Multilevel Pattern Mining Architecture for Automatic Network Monitoring in Heterogeneous Wireless Communication Networks

Zhiguo Qu [1,2], John Keeney[3], Sebastian Robitzsch[2], Faisal Zaman[2], Xiaojun Wang[2]

[1] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2] School of Electronic Engineering, Dublin City University, Dublin, Ireland
[3] Network Management Lab, Athlone, Ireland

Abstract: The rapid development of network technology and its evolution toward heterogeneous networks has increased the demand to support automatic monitoring and the management of heterogeneous wireless communication networks. This paper presents a multilevel pattern mining architecture to support automatic network management by discovering interesting patterns from telecom network monitoring data. This architecture leverages and combines existing frequent itemset discovery over data streams, association rule deduction, frequent sequential pattern mining, and frequent temporal pattern mining techniques while also making use of distributed processing platforms to achieve high-volume throughput.

Keywords: automatic network monitoring; sequential pattern mining; episode discovery module

I. INTRODUCTION

Recent advances in telecommunication network technologies and their evolution [1], [2], [3] to heterogeneous networks have intensified the need for more efficient network monitoring and management capabilities in providing high-performance communications services.

Considering the complex environment of heterogeneous networks, at least two main issues need to be considered: 1) considerable amounts of data with high data dimensionality; 2) large number of protocols or use cases with complex contexts. To handle high volume and dimensionality of data, potential algorithms must achieve high time and memory efficiencies to fit off-the-shelf hardware configurations. On one hand, enumerating and modeling all possible contexts and using cases in heterogeneous networks are next to impossible. Conversely, the drive towards automatic network monitoring aims to greatly reduce or remove the need for human input in the processing part of network monitoring. To operate in an automatic manner and without expert domain knowledge input, general-purpose algorithms remain a requirement [4], [5] for discovering potentially interesting patterns and contexts. The most appropriate techniques lie in the domains of frequent itemset discovery [7], [8], [9], [10], association rule deduction [14], sequential pattern mining [15], [16], [17], [18], and temporal pattern mining [19],

[20], [21], [22], [23].

When dealing with large volumes of data with high dimensionality, providing administrators with timely response is necessary. Thus, real-time processing is the primary challenge for this work on pattern mining. To achieve this goal, this work proposes to mine multilevel patterns according to different time efficiencies by incorporating frequent itemset mining, frequent sequential pattern mining, and frequent temporal pattern mining. In simple terms, frequent itemsets are frequent combinations of unique items supported by transactions [7]; moreover, frequent sequential patterns not only comprises all the items but also comprises those in the correct sequence [17]. Furthermore, frequent temporal patterns extend frequent sequential patterns by also considering the time between elements and the sequence of elements [22]. Of these mining tasks, frequent itemset mining requires the least time, memory, and computing resources and is often sufficient for applications with relaxed pattern accuracy requirements or strict time requirements. In addition, frequent itemset mining can often be used as a precursor to other mining tasks while supporting faster analysis and summarization for some use cases and contexts. Frequent sequential pattern mining has high time and memory requirements but achieves better pattern accuracy, thus supporting applications with high accuracy requirements but with less constrained response times. With high time and memory requirements, frequent temporal pattern mining can discover temporal patterns with greater accuracy. For each pattern discovery approach, parallel processing mechanisms (e.g., Hadoop [24] and Spark [25]) can be deployed to handle high volumes of data.

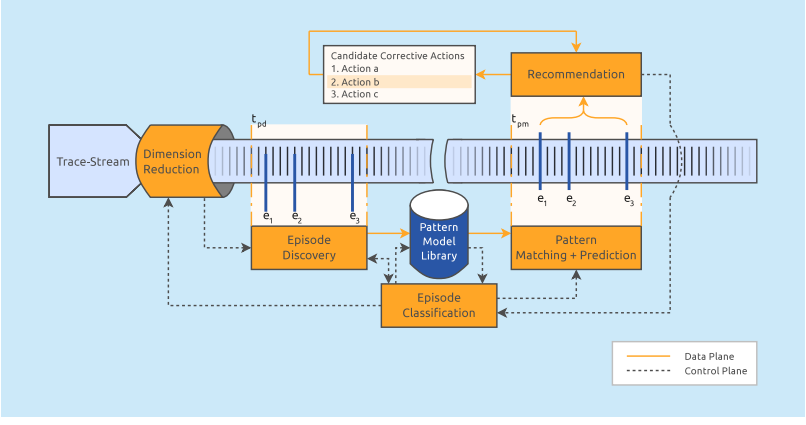

Fig.1 *Overall system architecture*

This paper presents an architecture to support the combination of tasks required to discover interesting patterns in high-volume high-dimensionality data. This architecture can efficiently achieve pattern mining for heterogeneous network monitoring data and discover potentially interesting patterns by leveraging and combining existing frequent itemset discovery over data stream, association rule deduction, frequent sequential pattern mining, and frequent temporal pattern mining techniques within distributed processing platforms.

II. Context of the Architecture

The presented novel multi-level pattern mining architecture forms part of a larger automatic network management system [6]. To achieve this goal, a pattern discovery work package strives to extract sequential event patterns from network monitoring data, especially in the context of ever-increasing management data volumes, which is the subject of this paper.

2.1 Overall architecture

The overall project architecture is shown in **Fig. 1**. The architecture is made up of five main modules: a dimension reduction module (DRM), an episode discovery module (EDM), an episode classification module (ECM), a pattern matching and prediction module (PMM), and a recommender system module (RSM).

1. The DRM [24] greatly reduces the dimensionality of the data and filters out most noisy events in the data stream to reduce the computation complexity of EDM and PMM.

2. The EDM discovers and identifies episodes or sequential patterns from the dimension-reduced event stream and then exports sequential pattern models into the Pattern

Model Library (PML), thus updating the set of discovered sequential patterns for the via the ECM.

3. The ECM models and classifies the discovered sequential patterns. Feedback from the ECM can help the DRM and the EDM optimize their parameter settings (such as transaction length and window size) to improve performance. The identified pattern models are output to the PML for the PMM.

4. The PMM scans the trace stream and matches the partial pattern models (from PML) to predict interesting patterns in the data just before they occur. These predictions will be exported to the RSM.

5. In response to PMM matches of pattern prediction rules, the RSM (retrieves and ranks suggestions to avert, alleviate, or mitigate against the effect of predicted issues on the basis of historic actions, experience of network administrators, or best practice.

2.2 Multilevel pattern structure

As mentioned, we classify patterns, and associated techniques into discover patterns, into three successive types: frequent itemsets, frequent sequential patterns, and frequent temporal patterns. This classification also establishes three different levels of patterns where mining tasks can be cascaded, as shown in **Fig. 2**.

2.3 Architecture of EDM

The task of the EDM is to constantly mine frequent episodes (pattern candidates) from dimension-reduced data streams. To facilitate distributed processing and to target and minimize time and memory consumption when mining frequent sequential patterns, the EDM adopts a three-level pattern structure. Frequent closed itemsets are first discovered, then frequent closed sequential episodes are mined based on the frequent closed itemsets, and finally, frequent closed temporal episodes can be selectively mined. This three-level pattern mining approach aims to reduce the discovery of unnecessary frequent patterns thereby minimizing the overhead. Notably, outputting three different types of patterns with different accuracy/overhead trade-offs is flexible and beneficial.

The EDM is made up of five sub-modules: the Data Analyzer, Mining Controller, Data Miner, Combiner, and the Episode Analyzer. These five sub-modules cooperate with each other to accomplish the task of the EDM by fulfilling their functions in turn. A flowchart of the EDM is presented in **Fig. 3**.

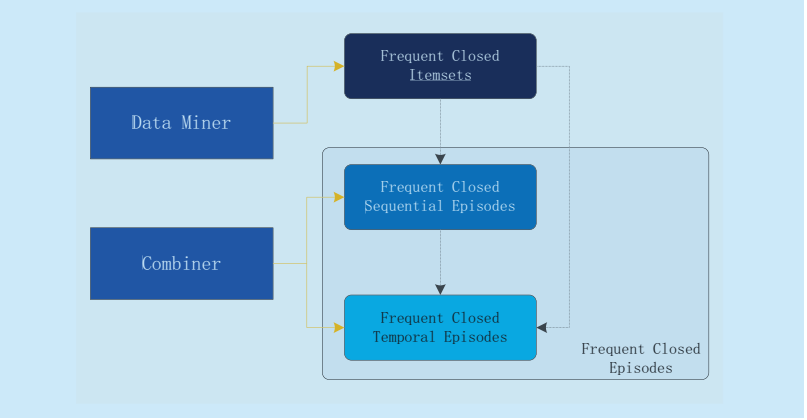

Fig.2 *Multilevel pattern structure*

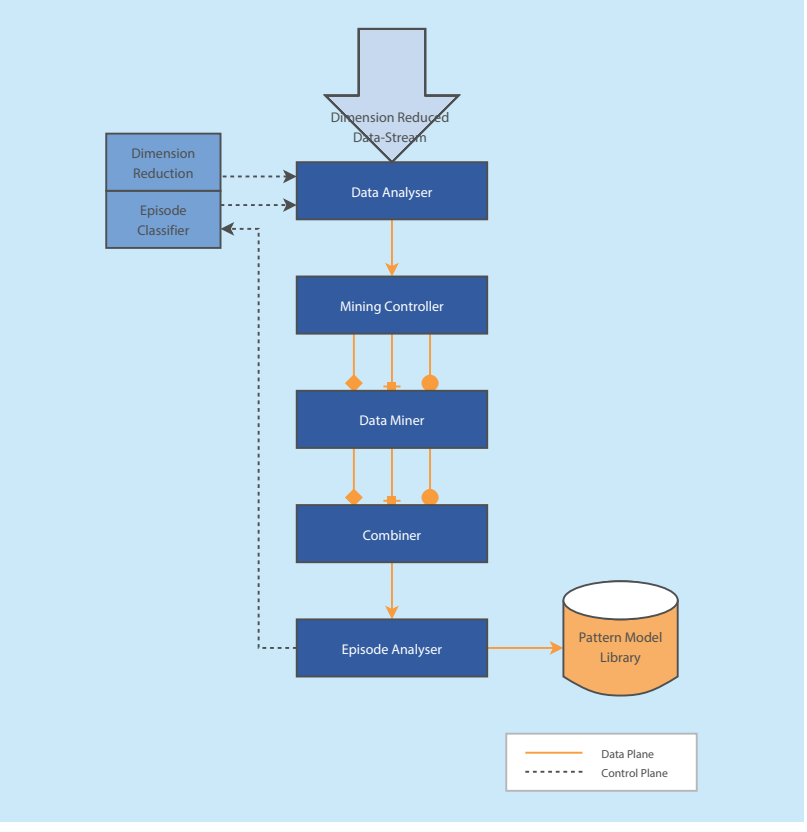

Fig.3 *Flowchart of the EDM module*

1. Data Analyzer: It determines appropriate configuration parameters for the data mining tasks (e.g., *Min-Sup*, window size, transaction length) and also determines if and how the stream could be split into sub-streams for parallel processing. For episode discovery in the EDM, automatic determination of algorithm settings is one of the significant challenges.

2. Mining Controller: After obtaining configuration information from the Data Analyzer, the Mining Controller splits the dimension-reduced trace stream into sub-streams or sub-groups and then assigns computation resources according to the configuration settings, such as I/O, memory and CPU, to each sub-stream or sub-group of the data. The Mining Controller then recursively calls the Data Miner to process the datasets split into different sliding window views over sub-streams to discover frequent episodes based on parameter settings provided by the Data Analyzer.

3. Data Miner: As the core sub-module of the EDM, the Data Miner continuously discovers frequent closed itemsets from each sliding window view according to its parameter settings. In this sub-module, the NewMoment [12] algorithm implementation can be selected to handle this task.

4. Combiner: After discovering new frequent closed itemsets from different sliding windows, the Combiner combines these frequent closed itemsets to form an overall set of frequent closed itemsets with a uniform format for each sub-stream. Then, on the basis of these discovered frequent closed itemsets and their corresponding projected databases, both PrefixSpan [17] and HTPM [22] can be selected to find the frequent closed sequential episodes or frequent closed temporal episodes for each sub-stream.

5. Episode Analyzer: This sub-module adopts an association rule mining approach to analyze discovered frequent episodes to filter out most episodes that are not relevant for the given use case. The identified episodes will be outputted and stored into the PML as pattern models.

III. EDM

1) Data Analyzer: As introduced, the Data Analyzer automatically determines a proper setting of parameters for the EDM module using statistics and feedback from DRM and ECM. Specifically, the Data Analyzer analyzes the dimension-reduced data by collecting and calculating essential characteristics of the data, such as average frequency of each event and the distribution of events. On the basis of the analysis, different data sub-streams of dimension-reduced for different contexts or use cases will be separated, and in some cases, data sub-streams shall be further divided into subgroups for parallel processing. Correspondingly, decisions about parameters for these data sub-streams will be automatically and accurately estimated and output after integrating feedback from DRM and ECM. Different sub-streams may have different parameter settings, but different time sliding windows within the same sub-stream or sub-group share the same parameter setting. Splitting the data stream into different sub-streams is mainly based on the following considerations: First, different use cases usually have different episodes with different contexts, and the episodes that cross several use cases are usually meaningless. Thus, reorganizing the data according to different use cases or contexts before the process of episode mining is better. Second, different sub-streams of the data that originate from different use cases generally have different data characteristics/features; thus, different optimized algorithm configurations may exist for mining episodes from different sub-streams. For example, consider two distinct telecom use cases, one where patterns related to individual mobile users are sought and another that focuses on patterns related to individual fixed cells in a mobile network. Clearly, when separating the data stream into individual sub-streams, a sensible approach is to partition the data in different ways, one where the data is partitioned by the user regardless of which cell they use and the other where the data are partitioned by cell regardless of users in the cell.

The Data Analyzer comprises seven components: Statistic Information Collector, Feedback Parser, Use Case Parser, Statistic Parser, DRM Parser, ECM Parser, and Parameter Setting Controller. A flowchart of the Data Analyzer is shown in **Fig. 4**.

2) Mining Controller: After receiving instructions from the Data Analyzer, the Mining Controller allocates computation resources to carry out the work specified by the Data Analyzer. Specifically, after obtaining the parameter settings from the Data Analyzer, the Mining Controller starts to split the reduced data into sub-streams (if necessary, the sub-streams could be further divided into sub-groups in the Mining Controller) and then assigns computation resources, such as I/O, memory, and CPU, for these sub-streams or even sub-groups according to their corresponding parameter settings. Afterwards, the Mining Controller will recursively call the Data Miner to process the data sub-streams or sub-groups to discover episodes according to the parameters set by the Data Analyzer.

The Mining Controller consists of three components, namely, Stream Splitter, Configuration Controller, and Source Controller. A flowchart of the Mining Controller is shown in **Fig. 5**.

Data Miner consists of four components, namely, Event Bit Representation, Itemset Bit Representation, Frequent Closed Itemset Discovery, and Occurrence Recorder. A flowchart of the Data Miner is shown in **Fig. 6**.

4) Combiner: As mentioned above, the frequent episode mining process consists of two phases. The first phase is to discover frequent closed itemsets, and it is handled by the Data Miner. The second phase is to mine frequent closed episodes from the frequent closed itemsets found in the first phase, and it is handled by the Combiner.

Combiner has three components, namely, Frequent Closed Itemset Combiner, the Projected Database Builder, and the Frequent Closed Episode Miner. A flowchart of the Combiner is shown in **Fig. 7**.

Specifically, after receiving FCITs and bit vectors of events and itemsets of each sub-stream and sub-group, the Combiner first combines the FCITs of sub-groups into an overall FCIT for each sub-stream. For this specific purpose, the algorithm [26] presents a set of FCIT combination policies and suitable approaches. Then, for each frequent closed itemset, a corresponding projected-database is built by filtering out all sub-sequences/transactions

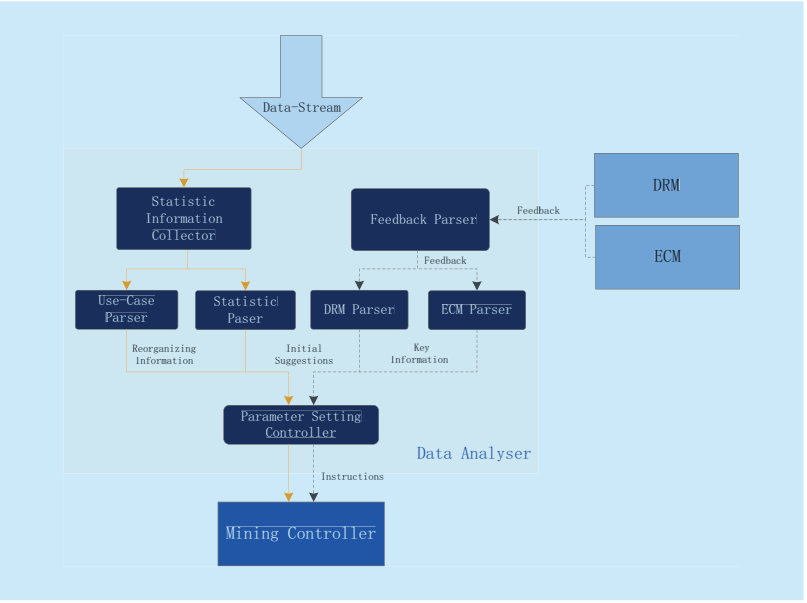

Fig.4 *Flowchart of Data Analyzer*

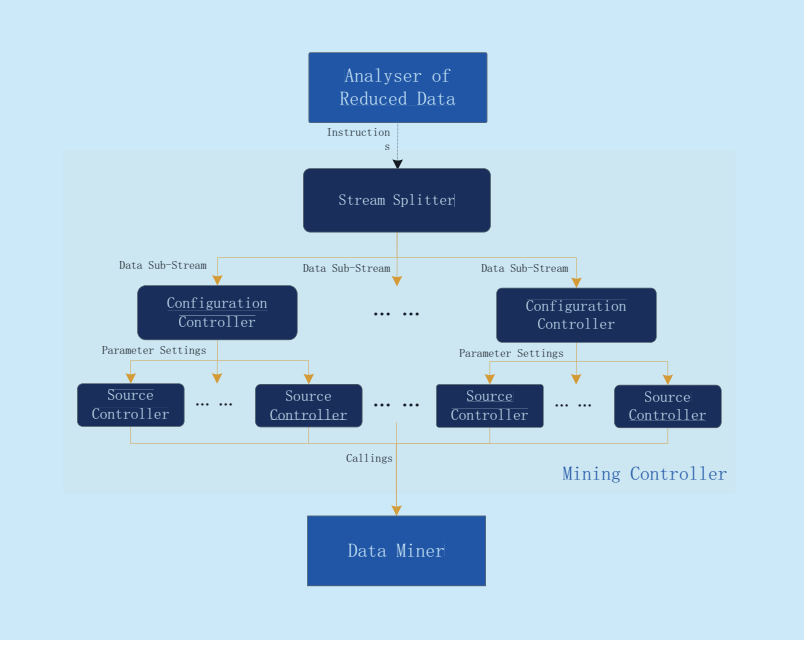

Fig.5 *Flowchart of Mining Controller*

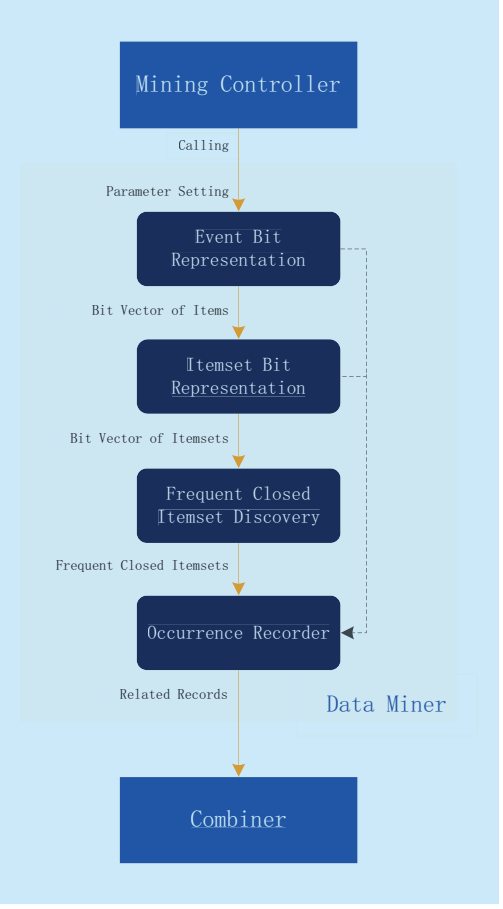

Fig.6 *Flowchart of Data Miner*

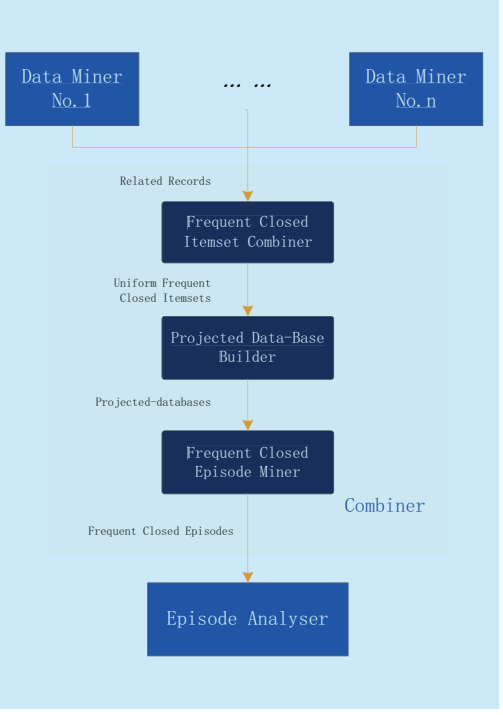

Fig.7 *Flowchart of Combiner*

in which the particular itemset is not contained and the events with the types other than those present in the particular itemset. Finally, the Combiner recursively invokes a frequent sequential or temporal mining algorithm, such as PrefixSpan [17] or HTPM [22], to discover frequent closed episodes in the projected database for each frequent closed itemset. Only frequent closed episodes that consist of all the different event types in the corresponding itemset and have sufficiently large support values in the corresponding projected databases will be outputted.

5) Episode Analyzer: The Episode Analyzer aims to deduce association rules from the discovered episodes. Most of the meaningless episodes will be filtered out in this sub-module. Specifically, after receiving frequent closed episodes from the Combiner, the *Confidence* and *Lift* of each episode are calculated in terms of the support values of the antecedent and consequent parts of the episode sequence. (*Confidence* denotes the conditional probability of the consequent actually occurring after the antecedent events occurred, which indicates whether useful association rules can be derived from the discovered patterns. *Lift* represents the degree to which one event or event sequence occurrence predicts another event or event sequence occurrence.) Then, discovered episodes are identified according to the thresholds of *Confidence* and *Lift*, and the episodes with values less than the thresholds will be filtered out; the default thresholds are 60% for *Confidence* and 100% for *Lift*. The frequent closed episodes with sufficient *Confidence* and *Lift* values are exported into the PML, while a notification will be sent to the ECM for further processing. The Episode Analyzer consists of two components, namely, the Confidence Calculator and the Lift Calculator. A flowchart of the Episode Analyzer is shown in **Fig. 8**.

IV. Experimental Evaluations

To prove the feasibility of the EDM module, three classic algorithms, namely, NewMoment, PrefixSpan, and HTPM, are implement-

ed in Java. All the experiments are performed on a 2.5 GHz Intel Core i5-3210M PC with 8 GB of main memory and running Microsoft Windows 7. Each approach was evaluated using a simulated dataset called a*PG_Sim*, which is generated by the open-source OpenMSC real-time emulator [27]. On the basis of a set of configurable parameters and predefined message sequences, correlations, and temporal pattern parameters, OpenMSC generates realistic real-time pattern traces obfuscated with a large corpus of random noise events. The *PG_Sim* dataset consists of a total of 100K events, including 500 + 12 unique types of events, incorporating 500 noise event types {100,.....,599} and 12 information event types. Three candidate sequence patterns are manually pre-defined for inclusion.

The performance of the NewMoment, PrefixSpan, and HTPM algorithms is tested by running them on the test dataset *PG_Sim*. Results show that all the predefined patterns are found by the NewMoment, PrefixSpan, and HTPM algorithms with different pattern forms. The time taken for the algorithms to complete their search given different *Min-Sup* values is shown in **Figs. 9** and **10**. The *Min-Sup* value is incrementally increased from 0.35 (35%) to 0.8 (80%), while transaction length is fixed at 20.

Figs. 9 and **10** clearly show significant differences in the time required for NewMoment, PrefixSpan, and HTPM to complete. The time requirements for HTPM are much larger than that of PrefixSpan, while PrefixSpan is slower than NewMoment. With the decreasing *Min-Sup*, this difference increases substantially. **Fig. 9** also shows that for relatively simple streams with a small number of event types and patterns, where streams can be adequately split into smaller transaction sets (in this case, 100K events), NewMoment can process the stream online in a near-real-time manner.

V. Conclusions

This paper outlined the challenges that need to be solved to perform pattern mining in a continuous data stream that contains complex context with unpredictable unbounded volumes. A multi-level pattern mining solution

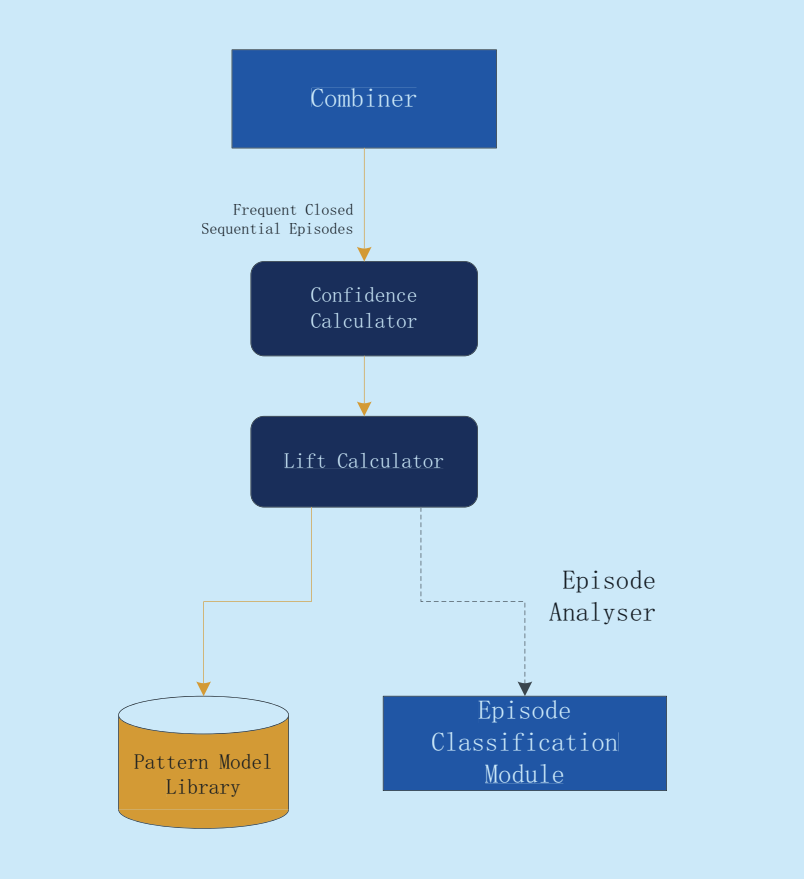

Fig.8 *Flowchart of Episode Analyzer*

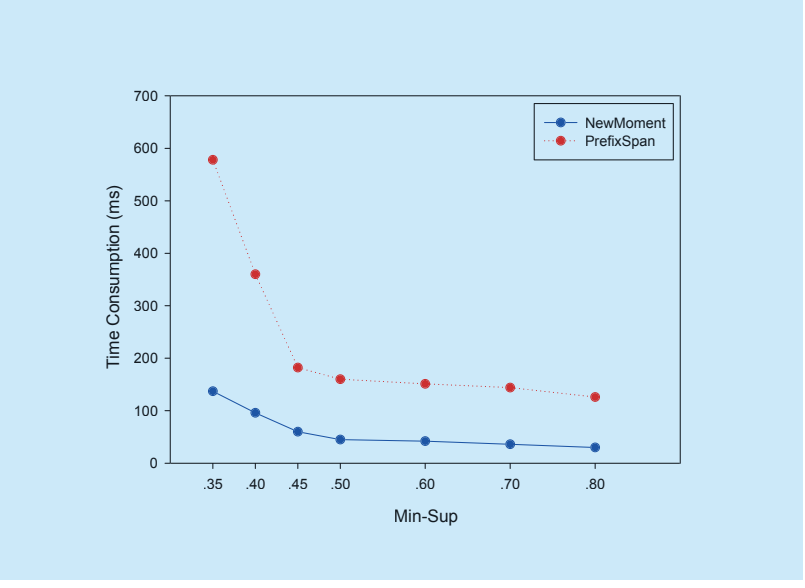

Fig.9 *Time consumption comparison between NewMoment and PrefixSpan (Min-Sup)*

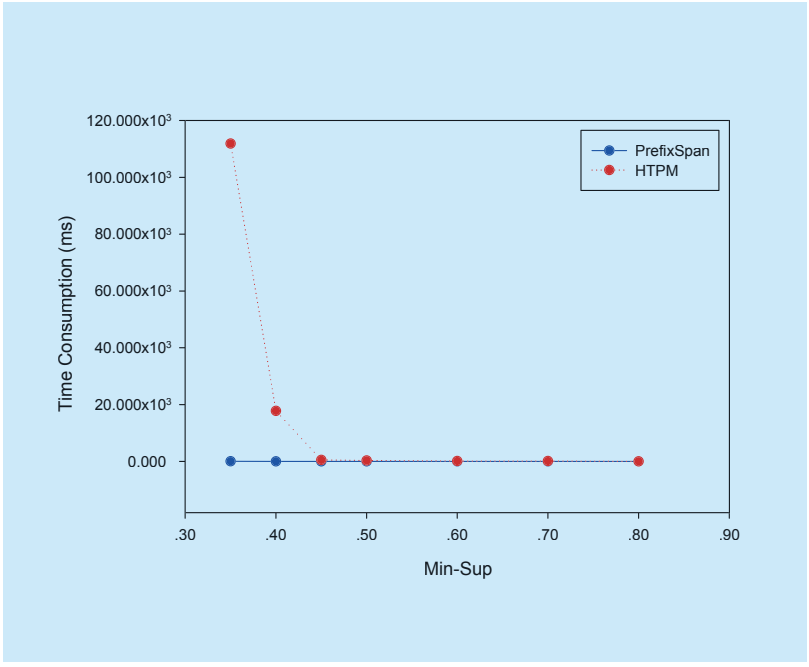

Fig.10 *Time consumption comparison between PrefixSpan and HTPM (Min-Sup)*

and architecture was presented to support automatic network management by discovering interesting patterns from Telecom Network Monitoring data. The architecture enables discovery or interesting patterns with data from different domains related to Telecom Network Monitoring. The architecture is composed of five sub-modules: Data Analyzer, Mining Controller, Data Miner, Combiner, and Episode Analyzer. Each sub-module is described in detail along with implementation approaches for each function. These five sub-modules cooperate to form the EDM by fulfilling their functions in turn. The approach leverages and combines frequent itemset discovery over data streams, association rule learning, frequent sequential pattern mining, and frequent temporal pattern mining techniques while also making use of distributed processing platforms to achieve high-volume throughput. To test the functionality of the model, three classic pattern mining algorithms, namely, NewMoment, PrefixSpan, and HTPM, are implemented. Their performances are compared in terms of different parameter settings by executing the algorithms with a simulated streaming dataset.

Acknowledgments

This work was funded by the Enterprise Ireland Innovation Partnership Programme with Ericsson under grant agreement IP/2011/0135 [6] and supported by the National Natural Science Foundation of China (No. 61373131, 61303039, 61232016, 61501247), and the PAPD and CICAEET funds.

References

[1] Z. J. Fu, X. M. Sun, Q. Liu, L. Zhou, and J. G. Shu, "Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing," IEICE Transactions on Communications, vol. E98-B, no. 1, pp.190-200, 2015.

[2] S. D. Xie, Y. X. Wang, "Construction of Tree Network with Limited Delivery Latency in Homogeneous Wireless Sensor Networks," Wireless Personal Communications, 2014, 78(1): 231-246.

[3] P. Guo, J. Wang, B. Li, S. Y. Lee, "A Variable Threshold-value Authentication Architecture for Wireless Mesh Networks," Journal of Internet Technology, 2014, 15(6): 929-936.

[4] T. H. Ma, J. J. Zhou, M. L. Tang, Y. Tian, A. AL-DHELAAN, M. AL-RODHAAN, and S. Y. LEE, "Social network and tag sources based augmenting collaborative recommender system," IEICE transactions on Information and Systems, vol. E98-D, no.4, pp. 902-910, Apr. 2015.

[5] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental Support Vector Learning for Ordinal Regression," IEEE Transactions on Neural Networks and Learning Systems, 2014, 26(7): 1403-1416.

[6] Dublin City University and Ericsson, E-Stream Project, 2014. http:// www.estream-project.com

[7] G. S. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams," In: Proceedings of the 28th International Conference on Very Large Databases (VLDB Endowment 2002), pp. 346-57, 2002.

[8] J. H. Chang and W. S. Lee, "Finding Recent Frequent Itemsets Adaptively over Online Data Streams," In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 2003), pp. 487-492, 2003.

[9] H. F. Li and H. Chen, "Mining Non-Derivable Frequent Item Sets over Data Stream," Data & Knowledge Engineering, vol. 68, no. 5, May 2009, pp. 481-498.

[10] S. K. Tanbeer, C. F. Ahmed, B. S. Jeong and Y. K. Lee, "Efficient Single-Pass Frequent Pattern Mining Using a Prefix-tree," Information Sciences, 2008, 179(5): 559-583.

[11] Y. Chi, H. Wang , P. S. Yu and R. R. Muntz, "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window," In: Proceedings of Data Mining (ICDM '04), pp. 59-66, 2004.
[12] H. F. Li, C. C. Ho and S. Y. Lee, "Incremental Updates of Closed Frequent Itemsets over Continuous Data Streams," Expert Systems with Applications, 2009, 36(2): 2451-2458.
[13] P. T. La, B. Le and B. Vo, "Incrementally Building Frequent Closed Itemset Lattice with Different Models," Expert Systems with Applications, 2014, 41: 2703-2712.
[14] S. Brin, R. Motwani and C. Silverstein, Beyond market baskets: generalizing association rules to correlations," In: Proceedings of the 1997 ACM SIGMOD international conference on Management of data (SIGMOD 1997), New York, NY, USA, pp. 265-276, 1997.
[15] R. Srikant, R. Agrawal, Mining sequential patterns: generalizations and performance improvements, In: Proceedings of the 5th International Conference on Extending Database Technology, Avignon, France, pp. 3-17, 1996.
[16] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen,U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining, In: Proceedings of 2000 International Conference on Knowledge Discovery and Data Mining (KDD00), Boston, MA, pp. 355-359, 2000.
[17] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M.C. Hsu, PrefixSpan: mining sequential patterns efficiently by prefix projected pattern growth, In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, pp. 215-226, 2001.
[18] M. J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning, 2001, 42(1): 3160-3165.
[19] S. Y. Wu and Y. L. Chen, Mining non-ambiguous temporal patterns for interval-based events, IEEE Transactions on Knowledge and Data Engineering, 2007, 19(6): 742-758.
[20] P. S. Kam and A. W. C. Fu, Discovering temporal patterns for interval-based events, In: Proceedings of Second International Conference on Data Warehousing and Knowledge Discovery, Springer, London, UK, pp. 317-326, 2000.
[21] S. de Amo, W. P. Junior and A. Giacometti, MILPRIT*: a constraint-based algorithm for mining temporal relational patterns, International Journal of Data Warehousing and Mining, 2008, 4(4): 42-61.
[22] S. Y. Wu and Y. L. Chen, Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events, Data and Knowledge Engineering, 2009, 68(11): 1309-1330.
[23] Y. L. Chen, S. Y. Wu and Y. C. Wang, etc, Discovering multi-label temporal patterns in sequence database, Information Sciences, 2011, 181(3): 398-418.
[24] Hadoop Map/Reduce tutorial, http://hadoop.apache.org/docs/r1.2.1/mapred tutorial.html
[25] M. Zaharia and M. Chowdhury, Spark: cluster computing with working sets in cloud computing, In: Proceeding of HotCloud 2010, (2010).
[26] C. Lucchese, S. Orlando, R. Perego, Fast and memory efficient mining of frequent closed itemsets, IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 21-36.
[27] S. Robitzsch, OpenMSC - An Open Source MSCgen-Based Control Plane Trace Emulator for Communication Networks, 2014. http://openmsc.blogspot.com

Biographies

Zhiguo Qu, received his PhD in Information Security from Beijing University of Posts and Telecommunications, China, in 2011. From 2012 to 2014, he worked as a post-doc researcher fellow at Dublin City University in Ireland. In July 2011, he joined Nanjing University of Information and Technology in China, where he is currently a lecturer in the College of Computer and Software. His research interests include quantum secure communication, quantum information hiding, data mining, and digital watermarking. E-mail: qzghhh@126.com