American Journal of Applied Sciences 8 (12): 1404-1412, 2011
ISSN 1546-9239

# Effective Processing of Constraints based Spatial Join using R-Trees

[1]R. Parvathi and [2]S. Palaniammal
[1]School of Computer Science and Engineering,
VIT University, Chennai Campus,
Chennai, 600 048, Tamil Nadu, India
[2]Department of Science and Humanities,
VLB Janakiammal College of Engg and Tech,
Coimbatore, 641 042, Tamil Nadu, India

**Abstract: Problem statement:** This study focuses on the spatial join effects with the constraints-based spatial data without any extra cost and Finding the minimum execution time of the spatial query and spatial selection method. **Approach:** Spatial joins are used to combine the spatial objects. The efficient processing depends upon the spatial queries. The execution time and I/O time of spatial queries are crucial, because the spatial objects are very large and have several relations. In this article, we use several techniques to improve the efficiency of the spatial join. (1) We use R*-trees for spatial queries since R*-trees are very suitable for supporting spatial queries as it is one of the efficient member of R-tree family. (2) The different shapes namely point, line, polygon and rectangle are used for isolating and clustering the spatial onjects. (3) We use scales with the shapes for spatial distribution. We also present several techniques for improving its execution time with respect to the CPU and I/O-time. In the proposed constraints based spatial join algorithm, total execution time is improved compared with the existing approach in order of magnitude. Using a buffer of reasonable size, the I/O time is optimal. The performance of the various approaches is investigated with the synthesized and real data set and the experimental results are compared with the large data sets from real applications. **Results:** The R*-tree concept reduce the number of search pages to combine spatial objects. By using this, CPU utilization time increases, the number of comparisons of spatial objects can be reduced and also reduces the I/O time. **Conclusion/Recommendations:** The performance of the various approaches is investigated with the synthesized and real data set and the experimental results are compared with the large data sets from real applications.

**Key words:** Spatial data mining, spatial clustering, spatial queries, spatial join, Minimum Bounding Rectangles (MBR), Akaike Information Criterion (AIC), real data, constraints based

## INTRODUCTION

Spatial join is one important spatial query in a spatial database system which combines two spatial datasets to retrieve the matched pair of objects based on the spatial predicate. The spatial predicate specifies the geometrical relationship between objects, and the most common one is the overlap of spatial objects. In a GIS system, the spatial join may be used for an efficient implementation of the map overlay. The map overlay constructs a new map from two or more given maps which is important for geographic analysis. Application areas contain city planning, ecological, demographic studies, and so on (Arge *et al*., 2000).

Spatial join operation is used to cluster two or more dataset with respect to a spatial predicate. Predicate can be a combination of direction, distance and topological relations of spatial objects. In non spatial join, the joining attributes must be of the same type and in spatial join they can be of different types. Each spatial attribute is represented by its Minimum Bounding Rectangles (MBR).

A typical example of spatial join is Find all pair of rivers and cities that intersect". For example in Fig. 1, the result of join between the set of rivers {R1, R2} and cities {C1, C2, C3, C4, C5} is { (R1, C1), (R2,C5)}( Shekar and Chawla, 2003).

**Corresponding Author:** R. Parvathi School of Computer Science and Engineering, VIT University, Chennai Campus, Chennai, 600 048, Tamil Nadu, India

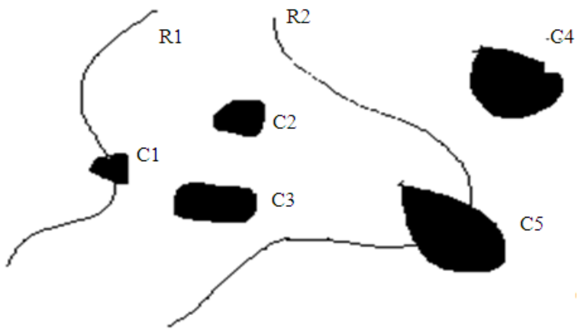


Fig. 1: Example of spatial joins (Shekar and Chawla, 2003)

The spatial objects can be movable and immovable objects. Hash based algorithms focus only on natural join and equijoin. Since spatial objects are multidimensional data, we need new efficient spatial join algorithm. The methods for computing the spatial join are discussed in great detail for quad trees and similar access methods (Shou *et al*., 2003).

**Key concepts:** The two steps involved in spatial join are Filter step and Refine step. In filtering step, tuples whose Minimum Bounding Rectangle (MBR) overlaps with query region are determined. This step is not computationally expensive but it requires at most four computations to determine rectangles intersection. In Refine step, the tuples which passed the filter step is fed to the refinement step where exact spatial representation is used and spatial predicate is checked on these spatial representations. Refinement step is computationally expensive, but the number of tuples it processed in this step is less, due to initial filter step. Spatial join algorithm can be classified into three categories. Nested Loop, Tree Matching and Partition-based spatial merge join.

In General the spatial objects are of two types, namely, movable objects and immovable objects. The process that includes movable objects are like to identify the cell phone towers of the respective cellular network service provider and in process of immovable object are to find identify the ideal object in the multi spectral image (spatial data). For the movable objects the spatial queries are classified into Single scan queries and Multi-scan queries.

In this study we propose the spatial joins for immovable objects like to find the banks that are nearer to the PSG College of Arts and Science. Initially, with the help of the constraints based spatial clustering algorithm the banks are identified within Coimbatore. Here we use R*-trees for spatial database querying.

**Clustering introduction:** Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subset (clusters), so that the data in each subset (ideally) shares some common trait-often proximity according to some defined distance measure. Data clustering is a common technique for statistical data analysis which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics. The computational task of classifying the data set into k clusters is often referred to as k-clustering.

**Types of clustering:** Data clustering algorithms can be Hierarchical. Hierarchical algorithms find successive clusters using previously established clusters. Hierarchical algorithms can be agglomerative (“bottom-up”) or divisive (“top-down”). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

**Partitioned algorithms** typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.

**Density-based clustering algorithms** are devised to discover arbitrary-shaped clusters. In this approach, a cluster is regarded as a region in which the density of data objects exceeds a threshold. DBSCAN (Ibrahim and Salman, 2011) and OPTICS are two typical algorithms of this kind.

**Two-way clustering, co-clustering or biclustering** are clustering methods where not only the objects are clustered but also the features of the objects, i.e., if the data is represented in a data matrix, the rows and columns are clustered simultaneously.

Another important distinction is whether the clustering uses symmetric or asymmetric distances. A property of Euclidean space is that distances are symmetric (the distance from object A to B is the same as the distance from B to A).

Spatial database system is a database system for managing spatial data. There is a rapid growth in number and the size of spatial databases for applications such as geo-marketing, traffic control and environmental studies . Spatial data mining or knowledge discovery in spatial databases refers to the extraction from spatial databases of implicit knowledge, spatial relations, or other patterns that are not explicitly stored (Shekar and Chawla, 2003).

Clustering analysis for data in a 2-D space is considered as spatial data mining and its applications are geographic information systems, pattern recognition, medical imaging, marketing analysis, weather forecasting. Clustering in spatial data is an active research area and most of the research focus on effectiveness and scalability.

## MATERIALS AND METHODS

**Historical background:** (EDWIN 2007) grid based technique to perform spatial join. It is the first known technique to solve spatial join operation. Using a multidimensional grid, spaces are divided into smaller blocks, known as pixels. Then, z-ordering is used to order the pixels. Each object is approximated by the pixels which intersect with its MBR. As pixels are ordered by z-ordering, each object is represented by a set of z-values which are one-dimensional. Now, any one-dimensional indexing (e.g., B+-tree) can be used to sort them and by using sort-merge, spatial join operation is done. The performance of this technique solely depends on the granularity of the grids. The finer grids gives the accurate results but with higher memory consumption. Later on to remedy this problem, multidimensional indices (e.g., R-tree) (Mouratidis *et al*., 2008), which can directly handle spatial data, were devised. Various new spatial join algorithms (e.g., R-tree join, sort and match, spatial hash join, slot index hash join) based on multi-dimensional index appeared (EDWIN 2007)).

**Base work:** In the base study, data file is in (xls) format and used for the spatial join1 algorithm. Data is read into buffers with different Last Recently Used (LRU) size in bytes 0, 8, 32, 128, 256 and512 and compared with data in the size 1KB, 2KB, 4KB and 8KB (while inserting in the buffer) to find the optimal comparisions time.

In spatial join2 algorithm, before taking the buffer for the comparisions, apply the sorting technique to arrange the data (line shape) in order. For example each areas are identified with the unique identified code namely FID and it is sorted in order using nearby locations with detailed information like name of the place. Then use the spatial join1 algorithm for further process.

In spatial join3 algorithm, based on spatial ordering this algorithm creates a sequence of pairs of intersectional rectangles. Obviously, this sequence can also be used to determine the read schedule of the spatial join.

**Partition clustering:**

**K-means clustering:** The K-means algorithm (Velmurugan and Santhanam, 2010) assigns each point to the cluster whose center (also called centroid) is the nearest. The center is the average of all the points in the cluster that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. For example, a data set has three dimensions and the cluster has two points: $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$. Then, the centroid Z becomes $Z = (z_1, z_2, z_3)$, where $z_1 = (x_1 + y_1)/2$ and $z_2 = (x_2 + y_2)/2$ and $z_3 = (x_3 + y_3)/2$.

**The algorithm steps are:**

- Choose the number of clusters, k
- Randomly generate k clusters and determine the cluster centers, or directly generate k random points as cluster centers
- Assign each point to the nearest cluster center
- Recompute the new cluster centers
- Repeat the two previous steps until some convergence criterion is met (usually that the assignment hasn't changed)

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantages are not yielding the same result with each execution, since the resulting clusters depend on the initial random assignments. It minimizes intra-cluster variance, but does not ensure that the result with global minimum of variance.

**QT clustering algorithm:** Quality Threshold (QT) clustering is an alternative method of partitioning data, invented for gene clustering. It requires more computing power than k-means, but need not to specify the number of clusters in prior and also it returns the same result when several executions takes place.

The algorithm is:

- The user chooses a maximum diameter for clusters
- Build a candidate cluster for each point by including the closest point, the next closest and so on, until the diameter of the cluster surpasses the threshold
- Save the candidate cluster with the most points as the first true cluster and remove all points in the cluster from further consideration. Must clarify what happens if more than 1 cluster has the maximum number of points
- Recurse with the reduced set of points

The distance between a point and a group of points is computed using complete linkage, i.e., as the maximum distance from the point to any member of the group.

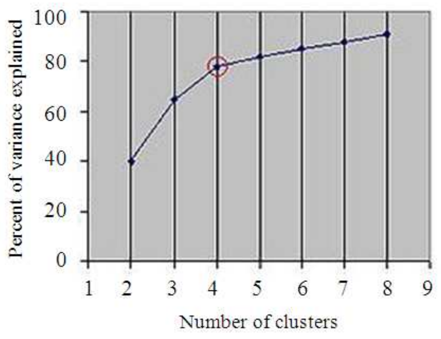


Fig. 2: Percent of variance with the number of clusters

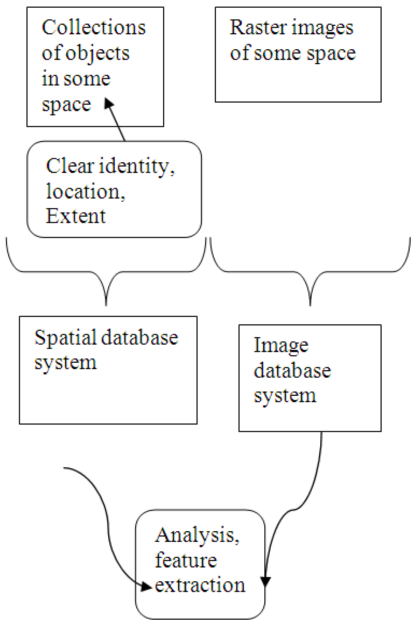

Collections of objects in some space
Raster images of some space
Clear identity, location, Extent
Spatial database system
Image database system
Analysis, feature extraction

Fig. 3: An example of a spatial database

**Locality-sensitive hashing:** Locality-sensitive hashing can be used for clustering. Feature space vectors are sets and the metric used is the Jaccard distance. The feature space can be considered high-dimensional. The min-wise independent permutations LSH scheme (sometimes Min Hash) is then used to put similar items into buckets. With just one set of hashing methods, there are only clusters of very similar elements. By seeding the hash functions several times (e.g. 20), it is possible to get bigger clusters.

**Graph-theoretic methods:** Formal concept analysis is a technique for generating clusters of objects and attributes, giving a bipartite graph representation for relations between the objects and attributes. Other methods for generating overlapping clusters are also discussed.

**Determining the number of clusters:** If the number of the clusters is not apparent from prior knowledge, it should be chosen in some way. Several methods for this have been suggested within the statistical literature where one rule of thumb sets the number to Eq. 1:

$$k \approx (n / 2)^{1/2} \qquad (1)$$

where, n is the number of objects (data points).

In the Fig. 2, the “elbow” is indicated by the red circle. The number of clusters chosen should be 4. Another thumb rule looks at the percentage of variance explained as a function of the number of clusters: We should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if we graph the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters chosen at this point is “elbow criterion”. This “elbow” cannot always be unambiguously identified. Percentage of variance explained is the ratio of the between-group variance to the total variance. A slight variation of this method plots the curvature within group variance. Other ways to determine the number of clusters use Akaike Information Criterion (AIC) or Baysian Information Criterion BIC), if it is possible to make a likelihood function for the clustering model. For example the k-means model is almost a Gaussian mixture model and thus also determine AIC and BIC values.

In a GIS application, studying the movement of pedestrians to identify optimal bank machine placements, for example, the presence of a highway hinders the movement of pedestrians and should be considered as an obstacle. To the best of our knowledge the following Clustering algorithms for Clustering Spatial data with the constraints are DBRS, DBRS+, COD_CLARNS, AUTOCLUST, AUTOCLUST+, DBCluc, IKSCOC, GKSCOC and PSOCOC.

**Spatial database:** Spatial database is a system related to some space represented in Fig. 3 (Shekar and Chawla, 2003). It should contain the collection of objects related to space of different types.

**The different objects of spatial database:**

- Point: represents a single location like city or particular place
- Line: Moving through space, connections in space (River, Highways)

- Region: is an abstraction of an object with extent (forest, Lake, City)

Fundamental operations on spatial data (Algebra):

- Spatial selection
- Spatial join
- Spatial function application
- Other set operations

**Spatial selection:** Selection based on a spatial predicate:

- "Find all cities in Bavaria"
- Cities select (center inside Bavaria)
- "Find all rivers intersecting a query window"
- Rivers select (route intersects Window)
- "Find all big cities no more than 100 km from Hagen"
- Cities select (dist(center,Hagen) 100 pop>500 000)

**Spatial join:** Join based on a predicate comparing spatial attribute values (Cheng *et al*., 2006):

- "Combine cities with their states"
- Cities states join(center inside area)
- "For each river, find all cities within less than 50 km"
- Cities rivers join(dist(center, route) <50)
- A REGION value
- A REGION value
- A POINT value

**Spatial function application:** How can we use operations of a spatial algebra computing new SDT values?

**Example:** Regions lines and lines intersection:

- In selection conditions.
- Object algebra operators allow one to apply functions to each member of a set:
- Filter operator (FAD)
- Replace
- Map
- Extend
- "For each river going through Bavaria, return the name, the part inside Bavaria and the length of that part."
- Rivers select(route intersects Bavaria)
- Extend(intersection(route, Bavaria) {part})
- Extend(length(part) {plength})
- Project(rname, part, length)

**Spatial query processing:** Since spatial database contains different objects, we need excellent access methods to handle the data.

In traditional cases MBR (Minimum Bounding Rectangle) concept was used.

In this case, two techniques are used.

- Filter step: Find all objects whose MBR intersects the query rectangle
- Refinement step: For those objects, check whether they really fulfill the query condition (if necessary, make use of the exact representation)

The basic idea of these methods is to decompose the data space into non-overlapping cells which can be computed by recursively cutting the space into two or four parts of equal size. Cells can be identified by a location code, called z-value and a size code, called level. In R*-trees also we should not give the redundant values (Bae *et al*., 2010).

**R*-Tree:** An R-tree (Francis and Thambidurai, 2007) is a B*-tree like access method that stores multidimensional rectangles as complete objects without clipping them or transforming them to higher dimensional points. The structure of R-Tree is given the Fig. 4. Until now the most efficient variant of the R-trees is experimentally shown to be the R*-tree. The R*-tree uses more sophisticated insertion and splitting algorithms than the original R-tree. However, there is almost no difference in the data structure .

**Advantage:** Spatial object (or key) in a single bucket.

**Disadvantage:** Multiple search paths due to overlapping regions.

**Spatial joins using R*-tree:** The CPU time is calculated by using the floating point comparisons.

**Algorithm:** (An *et al*., 2001)

```
SpatialJoinl (R,S: R_Node); (* height of R is equal height of S *)
FOR (all Es € S) DO
FOR (all $E_R$ € R with $E_R$.rect o Es.rect # Ǿ) DO
IF (R is a leaf page) THEN
 (* (S is also a leaf page)*)
 Output $E_R$,$E_S$
ELSE
 ReadPage($E_R$.ref); ReadPage(Es.ref);
```

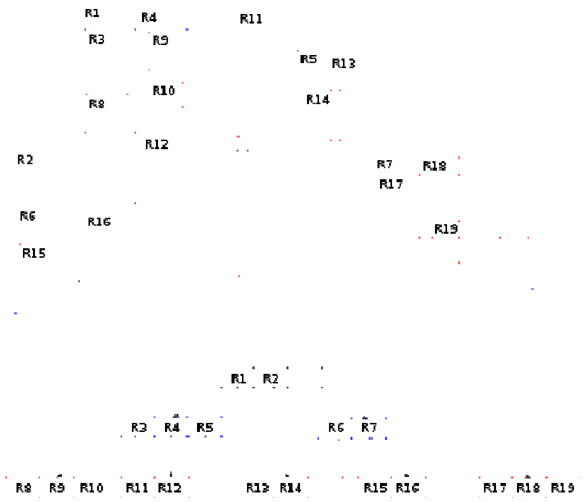


Fig. 4: Structure of R-tree

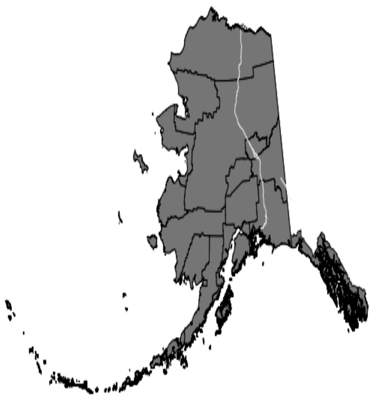


Fig. 5: Alaska region map with pipeline

```
SpatialJoinl $E_R$.ref, $E_S$.ref
END
END
END
END SpatialJoin 1;
```

Here B and C are joined with D.

We have to consider each and every point i.e. A with B, C, D and B with A, B and C and so on. This will take the comparisons more ($n^2$). So this is not effective for more spatial joins.

Sorting the objects belonging to their category can be done in prior to spatial Joins for time optimization of CPU process.

**The proposed spatial join algorithm:**

Step 1: Sort the data objects
Step 2: Compare the nodes for the relationship

The objects may be in single relation or multiple relations. For example combine the highways in Coimbatore city. This is simple relation. Combining the highways that are connecting the universities is multiple relations. Likewise the complexity may get increased.

**Constraints based spatial join:** The proposed Spatial Joins with the constraints manipulate whole sets of spatial objects in a special way. The spatial join operation is a conceptual unit which aims to address the constraints based spatial objects (Jacox and Samet, 2007; Papadias, 2001).

**K-medoids clustering under spatial data with obstacle constraints:** Because in spatial data there are two kinds of objects namely movable and ideal objects. To identify the moving and non-moving objects here we use K-medoids clustering with the obstacle using edge detection method. This helps to cluster the data with the obstacle constraints to identify movable and ideal objects under spatial data (images). It is easy and less time consumption (An *et al*., 2001). K-Medoids Clustering generates the clusters with much related data (Velmurugan and Santhanam, 2010).

**Spatial joins for datasets:** After K-Medoids it is easy to join our spatial data. By using the modified Spatial Join Algorithm we can combine our datasets.

**Spatial joins for images:** If the images are spatial images then it is possible to cluster by using image clustering algorithms. After that we can use the modified spatial join algorithm. Hence to identify the movable objects we can using K-NN (K-nearest neighnour algorithm) (Bae *et al*., 2009; 2010; Gao *et al*., 2011) and for immovable object Spatial Joins Algorithm. The above process is most essential for applying spatial joins under the spatial data (images). As an example, alaska region and pipeline maps were taken for experiment (Fig. 5). The spatial join operation shown in the Fig. 6. The resultant map shows the districts which are connected in the pipeline and list of districts are shown in the Fig. 7.

**Spatial joins with points:** With the help of clustering algorithm in data mining it is possible to group the data under spatial objects (images). Namely we can use K-mediods constraint based clustering algorithms under the spatial data to find the cluster with the shapes like (point, line and polygon) based on the constraints like river, mountain, highway and buildings. After finding the clusters under the spatial data we can apply spatial joins for easy identification of locations in geographical area.

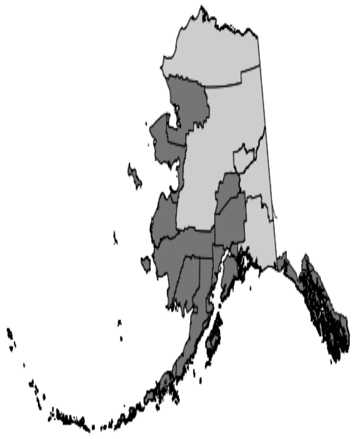
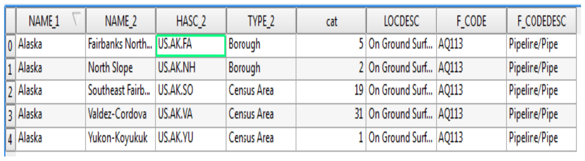


Fig. 6: Spatial join-Alaska region with Alaska pipelines



| | NAME_1 | NAME_2 | HASC_2 | TYPE_2 | cat | LOCDESC | F_CODE | F_CODEDESC |
|---|---|---|---|---|---|---|---|---|
| 0 | Alaska | Fairbanks North... | US.AK.FA | Borough | 5 | On Ground Surf... | AQ113 | Pipelire/Pipe |
| 1 | Alaska | North Slope | US.AK.NH | Borough | 2 | On Ground Surf... | AQ113 | Pipelire/Pipe |
| 2 | Alaska | Southeast Fairb... | US.AK.SO | Census Area | 19 | On Ground Surf... | AQ113 | Pipelire/Pipe |
| 3 | Alaska | Valdez-Cordova | US.AK.VA | Census Area | 31 | On Ground Surf... | AQ113 | Pipelire/Pipe |
| 4 | Alaska | Yukon-Koyukuk | US.AK.YU | Census Area | 1 | On Ground Surf... | AQ113 | Pipelire/Pipe |

Fig. 7: Spatial Join with Alaska pipeline with region

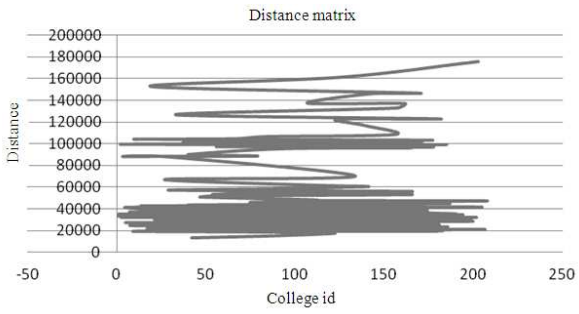


Fig. 8: Distance between the different college with respect to particular School Id 20010

The spatial joins are applicable only for the idle objects namely city, school, college and banks in the spatial objects. In the experiment, college (205 dataset) and school (2767dataset) point dataset were taken for analysis like nearest neighbour analysis, distance between the colleges and schools shown in Fig. 8. The school data set have attributes like id, name, address, grades and types. Similarly, the attribute of College Dataset are collegeid, name, address, degree, URL, type. Neareast neighbour analysis report for college point dataset shown in below:

Nearest Neighbor Analysis-college point data
Observed mean distance: 3011.73212725
Expected mean distance: 7488.66453847
Nearest neighbor index: 0.402172124519
Number of data point: 205
Z-Score:-16.3751005276
Nearest Neighbor Analysis-school point data
Observed mean distance: 802.629487441
Expected mean distance: 2140.12350239
Nearest neighbor index: 0.375038864133
N: 2767
Z-Score:-62.8909961022

The algorithm to compute a Z-Score is simple. Start with a list of numbers representing common values for something. Compute the mean of this list. The mean is just a simple average. Then compute the standard deviation of the list of numbers. The standard deviation is the average distance between each number in the list, and the mean of the list. Now take a new number that you want to compare to the list of numbers. Subtract the mean of the list from that number, then divide the result by the standard deviation of the list. The final result will be the Z-Score of the new number compared to the list of numbers.

In base study they did not focused on clustering by assuming that the cluster are already exist. Hence straight away they have taken towards spatial joins. But in our study first we use clustering technique to identify the clusters under the spatial data and then it is taken for the spatial query processing and also for spatial joins.

The data which is taken for the spatial join in the existing studies is only the line objects. But we have taken the various objects points namely city, school, college and banks, in the spatial objects (image) to find the spatial joins between these object points. Likewise in future it is possible to take the following combinations of spatial objects like point-point, line-line, polygon-polygon, point-line, line-point, point-polygon, polygon-line to find the spatial joins between them.

In the base study, without using algorithms with the properties they have taken two files namely R and S for R* tree comparisons. But it has high time consumption for comparisons. The proposed spatial join algorithms will take the optimum time for the comparisons. If we are applying the above strategies under the spatial data (image), it is very easy to identify or locate the exact geographical area which is needed for the user.

## RESULTS

The Fig. 9 shows that the performance comparisons chart based on the size of the pages with the size of the buffer. Comparing with the existing method, the proposed one gives better result without any extra cost and also the chart represents the multiples of bytes in terms of size of the pages with the buffer size.

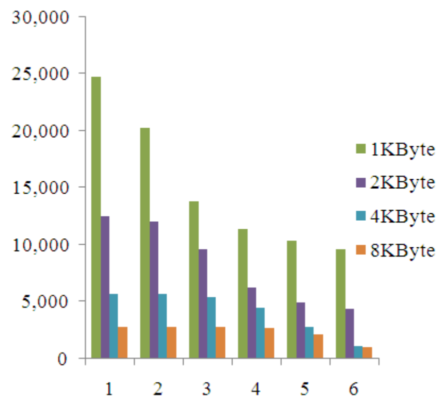


Fig. 9: Size of pages in buffer Vs buffer size

## DISCUSSION

**Performance comparisons of CPU and I/O time tunning:** Table 1 shows the CPU and I/O time tuning based on the buffer size 1024 kb and the objects are all point shape. In addition to preserve spatial locality in the buffer, this approach can be used without any extra cost. Hence call this approach as local plane-sweep order and the corresponding join algorithm as spatial join. Note that it is assumed in the algorithm sorted intersection test does not use sequences of rectangles as introduced originally, but sequences of entries for input and output and also assume that the spatial data to be taken for spatial join after applying the constraints based spatial clustering. Compared with the existing algorithm the proposed method provide less complexity with the best improvement of CPU time using R*-tree spatial joins with the constraints.

One of the key applications of spatial join is to find all the objects which either intersect or overlap with each other. Some variants of spatial join (e.g., distance join) are used in data mining for data analysis and clustering. It can also be used to process closest-pairs query, k-nearest neighbors query and ϵ-distance query (Gao *et al*., 2011).

**Future work:** There are some issues in spatial join that require further attention from research community. For processing spatial join queries, we usually follow filter and refine step in order. In some cases, some variants of this (e.g., interleaving) may give us more benefit. We can explore where probable variants can be beneficial and what information we need to collect for this. Although intersection join algorithms (e.g., R-tree join) can be directly extended for other types (e.g., distance join) it cause inefficient performance benefit. Various optimization techniques can be applied to remedy this.

Table 1: CPU and I/O time tunning (The data implemented up to by taking the buffer size 1024, the objects are all point shape)

| | | Size of pages | | | |
|---|---|---|---|---|---|
| | | 1KByte | 2KByte | 4KByte | 8Kbyte |
| Size of buffer | 0 | 24,727 | 12,479 | 5720 | 2837 |
| | 8 | 20318 | 12010 | 5720 | 2837 |
| | 32 | 13803 | 9589 | 5454 | 2822 |
| | 128 | 11359 | 6299 | 4474 | 2676 |
| | 512 | 10372 | 4964 | 2768 | 2181 |
| | 1024 | 9589 | 4372 | 1084 | 1027 |

Extending existing intersection join algorithms with various optimization criteria to other domain, it will be an interesting area for research.

## CONCLUSION

In this study the discussion was focused on the spatial join effects with the constraints-based spatial data without any extra cost. The constraint-based spatial clustering reduces the time taken to identify the required objects under the spatial image. The R*-tree concept reduce the number of search pages to combine spatial objects. By using this, CPU utilization time increases, the number of comparisons of spatial objects can be reduced and also reduces the I/O time.

## REFERENCES

Arge, L., O. Procopiuc, S. Ramaswamy, T. Suel and J. Vahrenhold *et al*., 2000. A unified approach for indexed and non-indexed spatial joins. Proceedings of the 7th International Conference on Extending Database Technology (EDBT'00), Springer Verlag, Berlin, Germany, pp: 413-429. DOI: 10.1007/3-540-46439-5_29

An, N., Z.Y. Yang and A. Sivasubramaniam, 2001. Selectivity estimation for spatial joins. Proceedings of the 17th International Conference on Data Engineering, Apr. 2-6, IEEE Xplore Press, Heidelberg, Germany, pp: 368-375. DOI: 10.1109/ICDE.2001.914849

Bae, W.D., S. Alkobaisi and S.T. Leutenegger, 2010. IRSJ: Incremental refining spatial joins for interactive queries in GIS. GeoInformatica, 14: 507-543. DOI: 10.1007/s10707-009-0089-0

Bae, W.D., S. Alkobaisi, S.H. Kim, S. Narayanappa and C. Shahabi, 2009. Web data retrieval: solving spatial range queries using k-nearest neighbor searches. Geoinformatica, 13: 483-514. DOI: 10.1007/s10707-008-0055-2

Cheng, R., S. Singh, S. Prabhakar, R. Shah and J.S. Vitter *et al*., 2006. Efficient join processing over uncertain data. Proceedings of the 15th ACM International Conference on Information and Knowledge Management, Nov. 05-11, ACM New York, USA., pp: 738-747. DOI: 10.1145/1183614.1183719

Francis, F.S. and P. Thambidurai, 2007. Efficient physical organization of r-trees using node clustering. J. Comput. Sci., 3: 506-514. DOI: 10.3844/jcssp.2007.506.514

Gao, Y., B. Zheng, G. Chen, Q. Li and X. Guo, 2011. Continuous visible nearest neighbor query processing in spatial databases. VLDB J., 20: 371-396. DOI: 10.1007/s00778-010-0200-z

Ibrahim, L.F. and H.A. Salman, 2011. Using hyper clustering algorithms in mobile network planning. Am. J. Applied Sci., 8: 1004-1013. DOI: 10.3844/ajassp.2011.1004.1013

Jacox, E.H. and H. Samet, 2007. Spatial join techniques. ACM Trans. Database Syst., 32: 7-51.

Mouratidis, K., D. Papadias and S. Papadimitriou, 2008. Tree-based partition querying: A methodology for computing medoids in large spatial datasets. VLDB J., 17: 923-945. DOI: 10.1007/s00778-007-0045-2

Papadias, D., 2001. Constraint-based processing of multiway spatial joins. Algorithmica, 30: 188-215 DOI: 10.1007/s00453-001-0005-y

Shekar, S. and S. Chawla, 2003. Spatial Databases: A Tour. 1st Edn., Prentice Hall, ISBN-10: 0130174807, pp: 262.

Shou, Y., N. Mamoulis, H. Cao, D. Papadis and D.W. Cheung, 2003. Evaluation of iceberg distance joins. Proceedings of the 8th International Symposium on Spatial and Temporal Databases (ISSTD'03), Springer, pp: 270-288.

Velmurugan, T. and T. Santhanam, 2010. Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. J. Comput. Sci., 6: 363-368. DOI: 10.3844/jcssp.2010.363.368