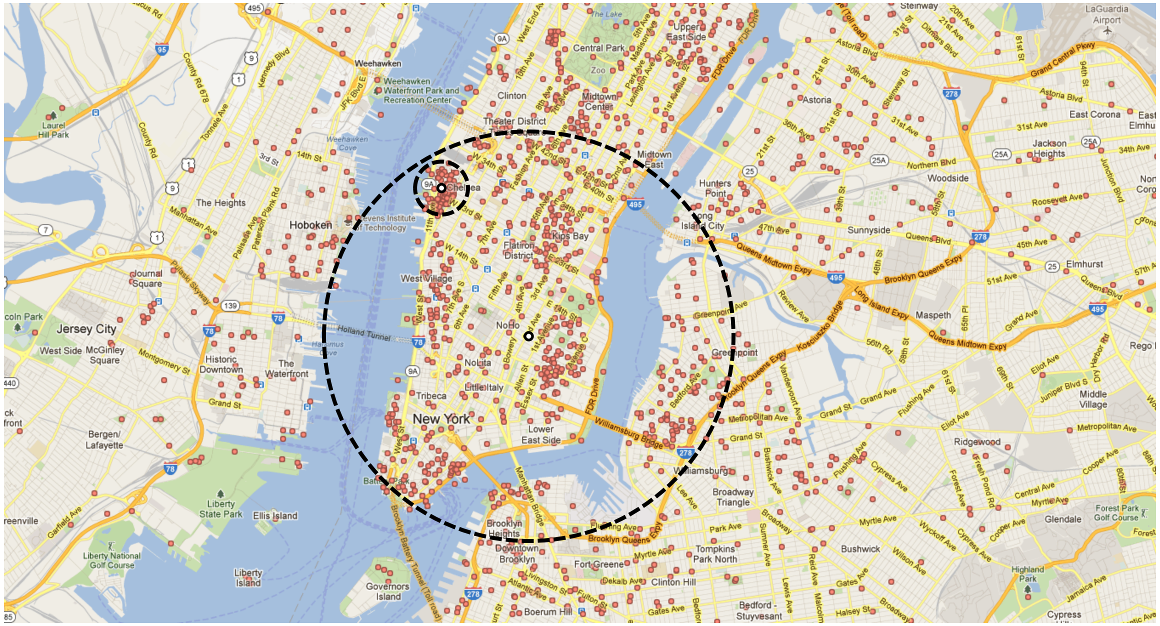
A Scalable Algorithm for Maximizing Range Sum in Spatial Databases

Dong-Wan Choi, KAIST, Korea
Chin-Wan Chung, KAIST, Korea
Yufei Tao, KAIST & CUHK, Hong Kong

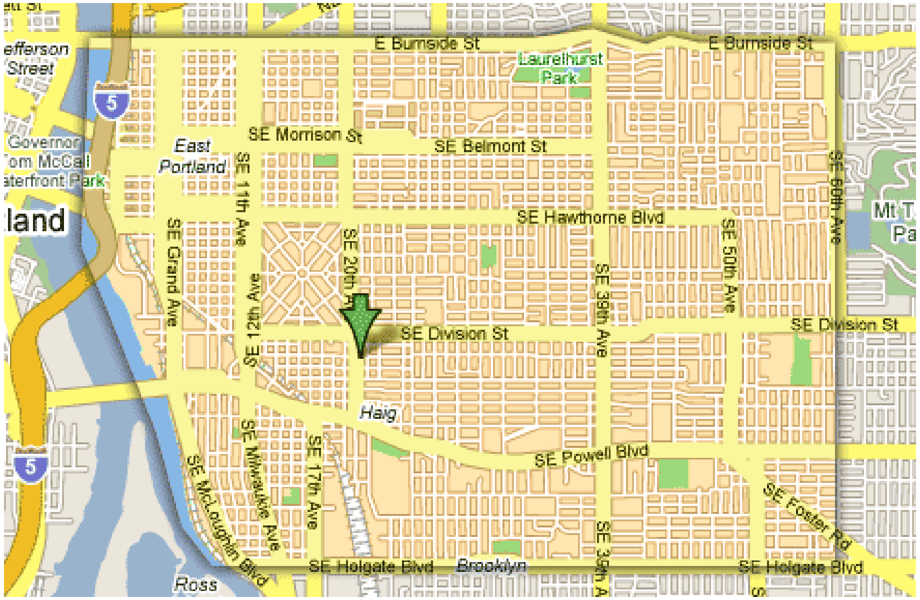
Motivation

- Where is the most representative spot in a city for a tourist?



  - Maximize the number of tourist attractions within a movable range

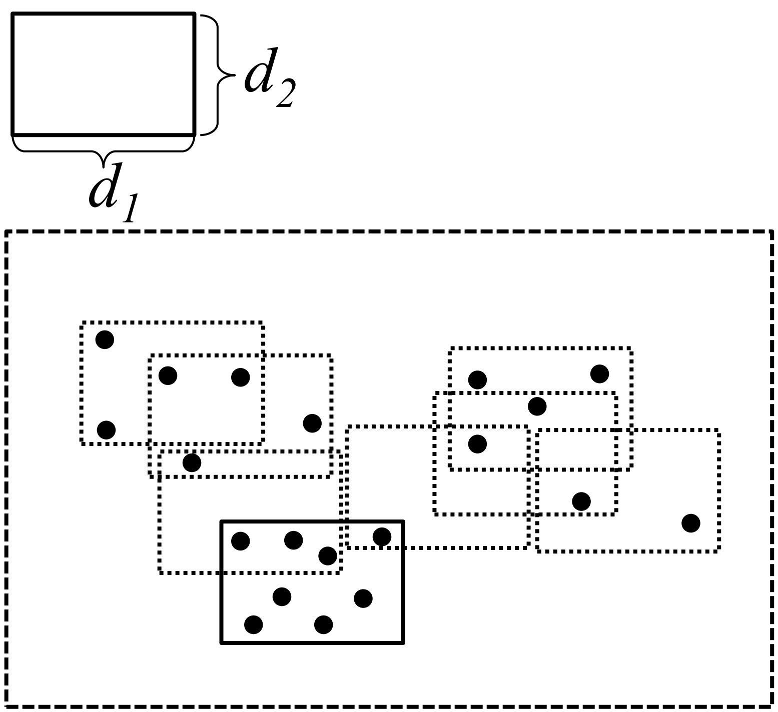
Motivation

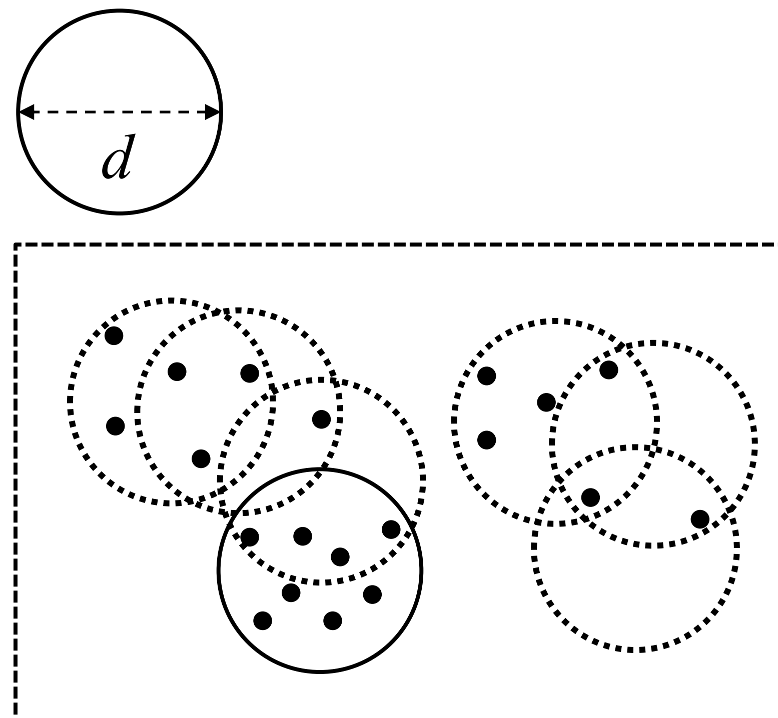


- Where is the most profitable place to set up a new pizza store?
  - Maximize the number of residents within a delivery range

MOTIVATION

- Where is the best location that maximizes the number of objects covered by a given range?





Formal Definition

- MaxRS (Maximizing Range Sum) problem
  - Given a set O of weighted objects and a rectangle r of a given size,
  - Find a location p of r that maximizes:

$$\sum_{o \in O_{r(p)}} w(o),$$ where

$r(p)$ is the rectangle centered at a location p,
$O_{r(p)}$ is the set of objects covered by $r(p)$, and
$w(o)$ is the weight of $o \in O$

- MaxCRS (Maximizing Circular Range Sum) problem
  - The circle version of the MaxRS problem

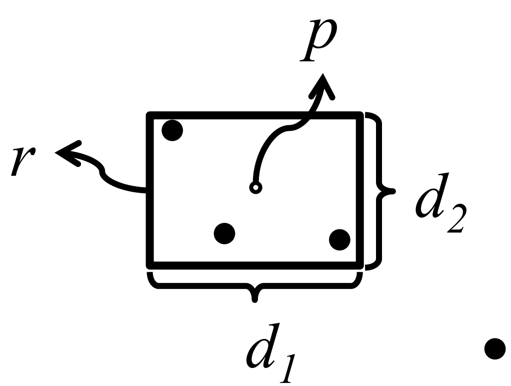
Contents

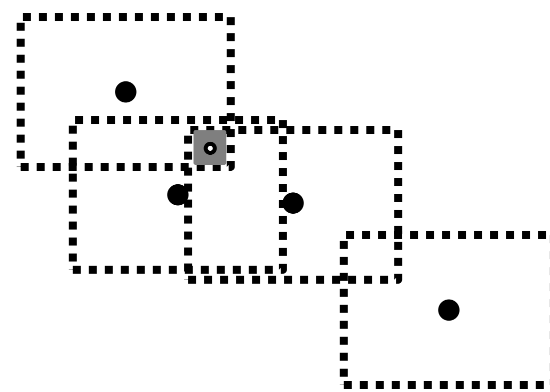
- ~~Motivation~~
- ~~Formal Definition~~
- Preliminary
- Our Algorithms
  - ExactMaxRS
  - ApproxMaxCRS
- Theoretical Results
- Experimental Results
- Conclusion and Future works

PRELIMINARY

- A naïve solution
  - Issuing range aggregate queries for every location
  - Problem: Infinite # of locations!

- Problem transformation



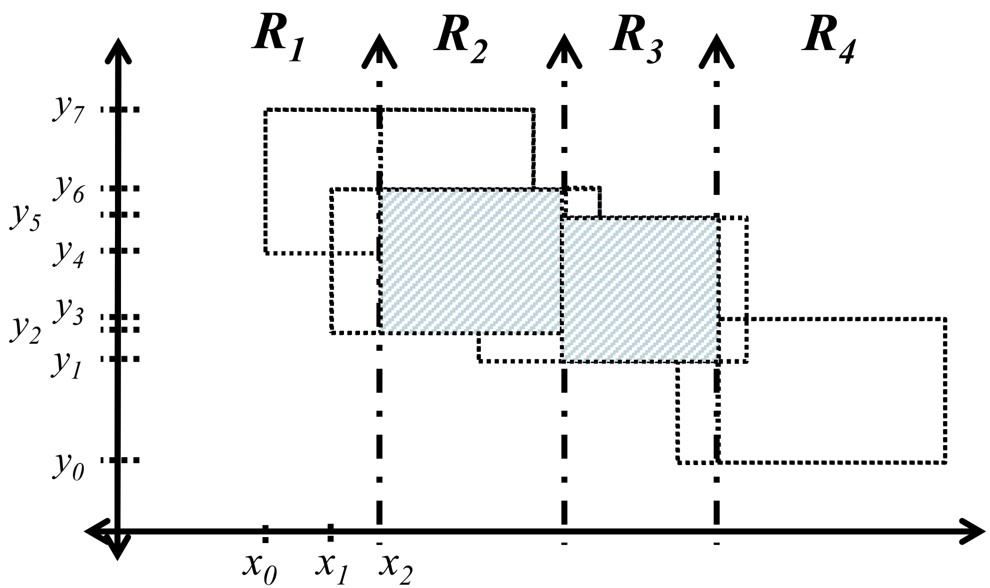


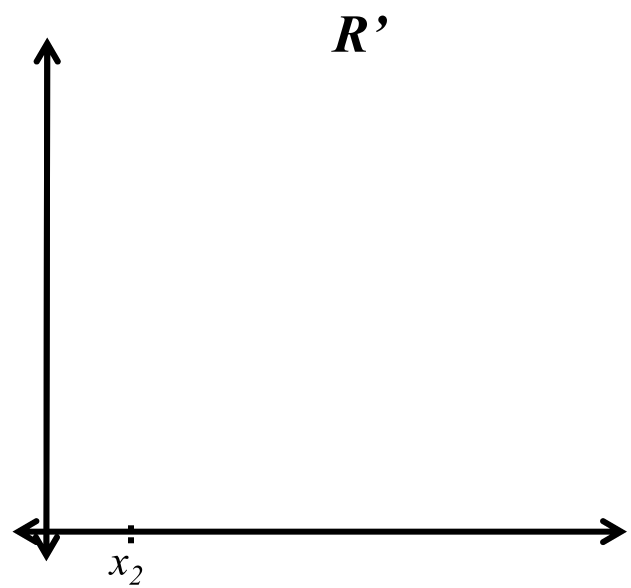
  - Given a set of rectangles, find the most dense region where the most rectangles intersect

ExactMaxRS

- Exact algorithm that solves MaxRS
- External-memory algorithm
  - Scalable for a massive dataset
- Follows the divide-and-conquer strategy:
  - Recursively divide the entire dataset into smaller subsets
  - Compute a local solution for each subset
  - Merge local solutions of subsets

Division Phase

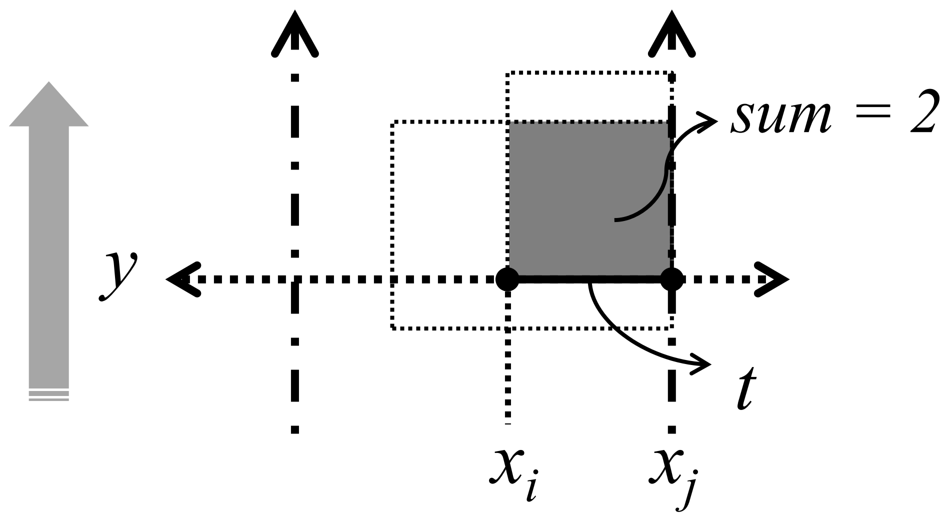




- Divide the space vertically into m sub-spaces, called *slab*s, each of which has roughly the same # of rectangles
  - Until the # of rectangles can fit in the main memory
- Do not pass spanning rectangles to the next level of recursion

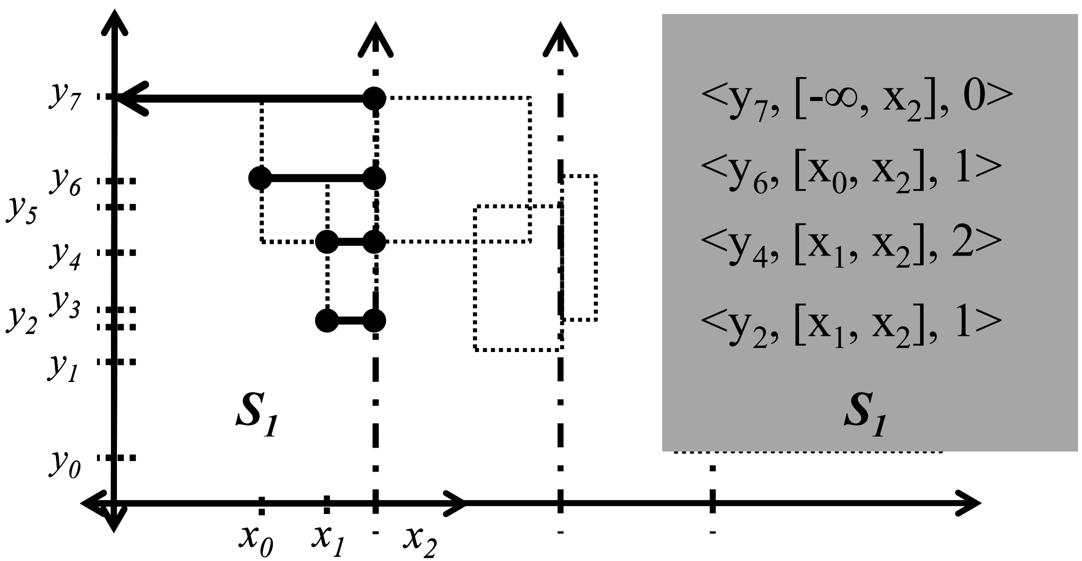
Slab-files

- The structure to be returned after conquering the sub-problem w.r.t. a slab
  - The set of tuples, each of which is $t = \langle y, [x_i, x_j], sum \rangle$
    - In the upward direction, after y, the most dense region (whose total weights is sum) is in the x-range $[x_i, x_j]$.

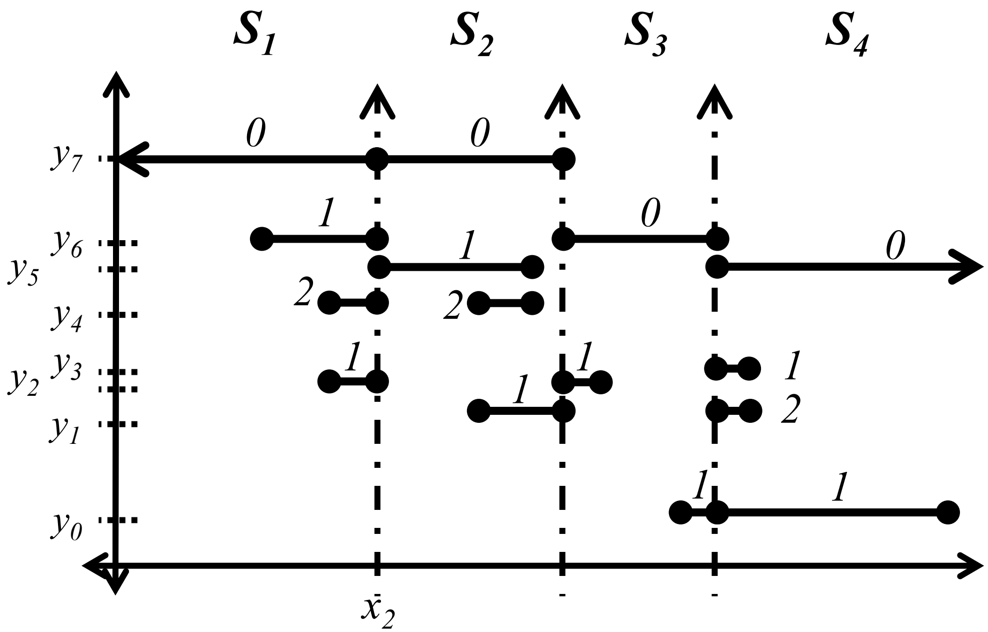


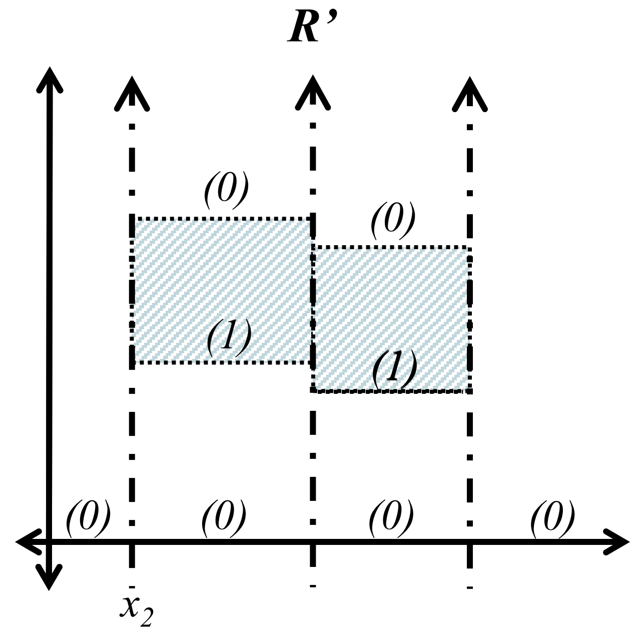
SLAB-FILES

- An example of a slab-file



Merging Phase





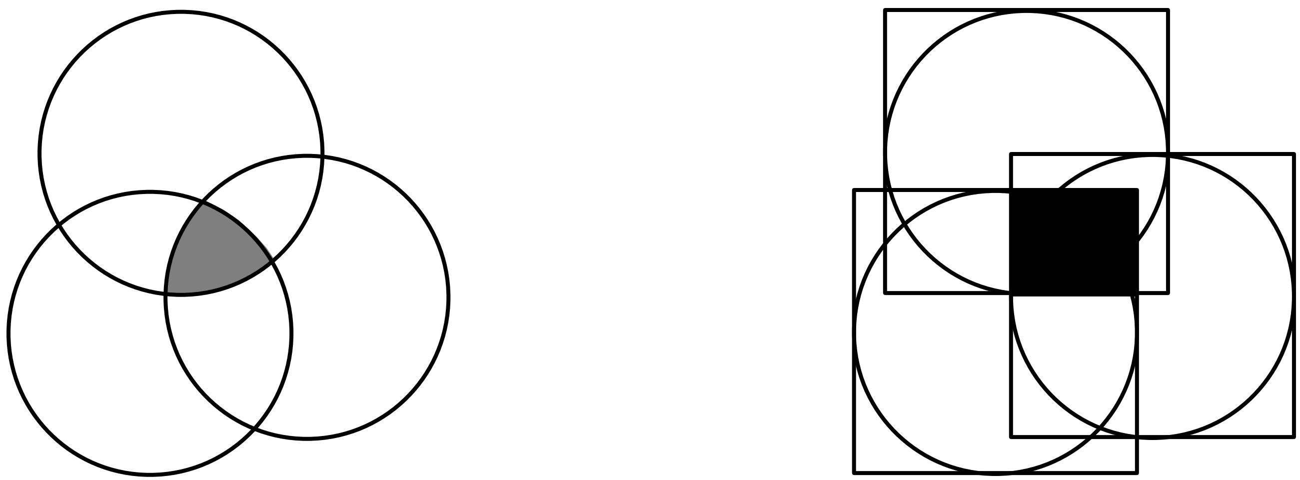
- Sweep a horizontal line across the slab-files (S_1, …, S_4) and the spanning rectangle file (R')
- When encountering several tuples at a horizontal line, choose a tuple with a maximum sum

ApproxMaxCRS

- Approximation algorithm for MaxCRS
- Uses the ExactMaxRS algorithm as a tool
- Overall Flow
  1. Transform MaxCRS into MaxRS with MBRs
  2. Do ExactMaxRS on the transformed dataset
  3. Generate candidate points based on the result from ExactMaxRS
  4. Choose the best point among the candidate points

Transformation

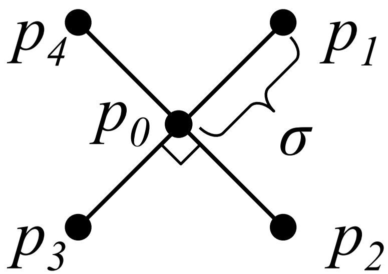
- MaxCRS → MaxRS with MBRs
  - Construct the MBR for each circle



- Find the most dense region of MaxRS with MBRs

Candidate Points

- Generate five candidate points based on the result from ExactMaxRS
  - p_0 : the center point of the most dense region returned from ExactMaxRS
  - p_1, p_2, p_3, p_4 : four shifted points from p_0



$$\sigma \in \left((\sqrt{2} - 1)\frac{d}{2}, \frac{d}{2} \right)$$

, where d is the diameter of circles

- Return the best point p_i among $p_0, \ldots, p_4$
  - such that the total weight of the circles covering p_i is maximized.

THEORETICAL RESULTS

- ExactMaxRS
  - Optimal in terms of the I/O complexity
    - $O((N/B)\log_{M/B}(N/B))$ I/O's, where N is the # of objects, M is the memory size, and B is the block size
      - The counterpart of O(nlogn) in the main memory

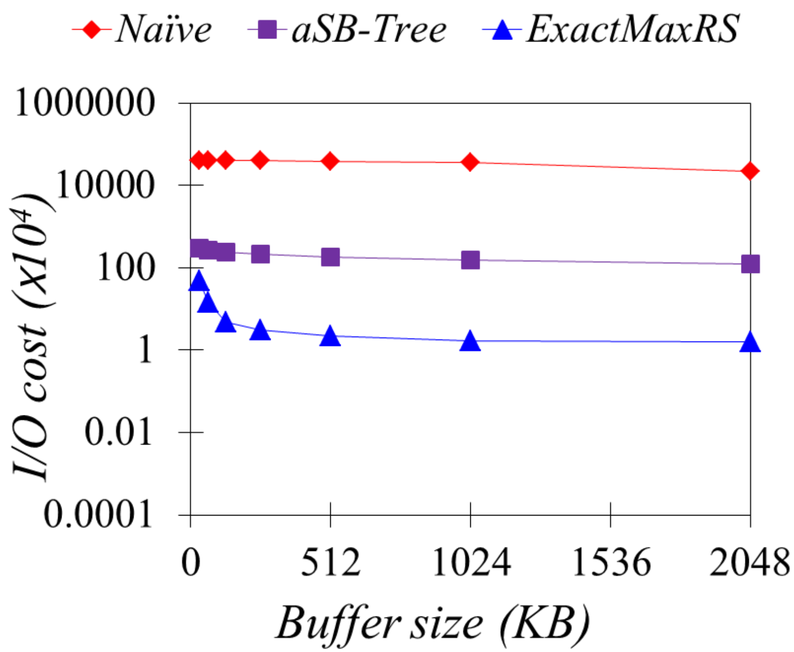
        → optimal time complexity
- ApproxMaxCRS
  - (1/4)-approximation bound
    - Much better in practice

Experimental Results

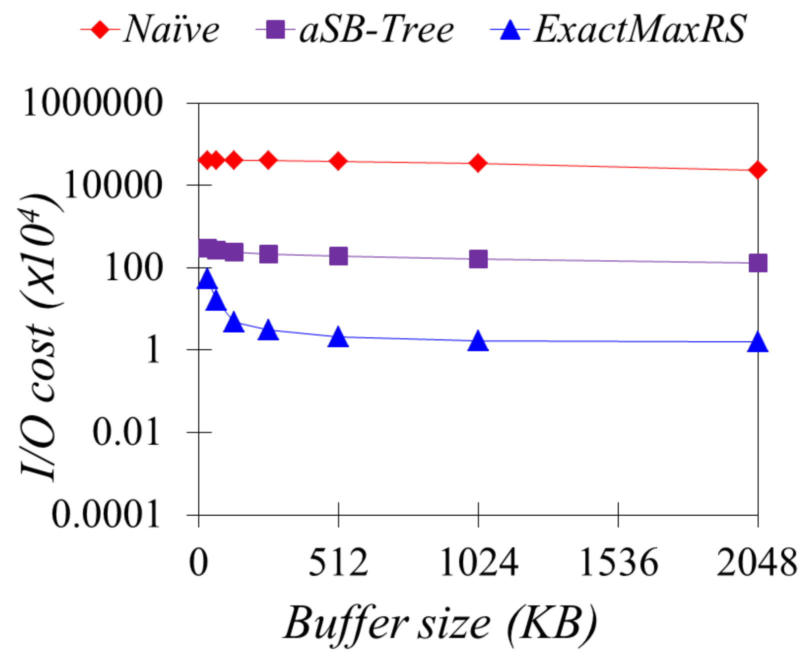
- Environment Setting
  - Synthetic datasets
    - Uniform distribution and Gaussian distribution
    - Cardinality: 100,000~500,000
  - Real datasets
    - NE and UX datasets from the R-tree Portal site
    - Cardinality: 123,593 for NE, 19,499 for UX
  - Compared approaches
    - Naïve plane sweep, $O(n^2)$
    - Plane sweep using aSB-tree, $O(n\log n)$
      - Adaptation of the optimal in-memory algorithm for the MaxRS problem

Experimental Results

- I/O cost with varying the buffer size



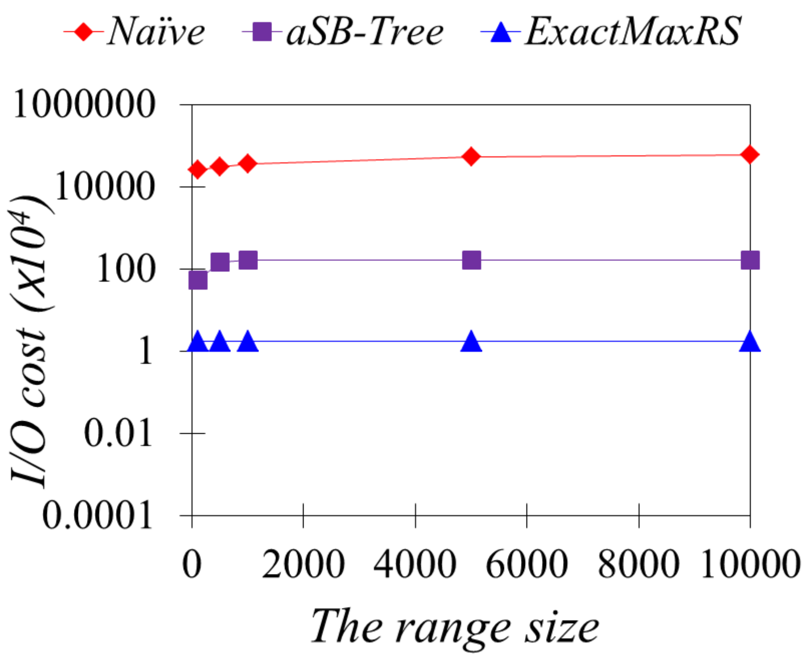
<Gaussian distribution>



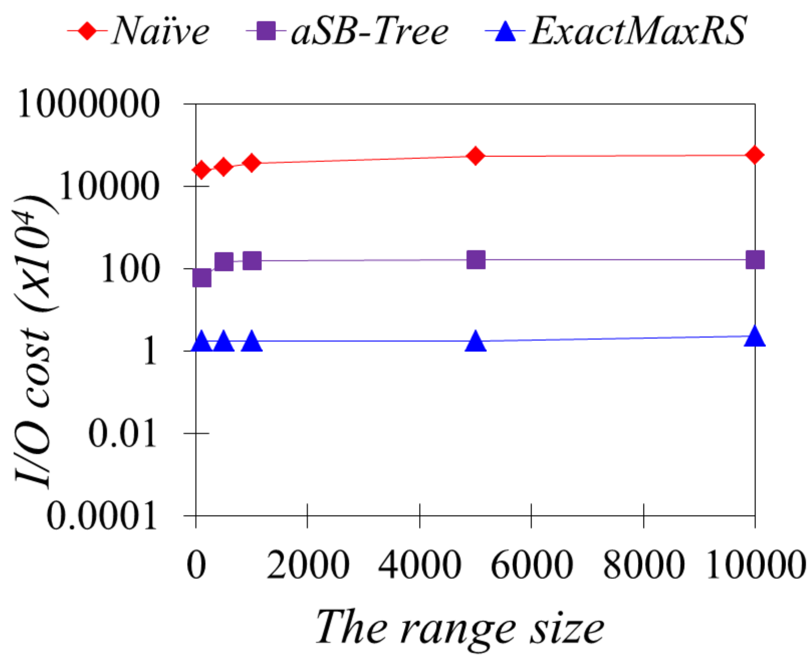
<Uniform distribution>

Experimental Results

- I/O cost with varying the range size



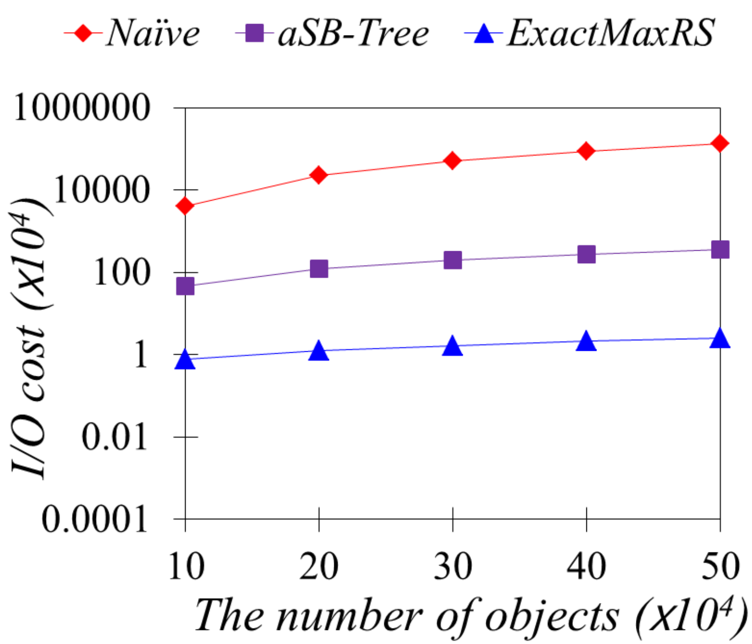
<Gaussian distribution>



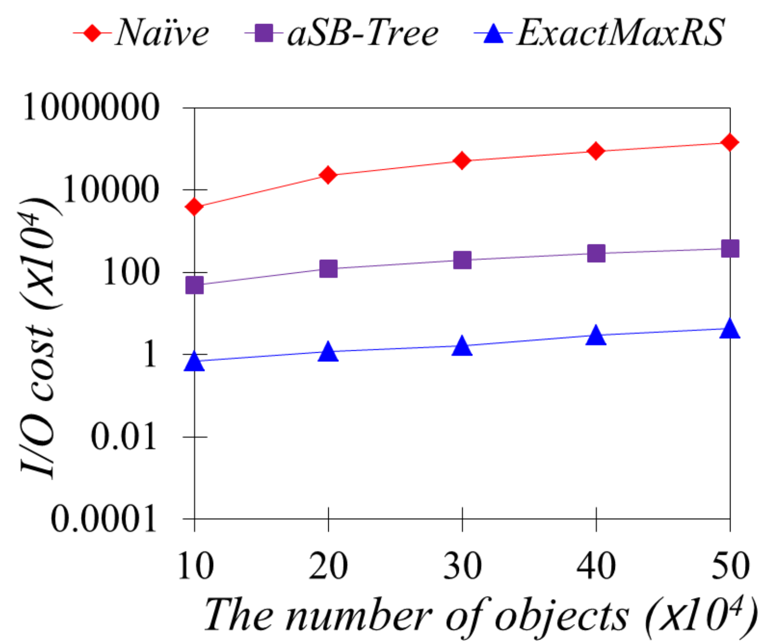
<Uniform distribution>

Experimental Results

- I/O cost with varying the cardinality



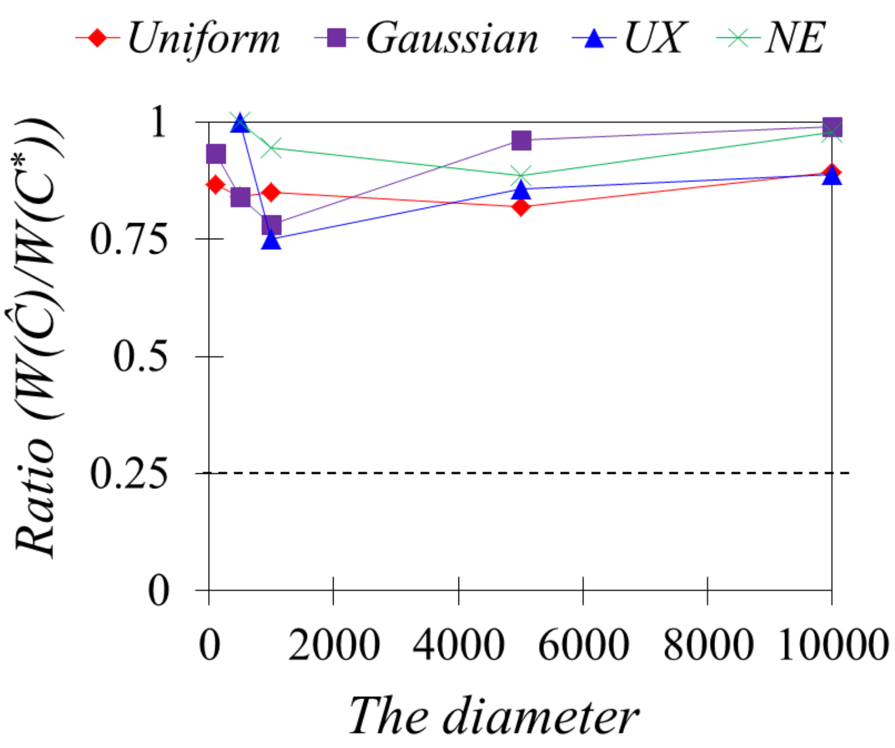
<Gaussian distribution>



<Uniform distribution>

Experimental Results

- Error ratio of ApproxMaxCRS with varying the diameter



Conclusion & Future works

- Contributions
  - Proposed the first optimal external-memory algorithm for the MaxRS problem
  - Proposed the (1/4)-approximation algorithm for the MaxCRS problem
  - Proved theoretically the correctness, optimality, approximation bound, and tightness of the bound
  - Evaluated experimentally the efficiency and accuracy
- Future works
  - Max*k*RS problem
  - MinRS problem

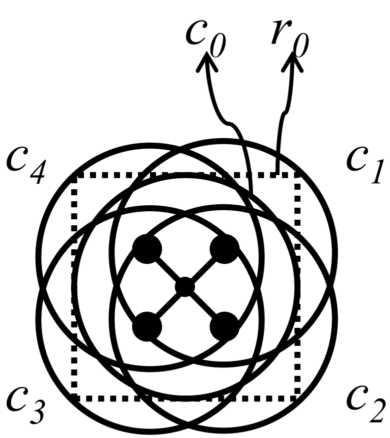
Thank You

APPENDIX

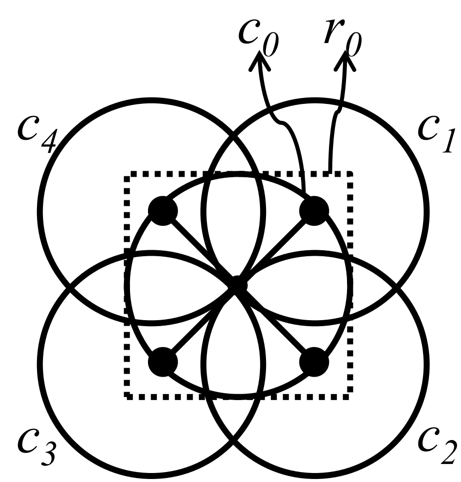
- The proof sketch of the correctness of ExactMaxRS
  - The x-range of a tuple in a slab-file is called "max-interval".
  - Let I^* be the max-interval w.r.t. entire space and $I^\wedge$ be a piece of I for a recursion level. Then $I^\wedge$ is also the max-interval w.r.t. the sub-space corresponding to the recursion level.
    - There cannot be an interval I' whose sum is larger than $I^\wedge$, since spanning rectangles can affect all or none of intervals in a slab.

APPENDIX

- The proof sketch of (1/4)-approximation bound
  - The four circles together cover the region for MaxRS, which covers *k* points. Hence, at least one of those circles covers *k/4* points.



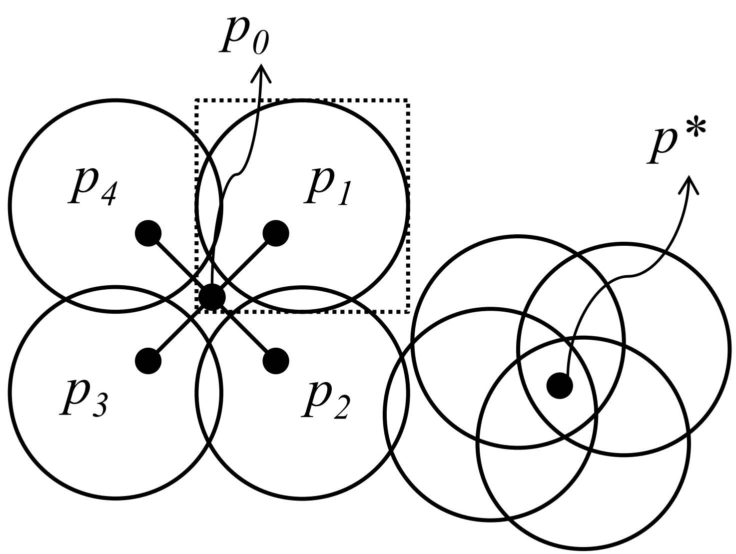
$$\sigma = \frac{d}{2}$$



$$\sigma = (\sqrt{2} - 1)\frac{d}{2}$$

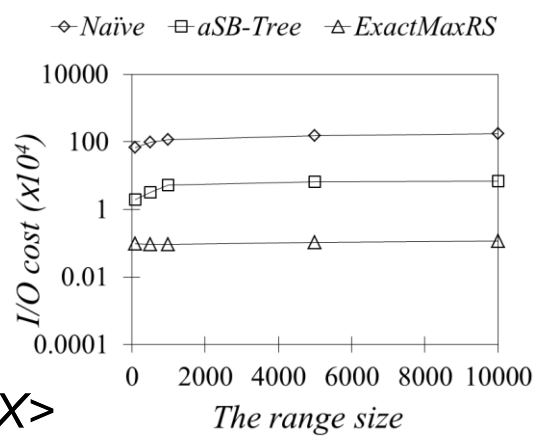
Appendix

- The proof sketch of the tightness of the approximation bound
  - $4p_i \leq p^*$

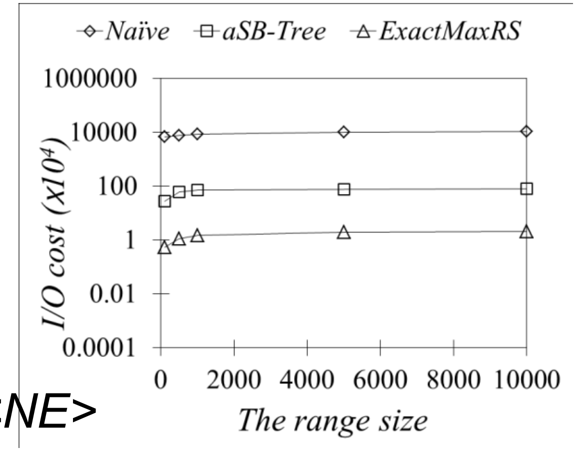


APPENDIX

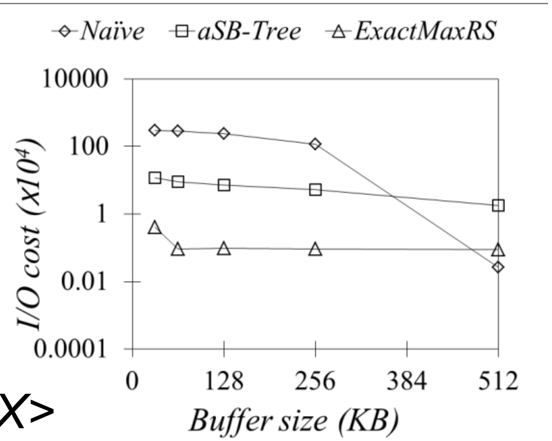
- Results on real datasets



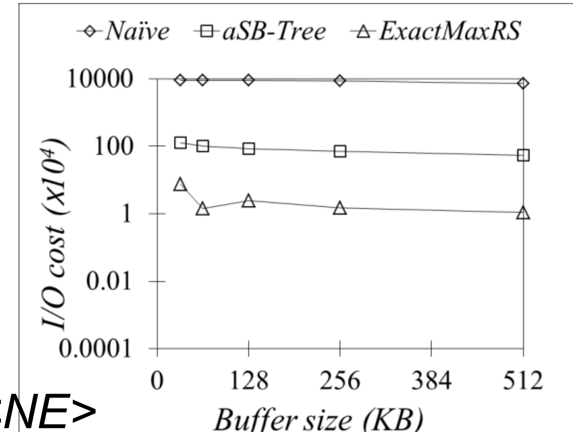
<UX>



<NE>



<UX>



<NE>