Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 175 (2020) 237–244

Procedia Computer Science

www.elsevier.com/locate/procedia

The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 9-12, 2020, Leuven, Belgium

Join processing in unbounded spatial data streams

Wendy Osborn[a,*]

[a]*Department of Mathematics and Computer Science, University of Lethbridge, Lethbridge, Alberta, T1K 3M4, Canada*

Abstract

The application of spatial joins for processing queries in spatial data streams is explored in this paper. Although their application has been studied in centralized and distributed systems, it has received little attention in spatial data streams. This work explores a particular issue with spatial join processing in spatial data streams - the lack of a bounded region of space from which the spatial objects are generated. Therefore, two strategies for join processing in spatial data streams in this situation are proposed. Both provide estimation of the common region shared by multiple spatial data streams to best process a spatial join between them. The strategies are evaluated and compared with a recently proposed approach. Results show that one strategy does an excellent job of estimating the common region given no existing bounded regions of space. Other results and future improvements are identified.

Keywords: spatial data streams; spatial join; stream query processing; performance

1. Introduction

Applications exist today that generate data continuously. It is impossible to store all continuous streaming data, as it becomes very large over time [3, 6]. Also, a data item in the stream may become invalid over time [3, 6]. Therefore, new approaches are required for managing data in data streams, as traditional approaches were proposed for managing stored data that is considered valid until it is explicitly deleted [3]. This scenario also applies to spatial data streams that generate and transmit object data (e.g. objects of non-zero area that represent regions of space) [11]. One such strategy that requires consideration in this context is the spatial join [15]. Given two object sets, a spatial join finds objects that are related based on a spatial predicate (e.g. overlap). A requirement for traditional spatial join processing is the existence of all objects when the join takes place, which is not the case for objects generated by a spatial data stream. Since objects are arriving continuously, each must be processed for the spatial join as soon as it arrives. Also, a decision must be made whether or not to keep the object, as it may be required for joining with a object that is arriving in the future [6].

* Corresponding author. Tel.: +1-403-329-2294 ; fax: +1-403-317-2882.
E-mail address: wendy.osborn@uleth.ca

1877-0509
10.1016/j.procs.2020.07.035

Existing strategies that utilize a spatial join have been proposed previously in the literature. Their use can be organized as follows: in a centralized database [14, 7, 18, 2, 8, 17], in a distributed database [1, 16, 9, 10, 5, 13], and in a spatial data stream [11, 12]. As we can see, significant work has been undertaken in the areas of centralized and distributed query processing. However, the main limitation for these strategies is that all objects in the participating spatial object sets must available for the joins. The only exceptions are the works of Kwon and Li [11] and Osborn [12]. Kwon and Li proposed a progressive spatial join strategy that joins the same pair of objects multiple times, beginning with a simplified representation (i.e. a minimum bounding box) and increasing the complexity of the representation for subsequent joins, in order to reduce the cost of the spatial join as simpler representations are computationally less expensive to join [11]. A limitation of this approach is that two objects - in particular, two objects that ultimate are matched based on the chosen spatial predicate - are processed several times. Osborn proposed several strategies that utilize both a conventional spatial join and a bit-array join (i.e. similar to a Bloom join [4]). They employ the idea of a common region between the two spatial data streams to identify objects that would participate in a join. One general limitation of this work is that the spatial extent that the data streams obtain their data from must be known in advance.

Therefore, this paper proposes two strategies that attempt to overcome the limitations of both previously proposed strategies stated above - not knowing in advance the overall spatial extent of the generated object set, and reducing the number of spatial joins that must take place. The first strategy (Sliding Window) estimates the common spatial extent (i.e. common region) shared by two data streams by creating it from an initial sliding window set of objects. The second strategy (Incremental) creates an initial common region from the first two objects to arrive as the spatial data stream processor, and incrementally (and selectively) grows the common region as more objects arrive from the spatial data streams. An experimental evaluation and comparison versus an existing approach that uses a pre-determined common region shows that the Sliding Window strategy achieves comparable results to that of the existing approach, and significantly outperforms the Incremental strategy.

The rest of this paper proceeds as follows. Section 2 summarizes the background information that is required for the proposed strategies. In Section 3, two strategies are proposed that attempt to overcome the limitation of an unknown spatial extent of the data streams. Section 4 summarizes the empirical evaluation of the two strategies, including a comparison against one of the previously proposed approaches. Section 5 provides a conclusion and future research directions for this work.

2. Background

This section summarizes some background that is required for this work. After an introduction of the symbolic notation used in this paper, a spatial data stream system, spatial join, and the *Spatial2* strategy [12] are summarized.

2.1. Preliminaries

The following symbols are used throughout this paper:

- S_1 and S_2: spatial data streams which transmit objects to the spatial data stream processor,
- E_1 and E_2: the extents in space that contains objects from S_1 and S_2 respectively,
- CR: a "common region", which is the region of space shared by both spatial data streams. It consists of the overlap of E_1 and E_2,
- RS: result stream,
- SW: sliding window.
- $numobj(SW)$: the number of objects currently in SW, and
- o_{swx}, $x = 1$ to $numobj(SW) - 1$: the objects in SW.

In Figure 1 a data stream system with two sensors is depicted. These sensors continuously generate and send data to a remote server for processing stream data, including query processing. Then, any results that are generated from processing on the sever are transmitted on a result data stream to a specific destination [6]. For spatial data streams, the data that is generated and transmitted is spatial data, which is some combination of point and/or object data.

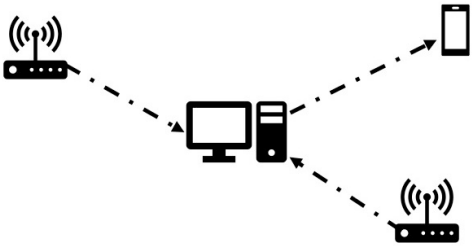


Fig. 1. Data Stream System

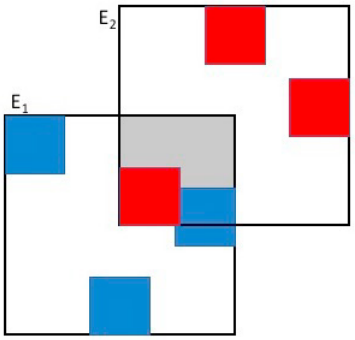


Fig. 2. Overlap of Two Regions (from [12])

A data stream processor is capable of storing only a limited amount of information that is sent from the streams. Several approaches exist [6] for selecting the data that will be stored at the data stream server. A sliding window is one such approach. It is a limited size memory or data storage that stores incoming stream data for some finite amount of time. When the sliding window has no more space left for incoming data, a decision must be made about which data to discard. In this work, the use of both the first-in-first-out (FIFO) and least recently used (LRU) strategies are utilized, to determine if one approach is superior to the other in terms of keeping the "best" data in the data stream.

Given spatial object sets, S_1 and S_2, a spatial join [15] relates pairs of objects from each set based on a spatial predicate (e.g. overlap). The following assumptions are made for this work. First, although any non-directional spatial predicate can be used, this work uses the overlap predicate. Second, this work also uses the nested-loop join for joining a portion of S_1 and S_2. Finally, a vector (i.e. geometric) representation of polygons is assumed, where a polygon is a sequence of connected points.

2.2. *Spatial2 Strategy*

Proposed in [12], the *Spatial2* strategy will be used for comparison against the new strategies. It is chosen as it was found to achieve high accuracy in smaller spatial data streams [12] and when compared to [11].

The strategy takes object input (o_1 and o_2) from two spatial data streams (S_1 and S_2) and processes them for spatial joins as they arrive. A sliding window SW of the most recent set of arriving objects from both streams is maintained. The strategy uses the concept of a common region CR to determine which spatial extent contains objects from both S_1 and S_2). Figure 2 depicts an example of a common region. Given spatial extents E_1 and E_2 which correspond to streams S_1 and S_2 respectively, the grey region is the spatial extent that is common to both E_1 and E_2. This is known as the common region CR, and is the only area that contains objects from both streams.

Spatial2 works by tests arriving objects o_1 and o_2 for overlap with the CR. The strategy is carried out in the following manner [12]:

1. First, the regions E_1 and E_2, which represent the regions in space that contain objects from S_1 and S_2 are transmitted to the stream query processor so that the common region CR between E_1 and E_2 can be determined.
2. Next, each of S_1 and S_2 transmits objects o_1 and o_2 respectively to the data stream processor.
3. If required, if SW does not have enough room for o_1 and o_2, then one or two objects must be chosen for removal. Spatial2 utilizes the First-in-First-out strategy for making this decision.
4. Object o_1 is tested for overlap with the CR. it is added to SW if the overlap test is successful, and o_1 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_2. Any pair that have a positive overlap result is transmitted on the result stream RS.
5. Object o_2 is tested for overlap with the CR. it is added to SW if the overlap test is successful, and o_2 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_1. Any pair that have a positive overlap result is transmitted on the result stream RS.
6. This is repeated from Step 2 while objects continue to be received from S_1 and S_2.

3. Strategies for Unbounded Spatial Data

In this section, two polygon-based nested-loop join processing strategies are proposed for spatial data streams. Both the Sliding Window and Incremental strategies attempt to overcome the limitation of having pre-determined spatial extents (and therefore, a pre-determined common region), as these may not be available, or would need to be significantly overestimated in order to be applied to existing strategies. Both strategies are described in summary and in detail below.

3.1. Sliding Window Strategy

In the Sliding Window strategy, the common region CR is formed using the spatial extents from the first $numobj(SW)/2$ objects that are transmitted from S_1 and S_2 respectively. At the same time, the first $numobj(SW)$ objects are processed to identify any spatial joins that exist between them. After CR is formed, all subsequent objects o_1 and o_2 that are transmitted from S_1 and S_2 are tested for overlap with CR to determine if they should be saved in SW for future joins. The strategy proceeds as follows:

1. First, the first $numobj(SW)/2$ objects from each of S_1 and $S2$ are obtained and join processed:
   (a) Each spatial data stream S_1 and S_2 transmits objects o_1 and o_2 respectively to the data stream processor.
   (b) Object o_1 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_2. Any pair that have a positive overlap result is transmitted on the result stream RS.
   (c) Object o_2 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_1. Any pair that have a positive overlap result is transmitted on the result stream RS.
2. Next, the common region CR is formed:
   (a) Spatial extent E_1 is formed from all objects o_{swx} in SW that were transmitted from S_1.
   (b) Spatial extent E_2 is formed from all objects o_{swx} in SW and were transmitted from S_2.
   (c) Finally, the common region CR is formed by determining the overlapped area between E_1 and E_2.
3. From here, the *Spatial2* strategy (summarized above) is applied to process joins with incoming objects from S_1 and S_2, and to determine which objects are to be added to and removed from SW.

3.2. Incremental Strategy

In the Incremental Strategy, CR is formed gradually as objects o_1 and o_2 arrive. An initial CR is formed by creating the minimum bounding rectangle that encompasses the first o_1 and o_2 that are transmitted from S_1 and S_2. Following this, for all subsequently transmitted o_1 and o_2, if either o_1 and/or o_2 overlap CR, then CR is increased the accommodate the new object(s), and the objects are added to SW (with space being made in SW for them if required). Spatial joins are also processed at this point. The strategy proceeds as follows:

1. An initial common region CR is created by forming a minimum bounding rectangle that encompasses o_1 and o_2.
2. Then, all subsequent objects o_1 and o_2 are processed as they arrive from S_1 and S_2:
   (a) Object o_1 is tested for overlap with the CR. If successful, CR is increased to accommodate o_1, and o_1 is added to SW. In addition, o_1 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_2. Any pair that have a positive overlap result is transmitted on the result stream RS.
   (b) Object o_2 is tested for overlap with the CR. If successful, CR is increased to accommodate o_2, and o_2 is added to SW. In addition, o_2 is evaluated for overlap with all other objects o_{swx} in SW that were transmitted from S_1. Any pair that have a positive overlap result is transmitted on the result stream RS.
   (c) If required, if SW does not have enough room for o_1 and o_2, then one or two objects are chosen for removal.

4. Evaluation

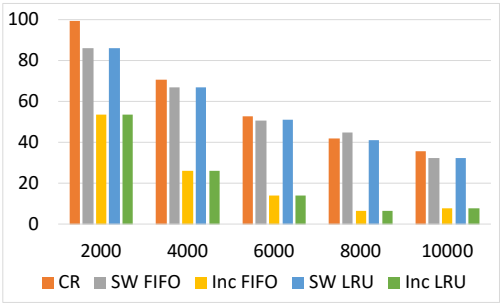
In this section, the empirical evaluation of the two spatial data stream join processing strategies is presented, along with a comparison against the *Spatial2* existing strategy [12]. The evaluation framework and methodology is summarized, followed by the summary and discussion of the results of the evaluation.

4.1. Framework and Methodology

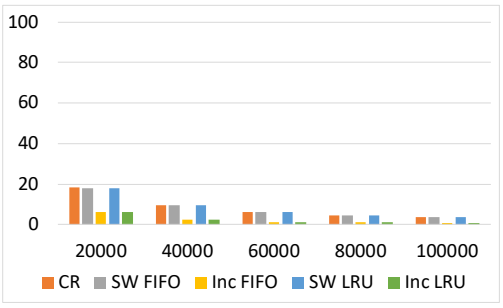
The experimental environment used for all experiments consisted of two simulated spatial data streams. The central stream query processor utilizes one sliding window for managing a subset of objects that were transmitted from both data streams. Both the First-In-First-Out (FIFO) and Least Recently Used (LRU) strategies are employed for selecting objects for removed when space is needed in the sliding window.

Both strategies proposed in Section 3 are implemented in Linux Centos 7 using C++. Several simulated spatial data streams that generated a sequence of $10x10$ rectangles were used for evaluating both strategies. This approach was used so specific aspects of the data could be controlled – in particular, the overall region of space covered by a stream of generated objects, as well as the overlap between the region of space covered by two streams. Altogether, 28 sets of rectangles were used in pairs for spatial joins, and groups into three evaluation sets. The first evaluation set used pairs containing 500, 1000, 1500 and 2000 rectangles, respectively. The second used pairs containing 2000, 4000, 6000, 8000, and 10000 rectangles, respective. Finally, the third set used pairs containing 20000, 40000, 60000, 80000, and 100000 rectangles, respectively. Each set of n rectangles are generated from a region of space of dimension $(\sqrt{n} * 10)x(\sqrt{n} * 10)$ (e.g. rectangles the1000-rectangle sets were generated from a region of space of size $310x310$). In addition, the overall region of space for each set of rectangles have 25% overlap between them.

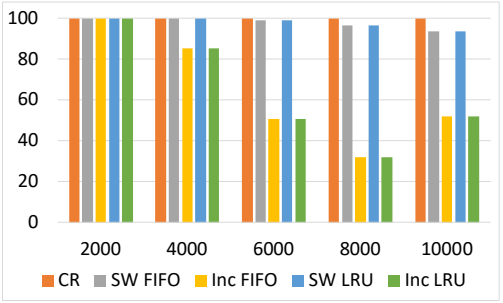
For both strategies in Section 3, two sets of tests were carried out that varied: 1) the number of objects sent through each data stream, and 2) the size of the sliding window. For each set of tests:



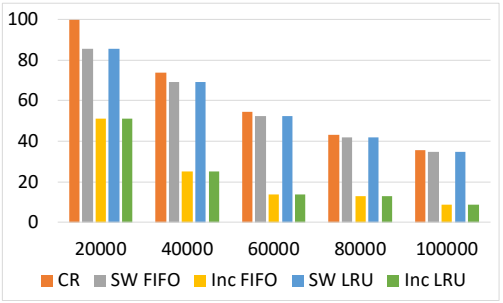
(a) Up to 10000 x 10000 Objects - WinSize 16K



(b) Up to 100000 x 100000 Objects



(c) Up to 10000 x 10000 Objects - WinSize 160K



(d) Up to 100000 x 100000 Objects - Winsize 160K

Fig. 3. Varying Number of Objects vs. Accuracy

- *Varying the number of objects.* Overall, 20 tests were carried out, which can be categorized into five groups:
  - 2000x2000, 4000x4000, 6000x6000, 8000x8000 and 10000x10000 spatial object stream pairs, with sliding window size 16000 bytes (i.e. 1000 rectangles),
  - 20000x20000, 40000x40000, 60000x60000, 80000x80000 and 100000x100000 spatial object stream pairs, with sliding window size 16000 bytes (i.e. 1000 rectangles),
  - 2000x2000, 4000x4000, 6000x6000, 8000x8000 and 10000x10000 spatial object stream pairs, with sliding window size 160000 bytes (i.e. 10000 rectangles),
  - 20000x20000, 40000x40000, 60000x60000, 80000x80000 and 100000x100000 spatial object stream pairs, with sliding window size 160000 bytes (i.e. 10000 rectangles),
- *Varying the window size.* Overall, 16 tests were carried out, which can be categorized into five groups:
  - 8000, 16000, 24000 and 32000 bytes (i.e., 500, 1000, 1500 and 2000 rectangles, respectively), using the 4000x4000 spatial object stream pair.
  - 8000, 16000, 24000 and 32000 bytes (i.e., 500, 1000, 1500 and 2000 rectangles, respectively), using the 40000x40000 spatial object stream pair.
  - 80000, 160000, 240000 and 320000 bytes (i.e., 5000, 10000, 15000 and 20000 rectangles, respectively), using the 4000x4000 spatial object stream pair.
  - 80000, 160000, 240000 and 320000 bytes (i.e., 5000, 10000, 15000 and 20000 rectangles, respectively), using the 40000x40000 spatial object stream pair.

For all tests, in addition to the final spatial join stream, two performance factors were recorded: the CPU time over the entire join at the stream query processor, and the number of joined tuples in the final result stream. Given this latter value, Accuracy was determined by calculating the number of tuples in the overall join result (i.e assuming a full spatial join with no sliding window) that were also determined in each strategy.

4.2. Results and Discussion

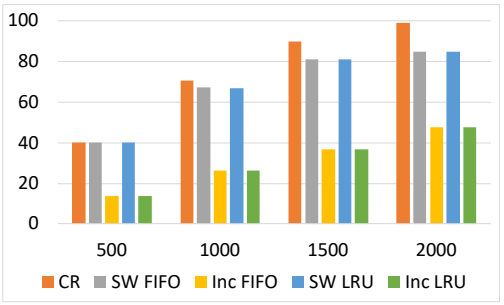
This section presents the results and discussion of the evaluation. The results for varying spatial data stream sizes are presented first, followed by the results for the varying sliding window sizes. In all result charts, the following is represented: *CR* represents the outcome of the *Spatial2* strategy [12], *SW FIFO* represents the outcome of the Sliding Window strategy where a FIFO strategy is utilized for selecting objects for removal from SW, *Inc FIFO* represents the outcome of the Incremental strategy where a FIFO strategy is utilized for selecting objects for removal from SW, *SW LRU* represents the outcome of the Sliding Window strategy where a LRU strategy is utilized for selecting objects for removal from SW, and *Inc LRU* represents the outcome of the Incremental strategy where a LRU strategy is utilized for selecting objects for removal from SW.

4.2.1. Varying Number of Objects

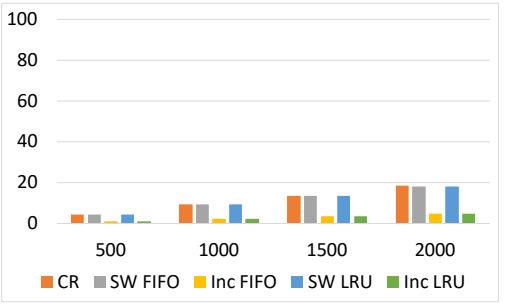
Figure 3 depicts the accuracy results, respectively, for varying the size of the spatial data streams while using sliding windows of 16,000 bytes and 160,000 bytes respectively. As we can see, results show that overall, there is a significant decrease in accuracy as the number of objects being transmitted from the data streams increase. In addition, we see that with respect to accuracy, the Sliding Window strategy significantly outperforms the Incremental Strategy. Finally, with respect to accuracy we see there is little to no difference between the use of FIFO and LRU for selecting which objects to remove from the sliding window.

In Figures 3(a) and 3(b), the accuracy of the Sliding Window strategy is high and almost equal to that achieved by the *Spatial2* strategy, and for smaller data sizes is at least 80%. This is significantly better than the Incremental strategy, which at best only achieves a 60% accuracy. Both strategies, however, have a significant decrease in accuracy as the number of objects increase, with less than 20% in the larger stream sizes. With an increase in sliding window size, which is shown in Figures 3(c) and 3(d), improvements - significant ones early on, and more modest one with larger stream sizes - are achieved by the Sliding Window strategy. However, again it must be noted that the Sliding Window strategy performs almost as well as the *Spatial2* strategy. Therefore, having a pre-determined spatial extent from which objects are streamed from is not necessary, and comparable results are achieved.

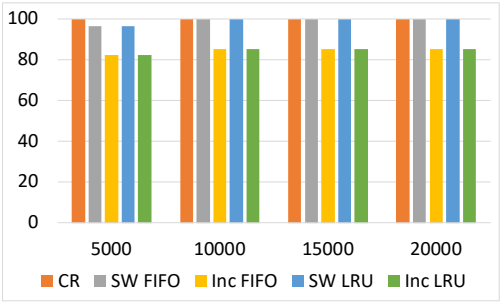
With respect to running time (not shown due to space limitations), we see a significant increase in running time as the size of the data streams increase. An unusual finding is that the Incremental strategy has an initial lower running



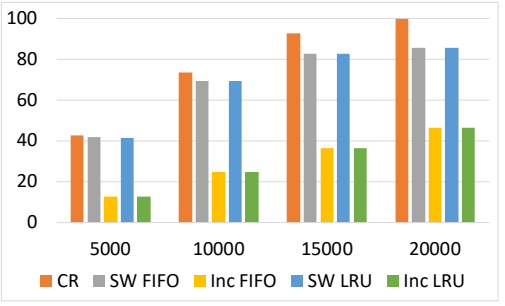
(a) Up to 32K, 4000 x 4000 Objects



(b) Up to 32K, 40000 x 4000 Objects



(c) Up to 320K, 4000 x 4000 Objects



(d) Up to 320K, 40000 x 40000 Objects

Fig. 4. Varying Sliding Window Size vs. Accuracy

time over the other approaches - including the Sliding Window strategy - but results in a higher running time as the number of objects increase. However, there is a local explanation for this. Since the Sliding Window strategy creates CR initially with $numobj(SW)$ objects, where the Incremental strategy only creates CR with two objects initially, more processing is required by Sliding Window to process not rectangles initially. However, Sliding Window only performs overlap checking afterwards, where Incremental must adjust CR numerous times. So for larger data sets, Incremental would lead to higher processing times overall.

4.2.2. *Varying Sliding Window Size*

Figure 4 depicts the accuracy results for varying the size of the sliding window under specific data size conditions. We can see that, the results show overall that for larger data sizes, the accuracy does improve as the size of the sliding window increases, but overall the size of the dataset requires a larger sliding window size in order to achieve better performance. We also see that there is little to no difference between the use of FIFO and LRU for selecting which objects to remove from the sliding window.

We see similar trends to those found for varying the number of objects transmitted from the data streams. Specifically, we see that the Sliding Window Approach achieves accuracies that are similar to those obtained by the *Spatial2* strategy, and that the Incremental strategy performs poorly. The only exception is for the larger sliding windows and the $4000x4000$ spatial stream join, where the Incremental strategy does achieve 80% accuracy.

With respect to running time (not shown due to space limitations), the running time increases as the size of the sliding window increases. We also observe the same trends as above - that the Sliding Window strategy has a higher running than the Incremental strategy for the smaller data set sizes but this is reversed as the dataset size and the sliding window size increase. Although the window size is what is increasing here, similar reasoning applies as it did for the varying dataset size. The Sliding Window strategy only processes CR once once $numobj(SW)$ objects have arrived, where the Incremental strategy must adjust CR numerous times, regardless of the window size.

5. Conclusion

In this paper, two strategies are proposed for processing spatial joins in spatial data streams. The goals of both strategies are: 1) to attempt to estimate the common spatial extent shared by both input spatial data streams so that having the spatial extents in advance is no longer required, and 2) to reduce the number of objects that will participate in a spatial join if there is no chance that an object will join with other objects. The first strategy, Sliding Window, estimates the common region by using the first $numobj(SW)$ objects that arrive from the spatial data streams. The second strategy forms an initial common region from the first objects - one from each stream) that arrive, and grow the common region incrementally as other objects that overlap the region arrive from the spatial data streams.

An experimental evaluation and comparison versus an existing approach that uses a pre-determined common region shows that the Sliding Window approach significantly outperforms the Incremental approach, and achieves accuracies that are equal to those achieved by the existing approach.

Future research directions include the following: 1) considering spatial data streams where the arrival times of objects differ, and 2) comparing the strategies with larger spatial data streams and against the original progressive join algorithm proposed by [11].

Acknowledgements

The Author would like to thank the referees for their constructive comments on the initial draft of this paper.

References

[1] Abel, D., Ooi, B., Tan, K.L., Power, R., Yu, J., 1995. Spatial join strategies in distributed spatial dbms, in: Proceedings of the 4th International Symposium on Advances in Spatial Databases.
[2] Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., Vitter, J., 1998. Scalable sweeping-based spatial join, in: Proceedings of the 24th International Conference on Very Large Databases, pp. 570–581.
[3] Babu, S., Widom, J., 2011. Continous queries over data streams. SIGMOD Record 30, 109–120.
[4] Bloom, B.H., 1970. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13, 422–426.
[5] Farruque, N., Osborn, W., 2014. Efficient distributed spatial semijoins and their application in multiple-site queries, in: Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications, IEEE. IEEE Computer Society.
[6] Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques. Morgan Kaufmann.
[7] Huang, Y.W., Jing, N., Rundensteiner, E., 1997. Integrated query processing strategies for spatial path queries, in: Proceedings of the 13th International Conference on Data Engineering, pp. 477 –486. doi:10.1109/ICDE.1997.582010.
[8] Jacox, E., Samet, H., 2007. Spatial join techniques. ACM Transactions on Database Systems 32.
[9] Kalnis, P., Mamoulis, N., Bakiras, S., Li, X., 2006. Ad-hoc distributed spatial joins on mobile devices, in: Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium.
[10] Karam, O., Petry, F., 2006. Optimizing distributed spatial joins using R-trees, in: Proceedings of the 43rd ACM Southeast Conference.
[11] Kwon, O., Li, K.J., 2011. Progressive spatial join for polygon data stream, in: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM.
[12] Osborn, W., 2019. Exploring bit arrays for join processing in spatial data streams, in: Proceedings of the 22nd International Conference on Network-Based information Systems, pp. 73–85.
[13] Osborn, W., Zaamout, S., 2016. Using spatial semijoins over multiple sites in distributed spatial query processing. Canadian Journal of Electrical and Computer Engineering 39, 71–81.
[14] Patel, J., DeWitt, D., 1996. Partition based spatial-merge join, in: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, pp. 259–270.
[15] Shekhar, S., Chawla, S., 2003. Spatial Databases: A Tour. Prentice Hall, New Jersey.
[16] Tan, K.L., Ooi, B., Abel, D., 2000. Exploiting spatial indexes for semijoin-based join processing in distributed spatial databases. IEEE Transactions on Knowledge and Data Engineering 12, 920–937.
[17] Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., Chen, G., 2012. Towards parallel spatial query processing for big spatial data, in: Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 2085–2094.
[18] Zhou, X., Abel, D., Truffet, D., 1998. Data partitioning for parallel spatial join processing, in: Geoinformatica, Springer-Verlag. pp. 175–204.