Stream Mining Revisited

Sayaka Akioka
School of Interdisciplinary Mathematical Sciences
Meiji University
Tokyo, Japan 164–8525
Email: akioka@meiji.ac.jp

***Abstract*—Big data applications have become popular in recent years. Stream mining is one of the major data mining methodologies, which are frequently used in big data applications. Stream mining differenciates itself from the other big data applications for its severe requirement, and is also known for its changing behaviros according to the characteristics of input data. The problem is, however, the parameters, or methodologies for data characterization are not clearly defined yet. There is no study investigating explicit relationships between the characteristics of input data, and the behaviors of stream mining applications. Therefore, the current optimization methodology for stream mining is basically heuristic. This paper provides comprehensive survey on modeling stream mining to seek the strategy for this modeling problem.**

***Keywords*—stream mining; modeling; characterization.**

I. Introduction

Big data applications have become popular in recent years. These big data applications are supposed to collect gigantic amount of data from various data sources, analyze these data from several points of view, uncover new findings, and then provide totally new values. Compared to the conventional applications, big data applications need to handle extremely huge amounts of data, and this situation leads high, and increasing demand for the computational environment, which accelerates, and scales out big data applications. The serious problem here, however, is that the behaviors, or characteristics of big data applications are not clearly defined yet. There is no established model for big data applications.

Big data applications can be classified into several categories depending on the characteristics of data usage. Among these big data applications, this paper has special focus on stream mining applications. A stream mining application is such an application that analyzes data in a line. That is, the target data arrive one after another in chronological order. A stream mining application differenciates itself from the other big data applications for its severe requirement. A stream mining application needs to finish the analysis on the fly. In many cases, there is no chance to save the target data somewhere to revisit the data later. Algorithms specialized for stream mining applications (stream mining algorithms) are intensively studied [1]–[30], and Gaber et al. gave an excellent survey report on these algorithms [31].

High performance computing community has been investigating data intensive applications, which analyze huge amount of data as well. Raicu et al. pointed out that data intensive applications, and stream mining applications are fundamentally different from the viewpoint of data access patterns. Therefore, the strategies for speed-up of data intensive applications, and stream mining applications have to be radically different [32]. Many data intensive applications often reuse input data, and the primary strategy of the speed-up is locating the data close to the target CPUs. Stream mining applications, however, rarely reuse input data, and the strategy for data intensive applications does not work in many cases.

Modern computational environment has been evolving mainly for speed-up of benchmarks such as Linpack [33], or SPEC [34]. These benchmarks are relatively scalable according to the number of CPUs. Stream mining applications are not scalable to the number of CPUs in many cases. Current computational environment is not necessarily ideal for stream mining applications for this reason. Additionally, many researchers from machine learning domain, or data mining domain point out that the behavior, or execution time of a stream mining application varies according to the characteristics, or features of input data. The problem is, however, the parameters, or the methodologies for data characterization are not clearly defined yet. There is no study investigating explicit relationships between characteristics of input data, and behaviors of stream mining applications. Therefore, the current optimization methodology for stream mining is basically heuristic.

The major purpose of this paper is to provide a comprehensive survey on modeling stream mining applications. This paper focuses on generic models for stream mining applications, but does not cover the details of execution models of existing middlewares, or frameworks for stream mining applications. The primary purpose of this paper is to find keys to generalize stream mining applications, and clues to connect characteristics of input data, and behaviors of stream mining applications. This paper also tries to give some considerations on the strategy to address this modeling problem based on the survey. The rest of this paper is organized as follows. Section 2 introduces conventional proposals for stream mining algorithms. Section 3 discusses possible strategies, or directions for a stream mining application model. Section 4 concludes this paper.

II. Models of Stream Mining Algorithms

A. A Three-layer Model

Junghans et al. proposed a three-layer model, which is illustrated as the shaded part in Figure 1. They argued that most stream mining algorithms follow this three-layer model [35]. First, the filter component filters incomping data as necessary for the purpose of sampling, or load shedding. Secondly, the online mining component analyzes the original incoming data stream, or the filtered stream. Thirdly, the results of the online mining component will be stored in the synopsis, which is the

 ISBN: 978-1-61208-457-2

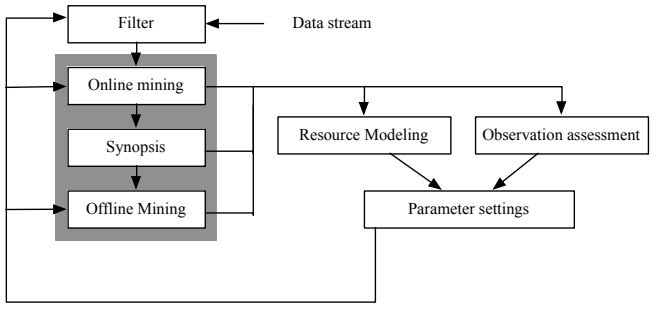

Fig. 1. Extended three-layer model.

second layer of the three-layer model. Here, synopsis indicates sketches, windows, or other dedicated data structures such as a pattern tree; those are often utilized in stream mining algorithms. Finally, the offline mining component answers user queries by accessing information stored in the synopsis. Here, the offline mining component does not need to fulfill the one pass requirement of stream mining.

As Junghans et al. developed their stream mining model with a background of embedded devices, they put an asumption that stream mining is conducted on limited available resources, such as limited number of CPUs, or limited amount of memory. Stream mining algorithms are often optimized for the better performance even under these constraints. Junghans et al., therefore, extended their three-layer model to include these optimization functionalities. Junghans et al. also extended the three-layer model for the better quality of the results by stream mining. The principle of this extension is to optimize the influential parameters in stream mining algorithms. This optimization contributes to the relaxed resource requirements, or the better quality of the mining results. Figure 1 illustrates this extended three-layer model. The shaded part of the figure is the original three-layer model as already described above, and the right part of the figure is the extension. The resource monitoring, and the observation assessment component collect information about the current system state. Based on the monitoring by the resource monitoring, and the observation assessment, the parameters are decided whether they should be adapted, or not. Then, the new parameters are set, and the stream mining algorithm run with the updated parameters.

B. Stream Mining and Data Dependencies

Akioka et al. proposed another stream mining model [36], and the modeling put the focus on data dependencies. Figure 2 illustrates the overall model of stream mining algorithm. The model shown in Figure 2 is quite similar to the model by Junghans et al, while Figure 3 illustrates the detailed model of stream mining algorithms. Figure 3 depicts data dependencies, and control dependencies, and these dependencies lie among threads, or processes in one stream mining algorithm.

In Figure 2, a stream mining algorithm consists of two parts, stream processing part, and query processing part. The stream processing part consists of stream processing modules, sketches, and analysis modules. First, the stream processing module in the stream processing part picks the target data

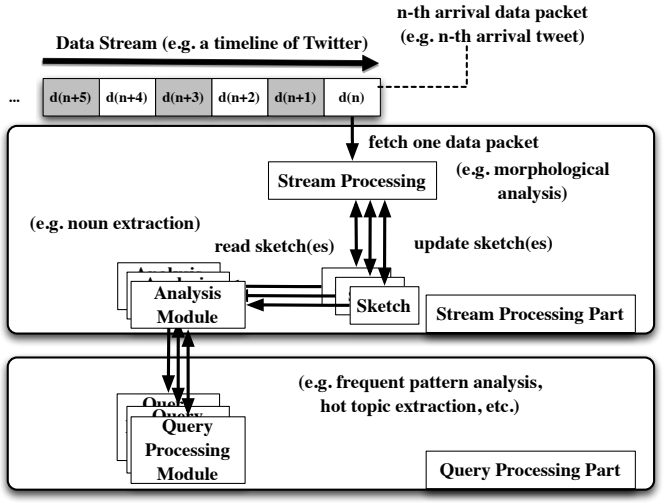

Fig. 2. A model of stream mining algorithms.

unit, and executes a quick analysis over the data unit. The quick analysis can be a preconditioning process such as a morphological analysis, or a word counting. Second, the stream processing module in stream processing part updates the data in one or more sketches. After this update, the data in the sketch(es) contains the latest results of the execution by the stream processing part. That is, the sketch(es) keeps the intermediate analysis, and the stream processing module updates the analysis incrementally as more data units are processed. Third, the analysis module in stream processing part reads the intermediate analysis from the sketch(es), and extracts the essence of the data in order to complete the quick analysis in the stream processing part. Finally, the query processing part receives this essence for further analysis, and the whole process for the target data unit is completed.

Based on the model shown in Figure 2, the major responsibility of the stream processing part is to preprocess each data unit for the further analysis, and that the stream processing part has the huge impact over the latency of the whole process. Therefore, the stream processing part also needs to finish the preconditioning of the current data unit before the next data unit arrives. Otherwise, the next data unit will be lost as there is no storage for buffering the incoming data. The query processing part takes care of the offline part of the analysis, and does not suffer from the strict requirement of stream mining.

Figure 3 focuses only on stream processing part as this is the part which impacts the overall performance. The figure illustrates data dependencies between two processes analyzing data units in line, and data dependencies inside each process. The assumption is that each process analyzes its own (different) data unit. The left top flow represents the stream processing part of the preceding process, and the right bottom flow represents the stream processing part of the successive process. Each flow consists of six stages; read from sketch(es), read from input, stream processing, update sketch(es), read from sketch(es), and analysis. An arrow represents a control flow, and a dashed arrow represents a data dependency. There are three data dependencies in total. These data dependencies are introduced by control flow for the correct executions, and the summary for these dependencies is as follows.

- The processing module in the preceding process should finish updating the sketch(es) before the processing module in the successive process starts reading the sketch(es) (Dep.1 in Figure 3).
- The processing module should finish updating the sketch(es) before the analysis module in the same process starts reading the sketch(es) (Dep.2 in Figure 3).
- The analysis module in the preceding process should finish reading the sketch(es) in the successive process starts updating the sketch(es) (Dep.3 in Figure 3).

C. Stream Mining with Multiple Data Streams

Wu et al. pointed out that many of the existing researches on stream mining assume that there is one data stream, and they proposed formal definition of mining over multiple data streams [37]. In actual situations, the assumption with multiple data streams is more realistic than the single data stream. Therefore, the formal definition of the problems with multiple data streams is more practical, and reasonable.

Accordng to Wu et al., multiple data stream mining should be approached in a separate way from the way for the single data stream mining. First, multiple data streams are from many local data sources to generate distributed data streams independently. These data sources are not capable of processing more than simple data preconditioning, or saving all the generated data. Second, multiple data stream mining is supposed to process the mining across the data streams, not only on one single data stream. Third, these multiple data streams are not modeled as one single huge data stream with different attributes. Timestamp on each data is not under uniform criteria in many cases. Sampling rate of the data is different. The format of the generated data, or privacy concern is not controlled. There is no reason to handle these data streams as if one single data stream.

Wu et al. represent each data in a data flow as a quadruple of the form (s, t, f, v), where s is the identification of the place, t is the time or sequence number identifying the event, f is a function, and v is a value vector of the output. Here, event refers whether data generation, or some other data processing. Each flow is a set of the quadruples, and fulfills the following properties.

- Each source specifies a single function to generate a single flow;
- For any pair of events, e_1 and e_2, that occur at the same source, if the two events have the same function invocation, and e_1 occurs before e_2, the value t of e_1 is smaller than that of e_2;
- For any pair of events, e_1 and e_2, that occur at different sources, there is no function or rule between e_1 and e_2.

In addition to these properties, flows can have some additional properties:

- Homogeneous or heterogeneous: A pair of flows is said to be homogeneous (or heterogeneous) if the respective sources at which the two flows generate specify the same (or different) function(s), which are checked in terms of initial conditions and output domain;
- Relational: A pair of flows, indicated by f_1 and f_2, is said to be relational if the value vectors of f_1 and value vectors of f_2 satisfy some relationship r (the relationship refers to values; events are independent).

Wu et al. also gave some considerations comparing multiple data streams, and other data stream models.

- Single stream with one dimension: This is the simplest model, and usually generated at a simple data stream application.
- Single stream with multiple dimensions: This applies to the applications in which there are multiple parameters, or attributes to be collected, and observed for each event occuring at a single source. The main difference between this model and multiple data streams is that single stream with multiple dimensions handles events of the same function invocation at a single source basically, while multiple data streams can invoke multiple functions distributed on different sources.
- Multiple data streams: This model is applicable to many real applications with multiple sources. These sources can be the same kind of devices, which are distributed at geometrically scattered locations. Basically, the multiple data sources can be viewed as a set of one or more dimensional single data streams.

III. Discussions

A. Comparison of the Three Models

Wu et al. defined multiple data streams, and mining multiple data mining (we refer to their model as MDS). The definition itself is beneficial in order to clarify the problem. Considering multiple data streams is more realistic as well. The point is, however, multiple data streams are still a set of single data streams as pointed out in their paper. Therefore, the priority for addressing this modeling problem should be the solid methodology for modeling the stream mining with one single data stream. How to superpose several models of stream mining models of a single data stream will be the next step.

Junghans et al. proposed their stream mining model in the context of embedded devices, such as sensors activated by batteries, and connected to the network by wireless (we refer to their model as Three-layer model). On the other hand, Akioka et al. proposed their stream mining model in the context of high performance computing (we refer to their model as DAP). Both of them proposed quite similar generic models for stream mining algorithms, and the discussions for the restrictions, and requirements for general stream mining are also in the same direction. These approximate models, however, do not deeply contribute for the strategy of scaling out stream mining algorithms. For the better choice of computational environment, size, allocations, preliminary estimations for resource requirements are indispensable. In this context, Akioka et al. proposed a model with data dependencies, and control dependencies. This model is quite similar to a task

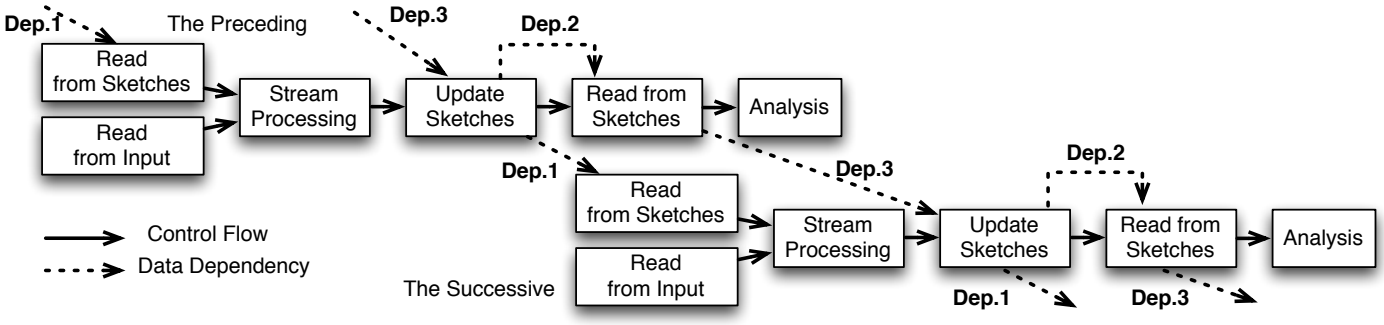

Fig. 3. Data dependencies of the stream processing part in two processes in line.

graph, which often used to solve scheduling problems. The problem is, however, there is no model for estimations of resources, or durations.

Junghans et al. put monitoring, and parameter update functionality into their model. They also picked up main memory, CPU cycles, bandwidth, and battery power as resources, and discussed stream properties (the stream rate, and characteristics of its individual elements such as value range, distribution, and size), input parameters, and query parameters influence the resource requirements of any stream mining algorithms. The solid models for these resources, or elements mentioned above are not proposed, however. The actual estimations are based on heuristics.

Table I summarizes the discussions in this section comparing the three models. As summarized in the table, there is no solid model proposal to fulfill the requirements for solving the load balancing problem, or scheduling problem of general stream mining applications. Here, we would like to remind you that a stream mining application changes its behavior according to the characteristics of input data, and that there is no model for input data. That is, everything is heuristic now for stream mining applications. Table I clearly shows that there is no successful project to give a solution on input data problem, and behavior characterizaion. These left problems are unavoidable for a direct solution for optimum execution of stream mining applications. We also need to be careful on the blanks regarding resource estimation. Currently, all of the modelings here heavily rely on a heuristic way to estimate the required resource, including the number of CPUs, or the duration of each stage of a stream mining application. There is no model here as well. Of course, as the application changes its behavior with input data, this problem is heavily connected to the input data problem. We still need to remember, however, that we have to prepare task graphs for all the stream mining applications without resource estimation models.

TABLE I. COMPARISON OF THE THREE MODELS.

	MDS	Three-layer model	DAP
generic model (single stream)	no	yes	yes
data/control dependencies	-	no	yes
resource estimation	-	no	no
input data characterization	-	no	no
input/behavior characterization	-	no	no
generic model (multiple streams)	yes	no	no
data/control dependencies	no	-	-
resource estimation	no	-	-
input data characterization	no	-	-
input/behavior characterization	no	-	-

B. Things to be Considered

We discussed how to understand, and model stream mining applications above. Here, another option for the optimum execution of stream mining applications would be a large-scaled cloud computing environment. If there is a good way to migrate whole, or some parts of running stream mining applications from one cloud environment to another environment, the restriction for the resource environment becomes loose. The challenges for this option will include the following items.

- Practical cloud computing environment with task migrations: This option requires any part of the implementation of stream mining to be ready for migration on the fly. Although there are many researchers, or products that enable task migrations, however, the question is how much they are practically usable. The time when those migration techniques are intensively studied, applications with migrations were implemented by people with background of computer architecture, parallel computing, or optimization techniques of programs. On the contrary, the major implementers of big data applications are more various. They are not necessarily with the detailed background of computer science. The point is how much those implementers accept, and happily utilize the migration techniques.

- Consistency and preservation of the data and results: As repeated in this paper, stream mining is a continuous, one-way, one-pass application. There is no way to save input data, or intermediate output without stopping the current execution. Once you stop the execution, you will lose both the input data, and the expected results while you are suspending the problem. Even if you resume the program, you will not be able to acquire intrinsic results for a while, as many of stream mining algorithms rely on the results from the previous data input. How to preserve the whole flow of stream mining should be an inescapable problem.

- Migration management: Even if a good methodology for migration is established to solve the problem mentioned above, there is another problem. The problem is the strategy for migration. There should be an

algorithm to decide which part of the whole program should be where, and when. This is basically load balancing problem, or scheduling problem. Therefore, we face the same modeling problem again. This time, we need to model both the behavior of stream mining applications, and current computational environment.

- Geographical placement: Except the case when the whole process (collection of input data, analysis with stream mining algorithms, and acquisition of the results) is performed in house, data source, and the actual computational environment are geographically scattered. Actually, this situation is quite common. Additionally, many of the current cloud computing services are employed in one place, or similar. This situation means that only a few points in the Internet accept almost all the input data collected all over the world. The inbound network load will be immeasurable, and have serious impact over performance of each of stream mining applications.
- Data privacy: Data privacy is one of the serious problems in recent concerns. Once you start migrating the whole, or some parts of stream mining applications, they will hop around with the data. One of the reasons why big data applications have became attractive rapidly is that many organizations have their own huge data without significance. The big data boom suggested that there is possible value in the huge sleeping data. The story will be different, however, once the data start moving around in the cloud, and it is difficult to protect the data from sniffing. People will become more conscious, and the boom will shrink.

IV. Conclusions

This paper provided a survey on generic modeling of stream mining. The results of the survey suggested that there is no successful solid modeling to address the problems surrounding stream mining applications. Even though there are several research projects sharing the same problem, however, the level of modeling is more abstract than the level for practical use. Currently, no project is free from heuristic.

The last half of the discussion in this paper argued another possible approach for the better environment for stream mining applications, beside direct modeling of stream mining. Although the discussion part might show some other directions, however, we could not find a brilliant strategy to solve the problem. We keep seeking the solution with a broader view.

Acknowledgment

This work was supported by JSPS KAKENHI, Grant-in-Aid for Young Scientists (B), and its Grant Number is 15K21423.

References

[1] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and k-medians over data stream windows," in *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '03. New York, NY, USA: ACM, 2003, pp. 234–243. [Online]. Available: http://doi.acm.org/10.1145/773153.773176

[2] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load shedding in a data stream manager," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 309–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315451.1315479

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. New York, NY, USA: ACM, 2002, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/543613.543615

[4] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Surfing wavelets on streams: One-pass summaries for approximate aggregate queries," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 79–88. [Online]. Available: http://dl.acm.org/citation.cfm?id=645927.672174

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 81–92. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315451.1315460

[6] ——, "A framework for projected clustering of high dimensional data streams," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 852–863. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316763

[7] ——, "On demand classification of data streams," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 503–508. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014110

[8] G. Cormode and S. Muthukrishnan, "What's hot and what's not: Tracking most frequent items dynamically," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 249–278, Mar. 2005. [Online]. Available: http://doi.acm.org/10.1145/1061318.1061325

[9] P. Yu, J. Pei, J. Han, H. Wang, G. Dong, and L. V. Lakshmanan, "Online mining of changes from data streams: Research problems and preliminary results," in *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Prcessing of Data Streams*, 2003.

[10] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE Trans. on Knowl. and Data Eng.*, vol. 15, no. 3, pp. 515–528, Mar. 2003. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2003.1198387

[11] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '03. New York, NY, USA: ACM, 2003, pp. 30–39. [Online]. Available: http://doi.acm.org/10.1145/780542.780548

[12] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/347090.347107

[13] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 97–106. [Online]. Available: http://doi.acm.org/10.1145/502512.502529

[14] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 685–694.

[15] C. Ordonez, "Clustering binary data streams with k-means," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD '03. New York, NY, USA: ACM, 2003, pp. 12–19. [Online]. Available: http://doi.acm.org/10.1145/882082.882087

[16] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: Implications for previous and future research," *Knowl. Inf. Syst.*, vol. 8, no. 2, pp. 154–177, Aug. 2005. [Online]. Available: http://dx.doi.org/10.1007/s10115-004-0172-7

 ISBN: 978-1-61208-457-2

[17] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 226–235. [Online]. Available: http://doi.acm.org/10.1145/956750.956778

[18] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Mining data streams under block evolution," *SIGKDD Explor. Newsl.*, vol. 3, no. 2, pp. 1–10, Jan. 2002. [Online]. Available: http://doi.acm.org/10.1145/507515.507517

[19] S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Adaptive, hands-off stream mining," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, pp. 560–571. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315451.1315500

[20] M. Last, "Online classification of nonstationary data streams," *Intell. Data Anal.*, vol. 6, no. 2, pp. 129–147, Apr. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1293986.1293988

[21] Q. Ding, Q. Ding, and W. Perrizo, "Decision tree classification of spatial data streams using peano count trees," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. New York, NY, USA: ACM, 2002, pp. 413–417. [Online]. Available: http://doi.acm.org/10.1145/508791.508870

[22] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proceedings of the 28th International Conference on Very Large Data Bases*, ser. VLDB '02. VLDB Endowment, 2002, pp. 346–357. [Online]. Available: http://dl.acm.org/citation.cfm?id=1287369.1287400

[23] P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying representative trends in massive time series data sets using sketches," in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 363–372. [Online]. Available: http://dl.acm.org/citation.cfm?id=645926.671699

[24] Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in *Proceedings of the 28th International Conference on Very Large Data Bases*, ser. VLDB '02. VLDB Endowment, 2002, pp. 358–369. [Online]. Available: http://dl.acm.org/citation.cfm?id=1287369.1287401

[25] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD '03. New York, NY, USA: ACM, 2003, pp. 2–11. [Online]. Available: http://doi.acm.org/10.1145/882082.882086

[26] D. Turaga, O. Verscheure, U. V. Chaudhari, and L. Amini, "Resource management for networked classifiers in distributed stream mining systems," in *Data Mining, 2006. ICDM '06. Sixth International Conference on*, Dec 2006, pp. 1102–1107.

[27] D. S. Turaga, B. Foo, O. Verscheure, and R. Yan, "Configuring topologies of distributed semantic concept classifiers for continuous multimedia stream processing," in *Proceedings of the 16th ACM International Conference on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 289–298. [Online]. Available: http://doi.acm.org/10.1145/1459359.1459398

[28] B. Thuraisingham, L. Khan, C. Clifton, J. Maurer, and M. Ceruti, "Dependable real-time data mining," in *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, May 2005, pp. 158–165.

[29] N. K. Govindaraju, N. Raghuvanshi, and D. Manocha, "Fast and approximate stream mining of quantiles and frequencies using graphics processors," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 611–622.

[30] K. Chen and L. Liu, "He-tree: a framework for detecting changes in clustering structure for categorical data streams," *The VLDB Journal*, vol. 18, no. 6, pp. 1241–1260, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00778-009-0134-5

[31] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1083784.1083789

[32] I. Raicu, I. T. Foster, Y. Zhao, P. Little, C. M. Moretti, A. Chaudhary, and D. Thain, "The quest for scalable support of data-intensive workloads in distributed systems," in *Proc. the 18th ACM International Symposium on High Performance Distributed Computing (HPDC'09)*, 2009.

[33] J. Dongarra, J. Bunch, C. Moler, and G. Stewart, *LINPACK Users Guide*, SIAM, 1979.

[34] S. P. E. Corporation. Spec benchmarks. [Online]. Available: http://www.spec.org/benchmarks.html

[35] C. Junghans, M. Karnstedt, and M. Gertz, "Quality-driven resource-adaptive data stream mining?" *SIGKDD Explor. Newsl.*, vol. 13, no. 1, pp. 72–82, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2031331.2031342

[36] S. Akioka, H. Yamana, and Y. Muraoka, "Data access pattern analysis on stream mining algorithms for cloud computation," in *Proc. the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2010)*, 2010.

[37] W. Wu and L. Gruenwald, "Research issues in mining multiple data streams," in *Proc. the First International Workshop on Novel Data Stream Pattern Mining Techniques (StreamKDD'10)*, 2010.

 ISBN: 978-1-61208-457-2