TOUCH: In-Memory Spatial Join by Hierarchical Data-Oriented Partitioning

A. Logins

Moscow Institute of Physics and Technology
Skolkovo Institute of Science and Technology

Course: Machine Learning and Data Analysis
(Strijov's practice)/Group 174, 2014 Fall

Goal of research

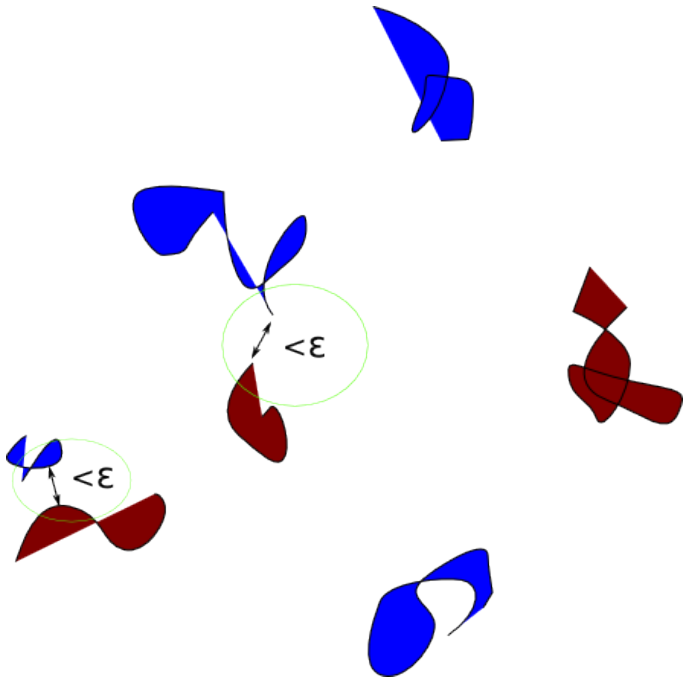
Motivation

- Finding close objects with complex shape in space from large datasets can be a hard problem, that requires effective indexing of data and search algorithms
- State-of-art solutions are restrained by taking into consideration time for reading data from main memory (hard disc)

+ Recent research TOUCH is designed for in-memory join and performs much faster
- It suffers from strong unbalance inside data structure used for indexing, which leads to extreme dependency from data distribution and prevents from effective parallelization.
Fixing the design of TOUCH would give new super-fast spatial join algorithm.

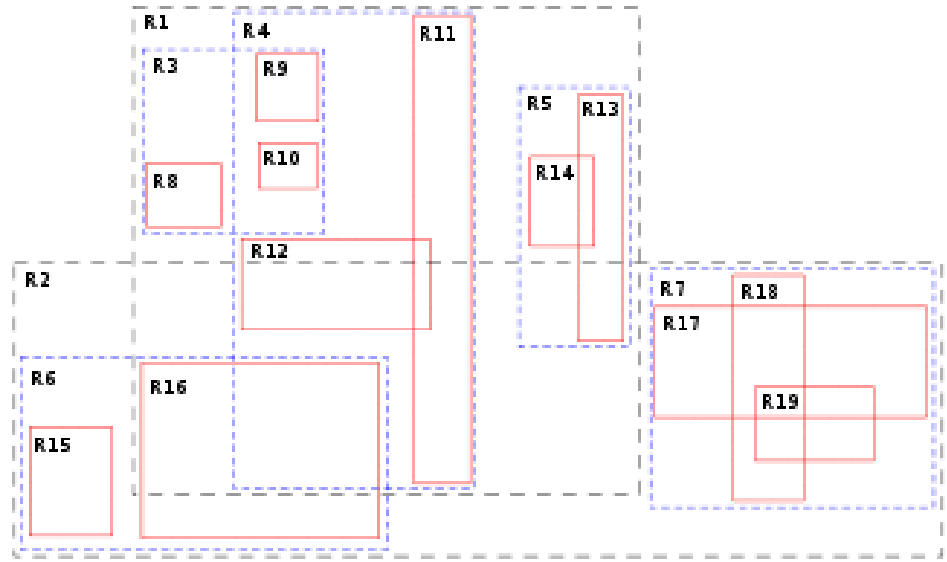
Problem of spatial join

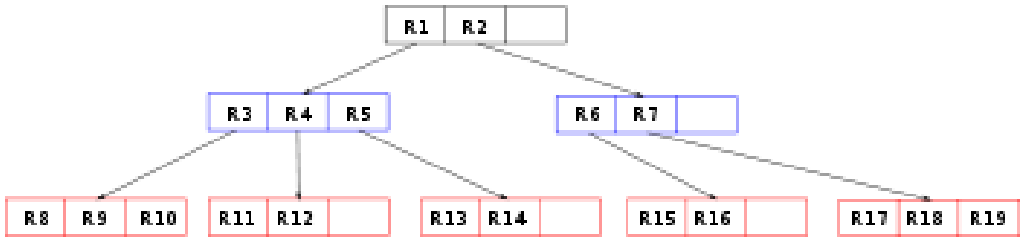
Given the parameter ϵ and two datasets of spatial objects A and B find all $a \in A$ and $b \in B$ such that minimum distance between them is less than ϵ.



Solution : Building step

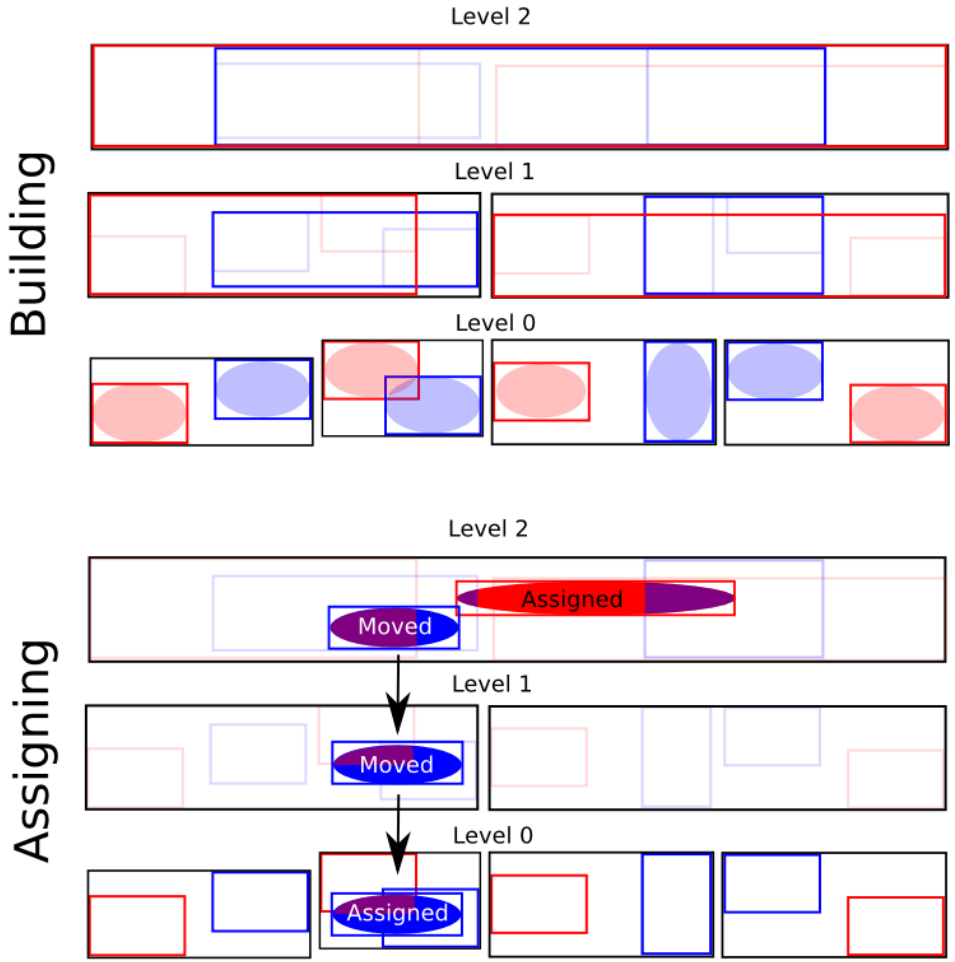
First step: building spatial index structure (R-tree) using objects of first type





Solution : Assignment step

Second step: assigning objects of second type to the nodes of the tree



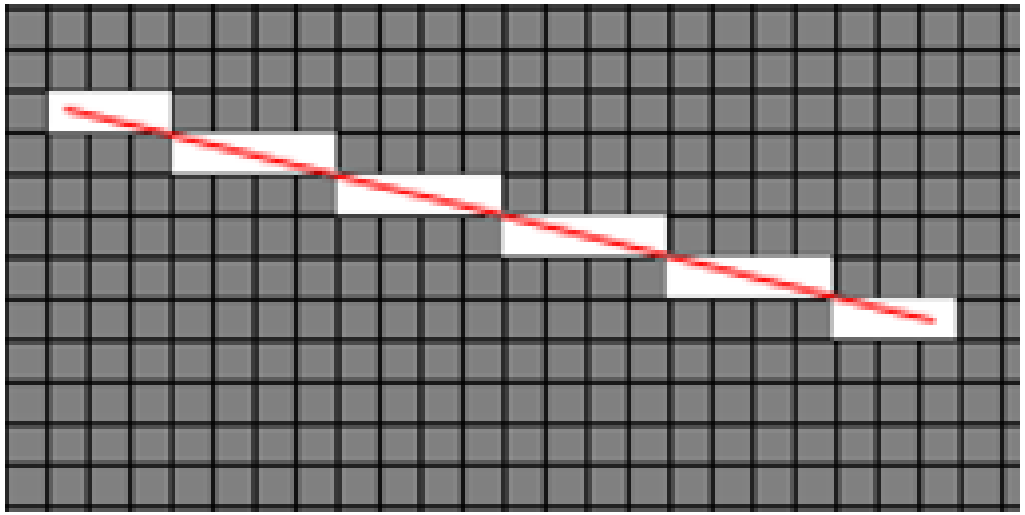
Solution : Assignment step (variations)

Second step: variations

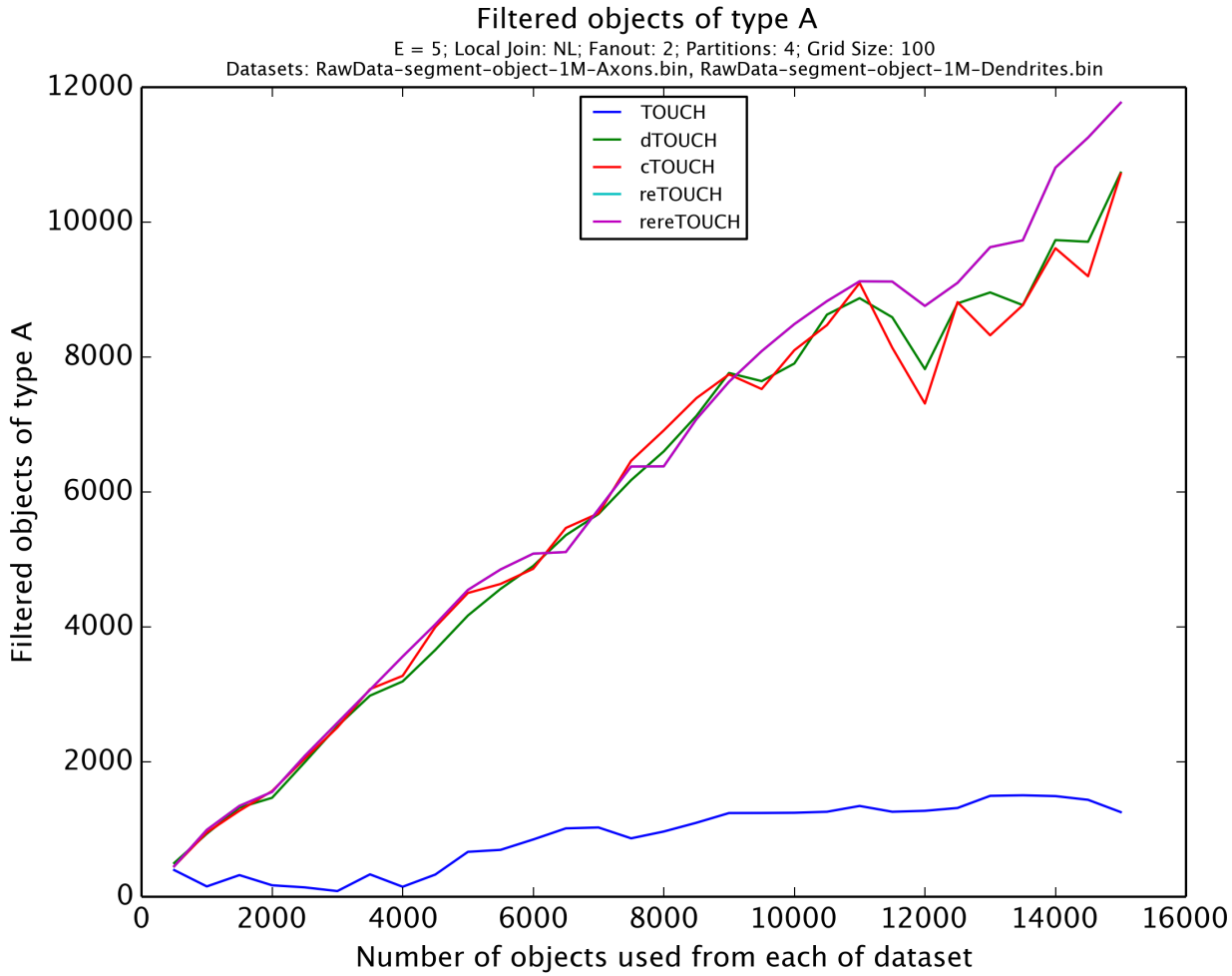
- (dTOUCH) do not assign higher than some level. Build another tree for skipped objects
- (cTOUCH) build initial index using both data types simultaneously. Assign objects that were cut from leaf nodes. Dynamically fix MBRs
- (reTOUCH) after usual assignment step remove all objects of first dataset, fix MBRs (build new tree using old skeleton) of left objects and reassign removed objects.
- (rereTOUCH) repeat one more reTOUCH iteration

Solution : Joining step

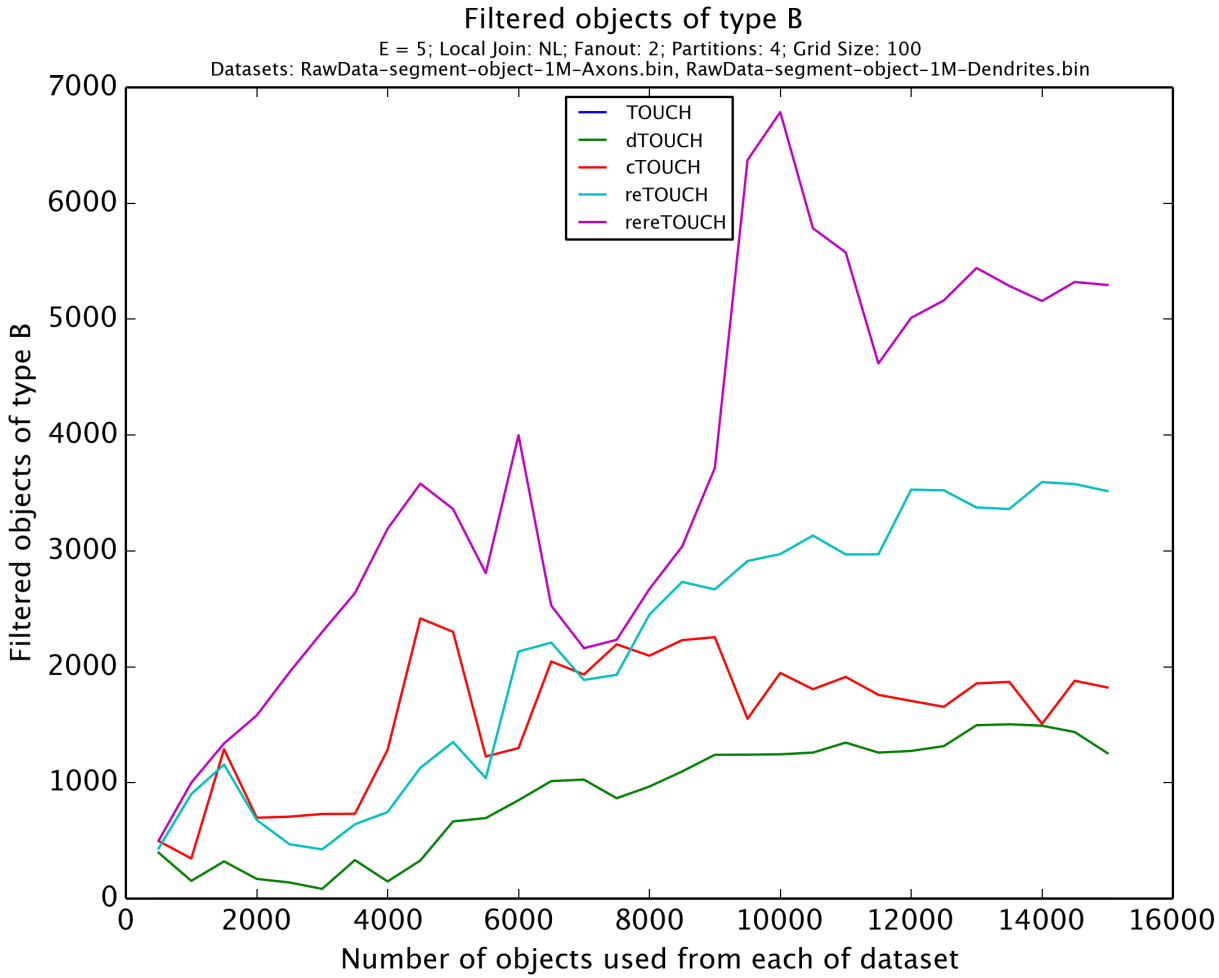
Third step: Joining step, checking two buckets of objects using their arbitrary spatial shape.
Using either Nested Loop or Spatial Grid Hash



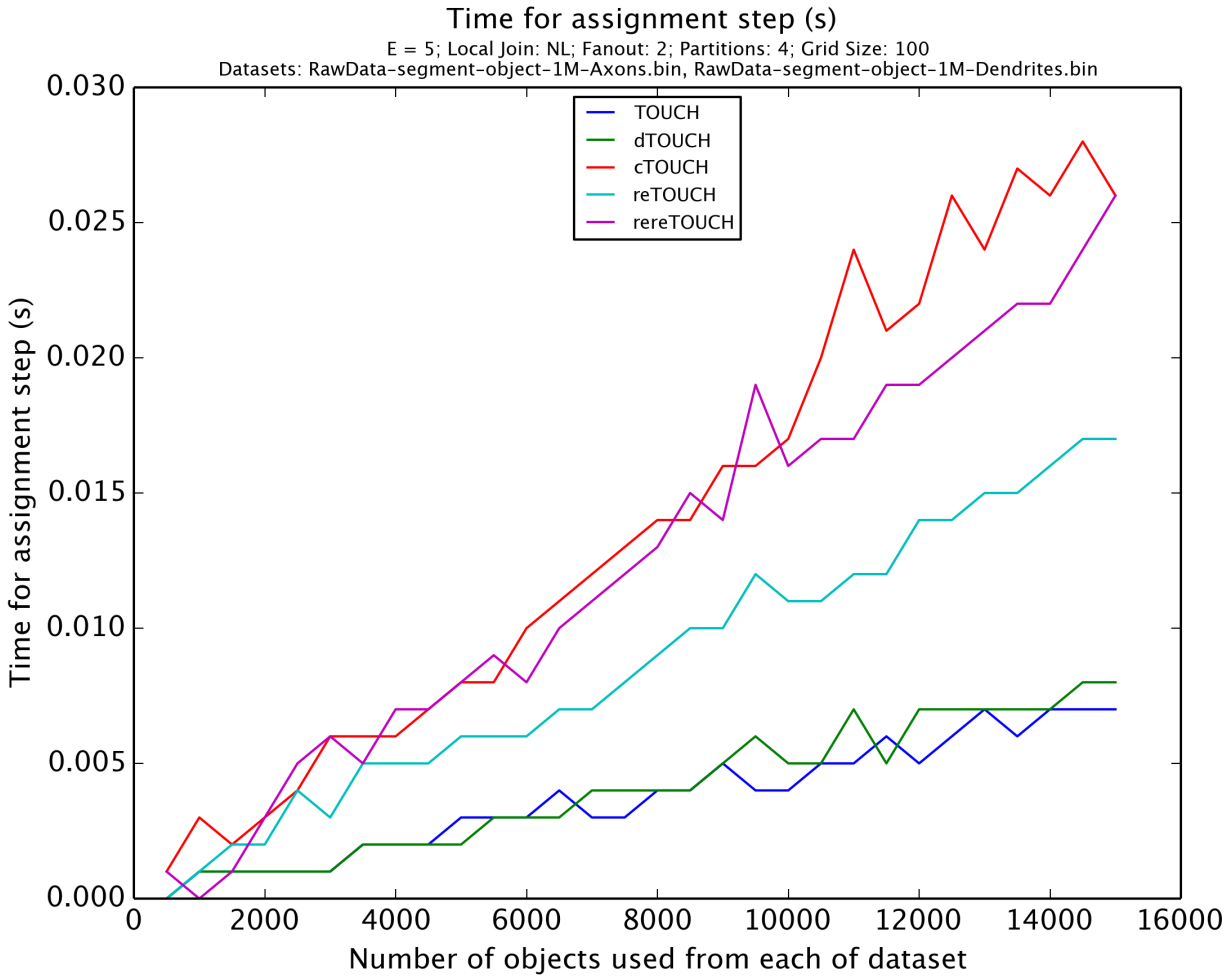
Computational experiment : Filtering A



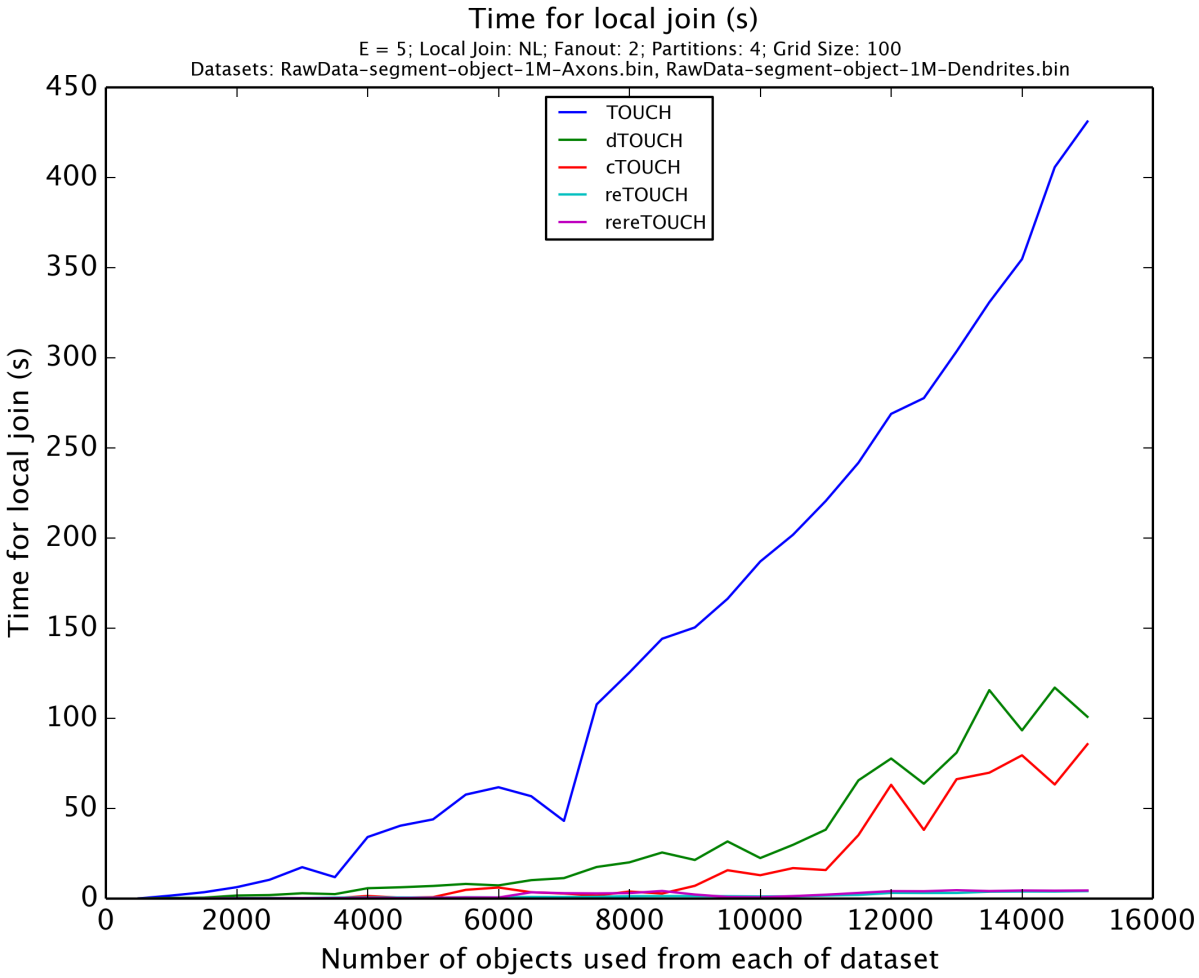
Computational experiment : Filtering B



Computational experiment : Assignment time



Computational experiment : Join time



Conclusion

All modifications considerably improve total spatial join time by increasing number of filtered objects and decreasing number of objects that should be checked for intersection.