jon Bentley [2]
  - Each nodes defines a discriminator (splitting dimension)
  - Each discriminator has an associated cut value: the dimension value of the points in the subset being partitioned
  - discriminators cycle through the k dimension on the path from root to leaf nodes
- Disadvantages
  - cycling through dimensions can still lead to leaf depth unbalancing
  - choice of cut value, a problem to be solved
  - distribution of work on $p > 1$ processors scenario can be complicated due to recursive nature
  - work balancing still a problem

Table of Contents

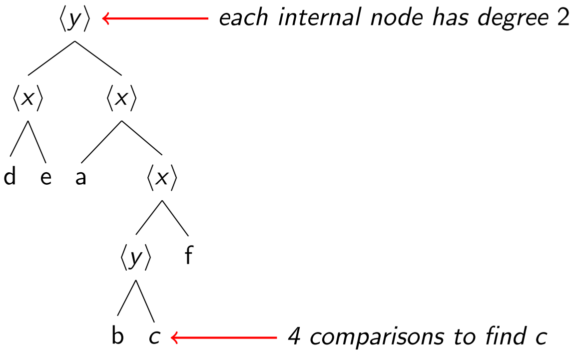
Fair split based K-d-trees (Callahan)

- Splitting criteria
  - each node defines a range of the space
  - split close to the middle of the range
  - split should guarantee either:
    - balanced distribution of points for children nodes
    - or total of comparisons on the order of the size of the node being split
- Fair split target balanced number of comparisons in sequential split
  - $O(n \log n)$ time for sequential algorithm
  - split adapts Bentley's "burning the candle from both ends" algorithm [1].

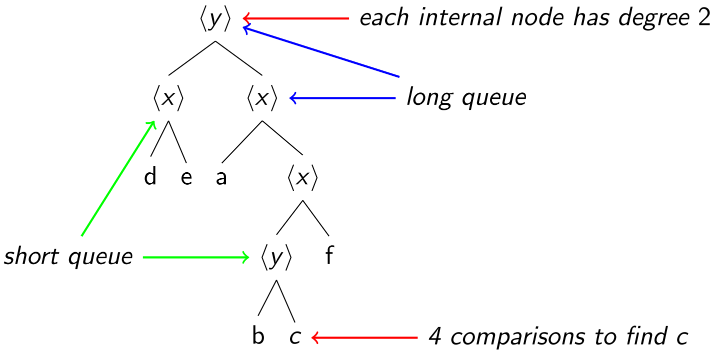
Fair split based K-d-trees (Callahan)

- Splitting criteria
  - each node defines a range of the space
  - split close to the middle of the range
  - split should guarantee either:
    - balanced distribution of points for children nodes
    - or total of comparisons on the order of the size of the node being split
- Fair split target balanced number of comparisons in sequential split
  - $O(n \log n)$ time for sequential algorithm
  - split adapts Bentley's "burning the candle from both ends" algorithm [1].
- Problem for parallel algorithm
  - split must "guess" (or brute force search) slabs of splitting
  - Callahan does "hand waving argument" to show that it is possible
  - Har-Peled [6] proposes splitting based on one of
    - radix splitting
    - k-enclosing disk splitting

Proposed solution



Proposed solution



Parallel K-d-tree build guarantees

- Variation from Bentley's K-d-tree structure
  - each node defines cutting discriminator
  - discriminator chosen as largest dimension available for splitting
  - cut value chosen by a median of medians algorithm

Parallel K-d-tree build guarantees

- Variation from Bentley's K-d-tree structure
  - each node defines cutting discriminator
  - discriminator chosen as largest dimension available for splitting
  - cut value chosen by a median of medians algorithm
- Building algorithm
  - median of medians algorithm uses $O(n)$ work and scales for p processors
  - heuristic variation of medians algorithm guarantees $O(n)$ (with small constant) splitter located between $\frac{3n}{10}$ and $\frac{7n}{10}$, giving logarithmic height

Parallel K-d-tree build guarantees

- Variation from Bentley's K-d-tree structure
  - each node defines cutting discriminator
  - discriminator chosen as largest dimension available for splitting
  - cut value chosen by a median of medians algorithm
- Building algorithm
  - median of medians algorithm uses $O(n)$ work and scales for p processors
  - heuristic variation of medians algorithm guarantees $O(n)$ (with small constant) splitter located between $\frac{3n}{10}$ and $\frac{7n}{10}$, giving logarithmic height
- asynchronously parallel algorithm: full use of all processors as they become available.

Splitting for balanced k-d-tree

- Split close to the median.
  - Based on Blum, Floyd, Pratt, Tarjan algorithm of kth-selection
  - Parallel map blocks of 5 elements to its median
  - Recursion until block of up to 5 central elements is found.
  - Easy elimination of recursion.
  - Guaranteed time in $O(n)$.
  - Median of final blocks always greater than $\frac{3n}{10}$ and less than $\frac{7n}{10}$ elements of initial set.
  - Additional element from final block used for a two pivots split.
- Option for split at the middle of the range based on A. Moore [7].

Parallel k-d-tree algorithm

- Sets of nodes to split are kept in two queues.

Parallel k-d-tree algorithm

- Sets of nodes to split are kept in two queues.
- Long queue split:
  - all processors all applied to split one node
  - split in three steps
    - parallel search for dimension to split
    - parallel search for splitters using median of 5
    - parallel split adapting Bentley's invariant for one pivot to a two pivot split.
  - each split node will always produce two or three children.
  - at least two nodes resulting from one split will have a minimum of $\frac{3n}{10}$ points each. $O(\log n)$ height guaranteed.

Parallel k-d-tree algorithm

- Sets of nodes to split are kept in two queues.
- Long queue split:
  - all processors all applied to split one node
  - split in three steps
    - parallel search for dimension to split
    - parallel search for splitters using median of 5
    - parallel split adapting Bentley's invariant for one pivot to a two pivot split.
  - each split node will always produce two or three children.
  - at least two nodes resulting from one split will have a minimum of $\frac{3n}{10}$ points each. $O(\log n)$ height guaranteed.
- Short queue split:
  - Each process available takes one node to split with same algorithm as long queue.

Performance

- Based on gcc (-fopenmp) 4.6, Ubuntu 13.04, Intel xeon 5660.
- Sets with up to 2^{16} points processed asynchronously with one processor.
- Speed-up and efficiency shown in the table only for the 1572864 set.

points in 3-d	1 proc	4 procs	8 procs	12 procs
65536	0.856s	0.868s	0.857s	0.854s
252144	3.206s	1.337s	1.235s	1.391s
524288	6.551s	2.545s	1.338s	1.512s
786432	9.724s	3.991s	2.781s	1.862s
1572864	18.43s	7.515s	4.532s	3.623s
S_p	1	2.45	4.14	5.08
E_p	1	0.61	0.51	0.42

- Speed-up $S_{12} = \frac{T_{12}}{T_1}$
- Efficiency $E_{12} = \frac{S_{12}}{12}$

Table of Contents

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?
  - Cut by the middle of the largest range is available.
  - (Extensive) Testing needed.

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?
  - Cut by the middle of the largest range is available.
  - (Extensive) Testing needed.
- Long queue move done in two steps that should be improved by fusion: local three-way split based on Bentley followed by pack pattern.

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?
  - Cut by the middle of the largest range is available.
  - (Extensive) Testing needed.
- Long queue move done in two steps that should be improved by fusion: local three-way split based on Bentley followed by pack pattern.
  - Can we fuse the two steps into a categorization pattern?
  - *Careful! Categorization pattern can be very costly!*

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?
  - Cut by the middle of the largest range is available.
  - (Extensive) Testing needed.
- Long queue move done in two steps that should be improved by fusion: local three-way split based on Bentley followed by pack pattern.
  - Can we fuse the two steps into a categorization pattern?
  - *Careful! Categorization pattern can be very costly!*
- Is *WSPDP* too heavy?

What could possibly go wrong?

- Too many comparisons: what if median search is too expensive and cut by the middle is good enough in practice?
  - Cut by the middle of the largest range is available.
  - (Extensive) Testing needed.
- Long queue move done in two steps that should be improved by fusion: local three-way split based on Bentley followed by pack pattern.
  - Can we fuse the two steps into a categorization pattern?
  - *Careful! Categorization pattern can be very costly!*
- Is *WSPDP* too heavy?
  - Yes, when number of dimensions increase [6].
  - Sequential approximation algorithm available. Parallel?

Wrong with parallel model?

- Parallelism too irregular.

Wrong with parallel model?

- Parallelism too irregular.
  - Algorithm described unsuitable for strict SIMD of most GPUs.

Wrong with parallel model?

- Parallelism too irregular.
  - Algorithm described unsuitable for strict SIMD of most GPUs.
  - Scan model of computing and Blelloch's radix sort pattern [3] to help.
  - Local SIMD available, but so far not exploited: widest range search, median computation, local three-way split.

Wrong with parallel model?

- Parallelism too irregular.
  - Algorithm described unsuitable for strict SIMD of most GPUs.
  - Scan model of computing and Blelloch's radix sort pattern [3] to help.
  - Local SIMD available, but so far not exploited: widest range search, median computation, local three-way split.
- Parallel processing in the long queue not decoupled enough for distributed memory will demand (maybe too many) data movements.

Wrong with parallel model?

- Parallelism too irregular.
  - Algorithm described unsuitable for strict SIMD of most GPUs.
  - Scan model of computing and Blelloch's radix sort pattern [3] to help.
  - Local SIMD available, but so far not exploited: widest range search, median computation, local three-way split.
- Parallel processing in the long queue not decoupled enough for distributed memory will demand (maybe too many) data movements.
  - Extensive testing/tuning for algorithm with an initial phase of sampling to distribute points when running on cluster.
  - MIC computation could be more feasible due to lower communication costs.

Conclusion

- The simple structure of k-d-trees offers promising alternatives regarding:
  - Restricting the height of resulting trees.
  - balancing of work load in parallel implementation.
- Challenges that need to be worked:
  - Irregular parallelism of present algorithm not the best regular SIMD as found in GPUs.
  - Improving memory management in parallel execution might result in huge gains in computing time and efficiency.
  - Scheduling of synchronised steps affects time and efficiency.

Jon Bentley.
Programming Pearls.
Addison Wesley, 1999.

Jon L. Bentley.
Multidimensional binary search trees used for associative searching.
Communications of ACM, 1975.

Guy E. Blelloch.
Preffix sums and their applications.
Technical Report CMU-CS-90-190, School of Computer Science – Carnegie Mellon University, 1990.

Paul B. Callahan.
Dealing with Higher Dimensions: The Well-Separated Pair Decomposition and Its Applications.
PhD thesis, John Hopkins University, 1995.

References II

Andrew William Moore et al.
Fast algorithms and efficient statistics: N-point correlation functions.
In *Proceedings of MPA/MPE/ESO Conference Mining the Sky*, 2000.

Sariel Har-Peled.
Geometric Approximation Algorithms.
American Mathematical Society, 2011.

Andrew William Moore.
Efficient memory-based learning for robot control.
Technical Report UCAM-CL-TR-209, University of Cambridge, 1990.

Hanan Samet.
Foundations of Multidimensional and Metric Data Structures.
Morgan Kaufman, 2006.