A multiple pattern complex event detection scheme based on decomposition and merge sharing for massive event streams

Wang, Jianhua; Ji, Bang; Lin, Feng; Lu, Shilei; Lan, Yubin; Cheng, Lianglun

2020

Wang, J., Ji, B., Lin, F., Lu, S., Lan, Y. & Cheng, L. (2020). A multiple pattern complex event detection scheme based on decomposition and merge sharing for massive event streams. International Journal of Distributed Sensor Networks, 16(10). https://dx.doi.org/10.1177/1550147720961336

https://hdl.handle.net/10356/146856

https://doi.org/10.1177/1550147720961336

Research Article

International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2020, Vol. 16(10)
© The Author(s) 2020
DOI: 10.1177/1550147720961336
journals.sagepub.com/home/dsn
SAGE

A multiple pattern complex event detection scheme based on decomposition and merge sharing for massive event streams

Jianhua Wang[1,2,3*], Bang Ji[1*], Feng Lin[3], Shilei Lu[1], Yubin Lan[1] and Lianglun Cheng[4]

Abstract
Quickly detecting related primitive events for multiple complex events from massive event stream usually faces with a great challenge due to their single pattern characteristic of the existing complex event detection methods. Aiming to solve the problem, a multiple pattern complex event detection scheme based on decomposition and merge sharing is proposed in this article. The achievement of this article lies that we successfully use decomposition and merge sharing technology to realize the high-efficient detection for multiple complex events from massive event streams. Specially, in our scheme, we first use decomposition sharing technology to decompose pattern expressions into multiple subexpressions, which can provide many sharing opportunities for subexpressions. We then use merge sharing technology to construct a multiple pattern complex events by merging sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results. As a result, our proposed detection method in this article can effectively solve the above problem. The experimental results show that the proposed detection method in this article outperforms some general detection methods in detection model and detection algorithm in multiple pattern complex event detection as a whole.

Keywords
Complex event detection, multiple pattern, decomposition sharing, merge sharing, massive event streams

Date received: 12 March 2020; accepted: 4 September 2020

Handling Editor: Yanjiao Chen

Introduction

Internet of Manufacturing Things (IOMT)[1] is an important technology to enhance perception, control and management ability of manufacturing and service process, promote enterprise's production and management innovation, and drive manufacturing transformation and upgrading. In the realistic environment of IOMT, a great number of sensing equipment—such as radio-frequency identification (RFID) tags and sensor nodes—are deployed in the manufacturing fields to monitor various manufacturing objects, such as people, materials, equipment, production process, products

[1]College of Electronic Engineering, South China Agricultural University, Guangzhou, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3]National Center for International Collaboration Research on Precision Agricultural Aviation, Guangzhou, China
[4]College of Automation, Guangdong University of Technology, Guangzhou, China
[*]These authors contributed equally to this work and should be considered co-first authors.

Corresponding author:
Jianhua Wang, College of Electronic Engineering, South China Agricultural University, No. 483, Wushan Load, Guangzhou 510642, China.
Email: wangjianhua655@163.com

and environmental changes due to its increasingly large manufacturing scale, more and more complex manufacturing process, temporal and spatial distribution of production process, and multi-source interference of manufacturing environment.[2] These sensing devices generate massive manufacturing event streams.

Since the generated massive manufacturing event streams have the following event characteristics: large volume, high velocity, many varieties, and small value, that make it difficult to quickly pick up some valuable information from it, thus influencing its extensive applications.[3,4] Therefore, how to quickly pick up some valuable information from massive manufacturing event streams has become a very important problem. Since CED (Complex Event Detection)[5] technology can quickly pick up some valuable information from massive event stream using the association between event attributes, detection rules, and algebraic operations; therefore, it has become a research hotspot recently. So how to use CED technology to rapidly detect some related information from massive event stream has become an important issue in this article.

At present, many CED methods have been developed to detect a complex event from event stream. For example, a Petri net–based CED method,[6] a diagram-based CED method,[7] a tree-based CED method,[8] and a nondeterministic finite automaton (NFA)–based CED method,[9] and some of their improved detection methods—for example, a CED method based on timed Petri net,[10] a CED method based on optimized directed graph,[6] a CED method based on compressed composition tree,[11] and a CED method based on pushdown automata[12]—have been proposed to detect related complex events from event streams. However, all of these detection methods above are based on single pattern, and they can only detect a single complex event at one time and cannot realize the simultaneous sharing detection for multiple complex events at the same time due to its unsharing characteristics. Detecting multiple complex events usually needs to construct multiple different detection models, thus leading to long detection time, high memory consumption, and low event throughput.

However, in real environment of IOMT, a large number of pattern expressions of complex event are registered in detection system to detect various production activities of manufacturing, and we need to rapidly detect multiple different complex events from massive even streams at the same time because its high responsiveness of detection result in IOMT. Under these circumstances, if we still use the above existing CED methods to detect multiple complex events from massive manufacturing event streams, there will be a long detection time, high memory consumption, and low detection efficiency due to its single pattern detection characteristic.

In order to address the above problem, some multiple pattern CED methods have proposed to simultaneously detect multiple different complex events. For example, a multiple pattern CED method based on hierarchical pattern sharing,[13] a multiple pattern CED method based on time event correlation sharing,[14] a multiple pattern CED method based on pattern sharing and pattern reordering,[15] and a multiple pattern CED method based on multi-core processor sharing[16] have been presented to detect multiple pattern complex events. Although these schemes above are well designed, but these methods do not completely explore the sharing relationships existed in pattern expressions and consider the sharing detection among these multiple pattern expressions. Therefore, there is still a need to further develop more efficient multiple pattern CED method to improve their whole detection efficiency.

To solve the problem, a multiple pattern CED scheme based on decomposition and merge sharing is proposed in this article. The achievements of this article include the following:

1. Decomposition sharing technology is used to decompose pattern expressions into multiple subexpressions, which can provide many sharing opportunities for the subexpressions.
2. Merging sharing technology is used to construct a multiple pattern detection model of complex event by merging sharing all the same prefix, suffix, or subpattern into one based on the decomposition results of subexpressions above.
3. A series of experiments is performed to verify the effectiveness of our proposed method in this article. And the experimental results show that our proposed detection scheme has a great improvement in detection model and detection algorithm compared with some general detection methods in detecting multiple pattern complex events.

The rest of this article is organized as follows. In section "Related works," the related works of CED are introduced. Our proposed CED method is presented in section "Proposed scheme." The experimental results and analysis with our proposed scheme are shown in section "Experimental results and analysis." In section "Conclusion," we give our conclusions.

Related works

In recent, a lot of research works have been taken for the detection of complex events. There are four general CED methods, for example, Wang et al.[6] presented a CED method based on Petri net, Bai et al.[7] proposed a CED method based on graph, Sun et al.[8] suggested a

CED method based on tree, and Mei et al.[9] developed a CED method based on finite automaton, and some of their improved detection methods—for example, Jin et al.[10] proposed a CED method based on timed Petri net, Wang et al.[6] presented a CED method based on optimized directed graph, Li et al.[11] proposed a CED method based on condensed composition tree, and Cao et al.[12] proposed a CED method based on pushdown automata structure—have been developed to detect a complex event from various event streams. Besides some other detection methods—including probability graph methods, deep learning methods, representation learning methods, and so on—are applied to perform event detection. For example, Chen et al.[17] proposed a nonparametric model for online topic discovery with word embeddings, where they mainly exploited auxiliary word embeddings to infer the topic number and employed a "spike and slab" function to alleviate the sparsity problem of topic-word distributions in online short text analyses. Hu et al.[18] presented a transformation-gated long short-term memory (LSTM) to enhance the ability of capturing short-term mutation information, where the function of transformation gate, input gate information, and the value range of the partial derivative corresponding to the transformation gate were studied. Chen et al.[19] developed a Dirichlet process biterm-based mixture model for short text stream clustering in their study, where the topic drift problem and the sparsity problem can be dealt with in short text stream clustering. Hu and Zheng[20] designed a multistage attention network to capture the different influence for multivariate time series prediction, where they mainly exploited the influential attention mechanism, temporal attention mechanism of model, and the prediction performance on two different real-world multivariate time series data sets. Wang et al.[21] developed a trust-enhanced collaborative filtering for personalized point-of-interest (POI) recommendation, where the trust-enhanced user similarity in user-based collaborative filtering based on network representation learning is calculated, and these two factors into POI recommendation are integrated by a fusion model.

However, the above existing detection methods can only detect a single complex event from event streams once a time, and they cannot realize the sharing detection for multiple complex events at the same time due to its unsharing characteristic. Detecting multiple different complex events usually needs to construct multiple different CED models; therefore, they cause long detection time, high memory consumption, and low event throughput due to their unsharing of the same prefix, suffix, or subpattern among them, thus affecting their whole detection performance.

In order to solve the above problem, some detection schemes about multiple query or detection on complex event processing (CEP) have been proposed recently. Liu et al.[13] presented a multi-dimensional event sequence analysis based on hierarchical pattern sharing, namely, E-Cube for huge massive high-speed streams. In their method, sharing results of hierarchical pattern from one level query to another was used to reduce their redundant computation among pattern queries. Ray et al.[14] developed a multiple detection optimizer of CEP called SPASS for real-time event streams, where the time-based event correlations sharing among detection expressions was used to identify opportunities for the effective sharing detection. Zhang et al.[15] presented a multi-query optimizer of CEP called MOTTO for real-time event streams, where they used three sharing technologies to realize the sharing opportunities in complex query workloads of SAP ESP. In Kolchinsky and Schuster,[16] a novel real-time multi-pattern CEP based on pattern sharing and pattern reordering was proposed for event streams. In their method, they mainly used the combination of sharing and pattern reordering techniques to construct an optimal plan. In Suhothayan et al.,[22] multiple pattern CEP architectures based on multi-core processors sharing were developed for event processing. In their architecture, they mainly used multi-threading to construct a pipeline-based event inference model, and then improve the performance of CEP. In Mayer et al.,[23] a predictable low-latency event detection with parallel CEP was proposed for data streams, in which pattern-sensitive partitioning pattern is used to achieve a high degree of parallelism in detecting event patterns with a sequential manner and a low parallelization degree. Kalyvianaki et al.[24] proposed a query planner of CEP called SQPR to exploit sharing opportunity between queries and partial sharing results in massive event stream, in which the relationships between SEQ (Sequence) operators in different queries were studied and used. Ray et al.[25] proposed a multiple pattern query processing and optimization system based on temporal patterns sharing called SPASS system for massive event stream, in which shared temporal patterns were used to identify the opportunities for the effective sharing processing of CEP. In Liu et al.,[26] a high-performance nested CEP query processing based on rewriting rules and subexpressions sharing was proposed for event streams. In their method, they mainly used rewriting rules to flatten a nested complex event expression and used sharing subexpressions to share the same nested complex event. However, the method cannot process other temporal relationship between events. Ma et al.[27] proposed an efficient multiple pattern event processing method based on multiple pattern state transition, failure transition, and state output for high-speed train data streams. In their method, they mainly used multiple pattern state transition, failure transition, and state output to realize the detection of complex event, and then improve their throughput. Ma and Wang[28] developed a high-efficiency joint event inference model based on patterns sharing, failure transitions, and conditional output for real-time context awareness and decision-making in the Internet of things (IoT) edge systems. In their work, they mainly used the patterns sharing, failure transitions, and conditional output of the joint inference model to eliminate the redundancies of inter-model. Giannikis et al.[29] proposed a shared workload optimization for shared work systems, in which common subexpressions were searched and a global plan is calculated using the branch-and-bound method. Zervakis et al.[30] presented an efficient continuous multi-query processing based on shared query sets for graph streams, in which a novel algorithmic solution was presented for efficient multi-query evaluation against graph stream by leveraging the shared query sets. Schultz-Møller et al.[31] proposed a distributed CEP based on query rewriting for massive event stream, where a cost to every candidate plan was assigned and a search algorithm is utilized to select the lowest cost evaluation scheme. In Xiao et al.,[32] a new parallelization model and three parallel processing strategies were proposed to address the difficulties of implementing parallel processing for distributed CEP systems. In Yuan et al.,[33] a distributed query plan of CEP structure and algorithm based on directed acyclic graph was developed to solve the problem of single complex event or small quantity of events. In Xiao,[34] an intelligent CEP method with D numbers under fuzzy environment was proposed to address the issues of intrinsic uncertainty in pattern rules. In Abbasnejad et al.,[35] a novel CED based on joint max margin and semantic was presented to address the limitations of semantic and temporal features. In Zhang et al.,[36] a series of optimized algorithms based on nondeterministic automata models was used to reduce its bottlenecks and to realize the detection of complex event after analyzing its complexity of expensive queries of complex event. In Wang et al.,[37] a multi-pattern sharing method based on multi-pattern sharing technology was used to share their same prefix, suffix, or subpattern in all pattern expressions. However, this method cannot share subexpressions in its decomposable pattern expressions. In addition, Scanagatta et al.[38] presented approximate structure learning algorithms for Bayesian networks, where they improved on state-of-the-art methods that rely on an ordering-based search by sampling more effectively the space of the orders, including parent set identification, structure optimization, and structure optimization under bounded treewidth. Liu and Liu[39] proposed a maximum relevance minimum common redundancy (mRMCR) algorithm based on the information theory and feature selection, where they established a mutual information solution formula on the preference database and designed a formula for calculating mRMCR. Jiang et al.[40] developed a fast and effective method, called CatchSync, where they mainly exploited two of the tell-tale signs left in graphs by fraudsters: synchronized behavior and rare behavior.

However, these methods do not completely explore the sharing relations between multiple pattern expressions and consider the sharing detection among multiple pattern expressions. Therefore, there is still a need to further develop more efficient multiple pattern CED methods to improve its whole detection efficiency.

Different from these above methods, a multiple pattern CED method based on decomposition and merge sharing is proposed for massive event stream on the basis of the analyzing and studying single pattern detection methods in this article. In our proposed scheme, we mainly use decomposition sharing technology to decompose pattern expressions into multiple subexpressions, which can provide many sharing opportunities for the subexpressions and use merging sharing technology to construct a multiple pattern CED model based on the above decomposition results by merging sharing all the same prefix, suffix, or subpattern into one, thus improving the detection efficiency of multiple pattern complex events.

Proposed scheme

In this section, a multiple pattern complex detection scheme based on decomposition and merge sharing is proposed for massive event streams.

Motivation resource

During the constructing processing of single pattern detection models from pattern expressions of complex event, there are a large number of the same subexpressions in their pattern expressions. Those same subexpressions cannot realize the sharing detection with each other before decomposition. Detecting these complex events usually needs to independently construct multiple different detection models of complex event, which leads to many repetitive building, storing, searching, and computing operations for them, thus influencing its whole detection efficiency. However, the existing multiple pattern CED methods can only achieve the detection sharing for the same prefix, suffix, or subpattern among pattern expressions, and they cannot realize the sharing detection for subexpressions in decomposable pattern expressions. Figures 1 and 2 show the same subexpressions existed in single pattern detection models constructed by pattern expressions SEQ1(A,B,D) and SEQ2(G,A,B), SEQ3(A,B,C,D) and SEQ4(B,C,D,E), respectively.

Aiming to solve the above problem, in this article, a multiple pattern CED method based on decomposition and merge sharing is proposed. In our scheme, we first use decomposition sharing technology to decompose

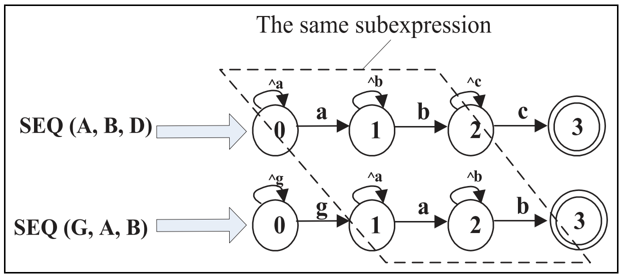

Figure 1. The subpattern in SEQ1(A,B,D) and SEQ2(G,A,B).

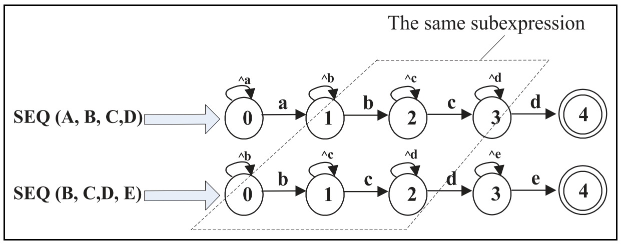

Figure 2. The subpattern in SEQ3(A,B,C,D) and SEQ4(B,C,D,E).

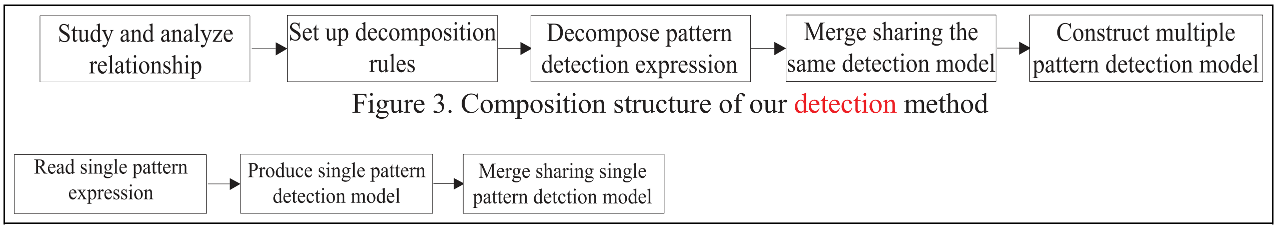
Study and analyze relationship
Set up decomposition rules
Decompose pattern detection expression
Merge sharing the same detection model
Construct multiple pattern detection model
Figure 3. Composition structure of our detection method
Read single pattern expression
Produce single pattern detection model
Merge sharing single pattern detction model

Figure 3. Composition structure of our detection method.

the pattern expressions into multiple subexpressions, which can provide many sharing opportunities for the subexpressions. We then use merging sharing technology to construct a multiple pattern complex event model by merging sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results. As a result, our proposed detection method can effectively solve the above problem. And the experimental results show that our proposed detection scheme has a great improvement in detection model and detection algorithm compared with some general detection methods in multiple pattern complex events.

Working principle

The working principle for our proposed detection scheme in this article is that, first, we study and analyze the relationship between single pattern detection expressions; second, we set up two important decomposition rules for the pattern expressions; third, we decompose the pattern expression using the above decomposition rules; fourth, we merge sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results; fifth, we construct a multiple pattern detection model using merging sharing technology; finally, we use the detection model to quickly detect multiple pattern complex events. As a result, our proposed detection model in this article can effectively solve the above problem, which can save many repetitive building, storing, searching, and calculating operations for them, thereby improving its overall detection performance.

Realizing processing

Figure 3 shows the realizing processing for our suggested detection method in this article, which includes the following five important processes: study and analyze relationship, set up decomposition rules, decompose pattern detection expression, merge sharing the same detection model, and construct multiple pattern detection model.

Study and analyze relationship. There are a large number of the same subexpressions during the constructing processing of single pattern detection models from pattern expressions of complex event. Those subexpressions are very different to realize the sharing detection with each other before decomposition due to its single pattern characteristic, which can lead to a lot of repetitive building, storing, searching, and computing operations for them, thus influencing its whole detection efficiency.

In this article, first, in order to realize the sharing detection for the subexpressions in pattern expressions, we use decomposition technology to decompose the pattern expressions into multiple subexpressions; second, in order to effectively reduce the repetitive building, storing, searching, and computing operations for them, we use merge sharing technology to construct a multiple pattern complex event detection model by merging sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results, thus improving its whole detection efficiency.

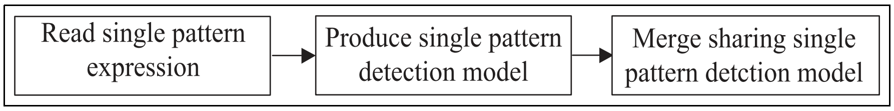
Read single pattern expression
Produce single pattern detection model
Merge sharing single pattern detction model

Figure 4. Realizing processing for our merging sharing.

Set up decomposition rules. Since a pattern expression of complex event can be decomposed into multiple subexpressions without changing its semantic meaning. Based on the principle, we set up two important decomposition rules for a pattern expression and realize the decomposition for a pattern expression. The two built decomposition rules in this article are shown as follows:

1. Decomposition rule 1

Left decomposition scheme: a pattern expression itself can be sequentially break into the prefix two operands and the rest expression in such way, and the suffix two operands are referred as a subexpression.

2. Decomposition rule 2

Right decomposition scheme: a pattern expression itself can be sequentially break into the suffix two operands and the rest expression in such way, and the suffix two operands are referred as a subexpression.

In this article, we denote the decomposed scheme with the form of "subexpression"→ "the rest part of original query," where → means connecting the output of left operation to the input of right operation.

Decompose pattern detection expression. Based on the above decomposition rules, we can easily decompose a pattern expression at its arbitrary places. For example, the decomposition results for decomposing the pattern expression SEQ1(A,B,D) and SEQ2(G,A,B), SEQ3 (A,B,C,D) and SEQ4(B,C,D,E) using above decomposition rules are shown as follows, respectively. And there will be many the same prefix, suffix, or subpattern among subexpressions after decomposition process:

SEQ1(A,B,D): SEQ(A,B) → SEQ1({A,B},D);
SEQ2(G,A,B): SEQ(A,B) → SEQ2(D,{A,B});
SEQ3(A,B,C,D): SEQ(B,C,D) → SEQ3(A,{B,C,D});
SEQ4(B,C,D,E): SEQ(B,C,D) → SEQ4(A,{B,C,D}).

Merge sharing the same detection model. Since a large number of the same prefixes, suffixes, or subpatterns can be obtained from the pattern expressions after completing the above decomposing process. For example, SEQ1(A,B,D) and SEQ2(G,A,B) can produce the same sharing subexpression SEQ(A,B) in them; SEQ3(A,B,C,D) and SEQ4(B,C,D,E) can produce the same sharing subexpression SEQ (B,C,D) in them. Therefore, in order to effectively reduce the repetitive building, storing, searching and computing operations caused by them, we use merge sharing technology to share the same prefix, suffix, or subpattern based on the above decomposition results, and then reduce their redundancy.

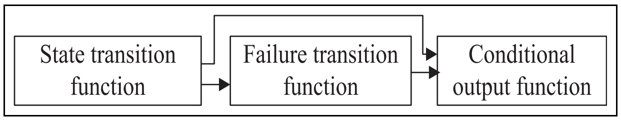
State transition function
Failure transition function
Conditional output function

Figure 5. The relationship among three functions.

Figure 4 shows the realizing processing for our merging sharing in our scheme. It includes three important realizing processes as follows: read single pattern expression, produce single pattern detection model, and merge sharing single pattern detection model.

1. Read single pattern expression

Read single pattern detection expression mainly executes the reading operation for each single pattern expression of complex event from system.

2. Produce single pattern detection model

Produce single pattern detection model mainly executes the production operation for each single pattern CED model according to the above read results.

3. Merge sharing single pattern detection model

Merge sharing single pattern detection model mainly executes the merge sharing operations for single pattern detection model by merging sharing the same prefix, suffix, or subpattern among single pattern detection models and to produce a multiple pattern CED model based on the above decomposition results.

In our proposed scheme, three important functions—state transition function, failure transition function, and conditional output function—are used to realize the merging sharing function. Figure 5 shows the relationship among three functions.

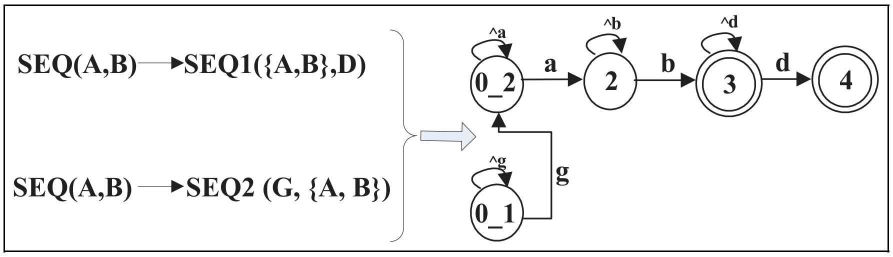

Figure 6. Merge sharing results of SEQ1(A,B,D) and SEQ2(G,A,B).

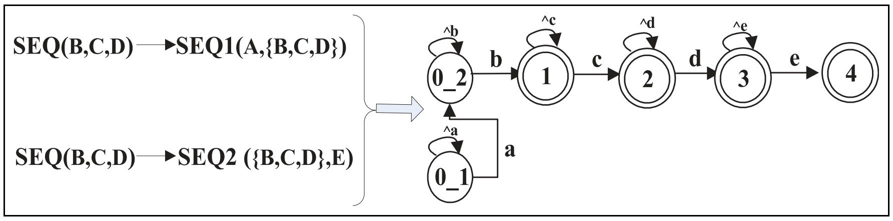

Figure 7. Merge sharing results of SEQ3(A,B,C,D) and SEQ4(B,C,D,E).

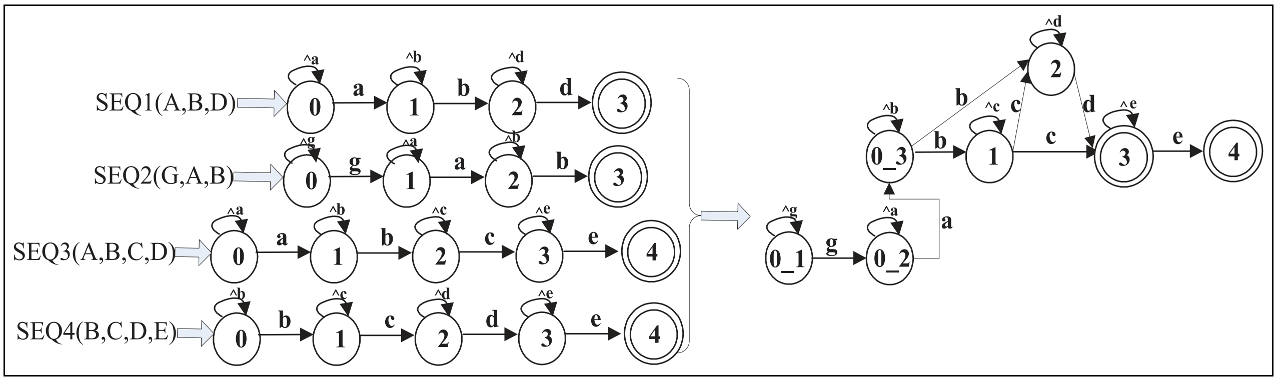

Figure 8. Multiple pattern detection model for SEQ1(A,B,D), SEQ2(G,A,B), SEQ3(A,B,C,D), and SEQ3(B,C,D,E).

Where state transition function mainly executes the transition state path when traversing single pattern detection model. Failure transition function mainly executes the transition path of the next step when it fails in its traversing process. Conditional state output function mainly executes the output operation for pattern detection results when it needs to output detection results. Figures 6 and 7 are the merge sharing results for SEQ1(A,B,D) and SEQ2(G,A,B), SEQ3(A,B,C,D) and SEQ4(B,C,D,E), respectively.

Construct multiple pattern detection model

Construct multiple pattern detection model mainly executes the construction operation for our suggested multiple pattern detection model using merging sharing technology. Figure 8 shows the construction results of multiple pattern detection model using merge sharing technology from the above SEQ1(A,B,D), SEQ2(G,A,B), SEQ3(A,B,C,D), and SEQ4(B,C,D,E).

From Figure 8, we can clearly see that our proposed multiple pattern detection model of complex event in this article can include less automata states, migration edges, and computation operations compared with the single pattern CED model due to its merging sharing for the same prefix, suffix, or subpattern based on their decomposition results, which can reduce many redundant building, storing, searching, and calculating operations for them, thus improving its whole detection performance.

Detection of multiple pattern detection model

After finishing the constructing of a multiple pattern detection model of complex events by merging sharing technology, we use the detection model to detect multiple complex events from massive event streams.

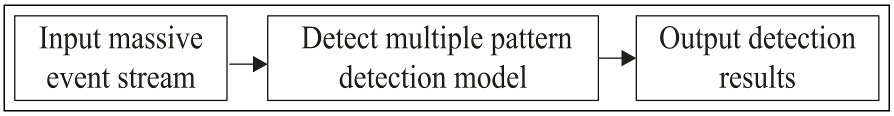
Input massive event stream
Detect multiple pattern detection model
Output detection results

Figure 9. Detection composition of our proposed method.

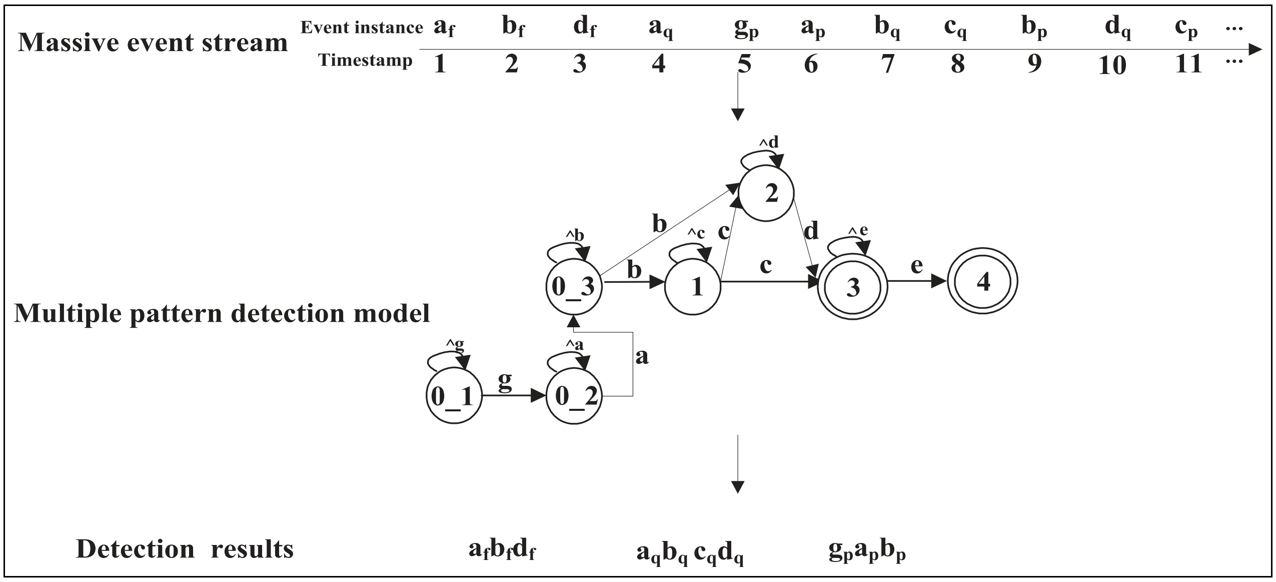

Figure 10. Detecting processing with our proposed method.

In our scheme, three important detection processes need to go through to realize the detection of multiple pattern complex events from massive event streams: input massive event stream, detect multiple pattern detection model, and output detection results. Figure 9 shows the detection composition of our proposed method.

Where input massive event stream mainly executes the input operation for the massive event stream into multiple pattern detection model; detect multiple pattern complex event mainly executes the detection operation for multiple pattern complex events with our proposed multiple pattern detection model from the above input massive event stream; output detection results mainly executes the output operation for multiple pattern complex events when finishing detection process. Figure 10 shows the detailed detecting processing for multiple pattern complex events with our proposed method in this article.

Figure 10 shows that our proposed detection method can quickly detect the related multiple pattern complex events from massive event streams with a general detection model one time by decomposing different pattern expressions into multiple subexpressions and merging sharing the same prefix, suffix, or subpattern into one to construct a general multiple pattern detection model. It does not need to independently construct multiple complex event detection models, thereby improving its whole detection efficiency for multiple pattern complex events.

Realizing steps of our algorithm

Figure 11 shows the algorithm flow of our proposed scheme, which mainly includes read event stream, read pattern expressions, decompose pattern expression, obtain the same subexpressions, build single pattern detection model, traverse single pattern detection model, merge sharing single pattern detection model, construct multiple pattern detection model, detect multiple pattern complex event, and output detection results. The detailed realizing steps for our proposed scheme can be summarized in the following steps:

In step 1: read two different pattern expressions of complex event.
In step 2: decompose the pattern expressions using our built composition rules.
In step 3: obtain the same subexpressions from the above decomposed pattern expressions.
In step 4: build single pattern CED models according to their pattern expressions of complex event above.

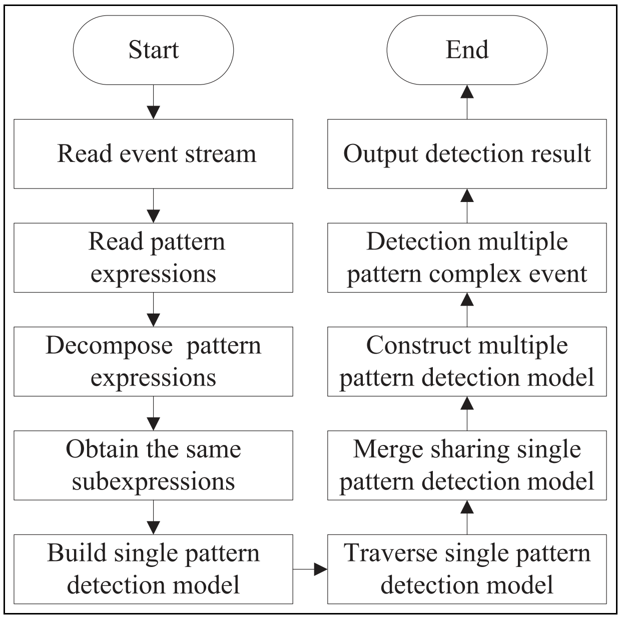
Start
Read event stream
Read pattern expressions
Decompose pattern expressions
Obtain the same subexpressions
Build single pattern detection model
Traverse single pattern detection model
Merge sharing single pattern detection model
Construct multiple pattern detection model
Detection multiple pattern complex event
Output detection result
End

Figure 11. Algorithm flow.

In step 5: traverse single pattern detection model above using depth-first search algorithm from its starting state.
In step 6: merge sharing single pattern detection model using three important functions as follows: state transition function, failure transition function, and conditional output function.
In step 7: construct a multiple pattern detection model of complex event by merging sharing technology.
In step 8: detection multiple pattern complex events using the above multiple pattern detection model.
In step 9: output detection result of multiple pattern complex events.

Cost model

In order to realize the rapid detection for multiple complex events from massive event streams, our proposed detect model in this article needs to meet the following cost models

$$\begin{cases} Cost_{ord}(DS+TS) \leqslant Cost(DS) + Cost(MS) \\ Cost(DS) = Cost_a + Cost_r + Cost_v\left(T, P_{N(s)}\right) \\ \quad + Selv\left(T, P_{N(s)}\right).Cost_n + Cost(D) + Cost(filter) \\ Cost(MS) = Cost_a + Cost_r + Cost_v\left(T, P_{N(s)}\right) \\ \quad + Selv\left(T, P_{N(s)}\right).Cost_n + Cost(M) + Cost(filter) \end{cases} \tag{1}$$

where $Cost_{ord}(DS+TS)$ refers to the optimal multiple pattern CED models based on decomposition and merge sharing; $Cost(DS)$ refers to all the multiple pattern CED models based on decomposition sharing; $Cost(MS)$ refers to all the multiple pattern CED models based on merge sharing; $Cost_a$ refers to the accessing cost of event instances from model; $Cost_r$ refers to the cost of removing the event instances from model; N refers to any given node in model; P_N represents the path from the root node to the N node in model; $N(s)$ represents the corresponding node S in model. $Cost_v(T, P_{N(s)})$ refers to the conditional verification cost between the new event type T and the event before it; $Selv(T, P_{N(s)})$ refers to the cost of conditional selection; $Cost_n$ refers to the cost of creating a new instance and inserting it into model; $Cost(D)$ refers to the cost of decomposing pattern expression in model; $Cost(M)$ refers to the cost of merging sharing the same prefix, suffix, or subpattern in model; and $Cost(filter)$ refers to the cost of filtering redundant the prefix, suffix, or subpattern in model.

Table 1. Main setting parameters in our experiment.

Parameter name	Parameter value
Sliding window size	40
Number of event types	10
Number of complex event detection expressions	40
Length of complex event detection expression	3 ~ 10
Scale of event stream	10^5
Number of attributes of each event	2

Experimental results and analysis

In this section, in order to verify the effectiveness of our proposed detection scheme above, some simulation experiments are designed. Our designed experiments mainly include the following two contents: build experimental environment and test the effectiveness of our proposed detection scheme, respectively.

Build experimental environment

Our simulation experiments are implemented on the Microsoft Windows 7 operating systems, AMD A6-3420M 4 core CPU Processor, 2 GB memory, and 500 GB hard disk. The Visual C++6.0 tool is used to develop an event generator, and then uses it to generate different kinds of RFID event streams by controlling some parameters. The main setting parameters are shown in Table 1 in our experiment.

Where sliding window size refers to the total number of events processed in a certain time period in our experiment. Number of event types refers to the total number of different event types used in our experiment. Number of CED expression refers to the number of detection expressions of complex event used in our experiment, which needs to transform into the corresponding detection models. Length of CED expression specifies the length of each CED expression used above in our experiment. Scale of event stream specifies the total testing scale of event stream used in our experiment, which is generated by our event generator. Number of attributes of each event refers to the number for each event type excluding the timestamp.

In our experiments, we mainly test the effectiveness of our proposed detection method from detection model and detection algorithm. In the detection model aspects, number of automata states and number of migration edges are selected as two important indicators to evaluate our detection model. Where the number of automata states refers to the total number of automata states that are used to generate multiple pattern detection model of complex event, while the number of migration edges represents the total number of migration edges that are used to build multiple pattern detection model of complex event under the same pattern expression of complex event. In the detection algorithm aspects, detection time, memory consumption, and event throughput are used as comparison indicators to evaluate our detection algorithm, respectively. Where the detection time refers to the total detecting time for the expected multiple complex events from massive RFID event streams. The memory consumption refers to the total memory consumption used for detecting the desired multiple complex events from massive RFID event streams. The event throughput refers to the detection number of the expected complex events from the massive RFID event streams in one unit time.

In our experiments, single pattern detection method, optimized method,[36] and multi-pattern sharing detection method[37] are selected as comparison methods to test the performance of our detection methods. Where single pattern detection method mainly executes pattern expressions of complex event independently (without any sharing technology); optimized method mainly uses a series of optimized algorithms based on nondeterministic automata model to reduce its bottlenecks and realize its CED; multi-pattern sharing method mainly uses multi-pattern sharing technology to share the same prefix, suffix, or subpattern in its all the pattern expressions. However, it cannot share the subexpressions before decomposition; while our proposed multiple pattern processing method in this article mainly use decomposition and merge sharing technology to realize the detection of multiple different complex events, which is introduced in section "Proposed scheme" of this article. In order to reduce the randomness of testing, five tests are taken in our experiment of this article.

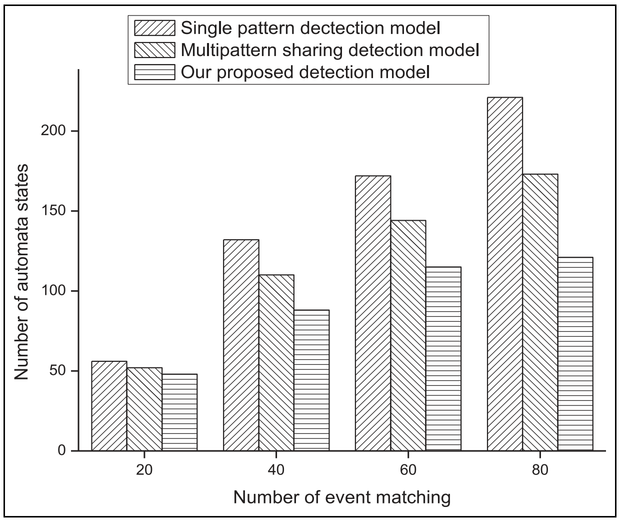

Figure 12. Comparison in number of automata states.

Test the effectiveness of our detection method

Test the effectiveness of our detection model

Test number of automata states. In this section, we mainly test the effectiveness of our detection model from the number of automata states with single pattern detection model and multi-pattern sharing detection model. The experimental result is shown in Figure 12.

From Figure 12, we can see clearly that that our proposed detection model needs less automata states compared with other two detection models under the same testing conditions. The reasons for that lies that, in our detection model, we mainly use decomposition sharing technology to decompose the pattern expressions into many subexpressions, which can provide many sharing opportunities for pattern sharing, and use merge sharing technology to merge sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results, which can reduce lots of redundant automata states for them, therefore needing less number of automata states. Multi-pattern sharing detection model needs more automata states compared with our detection model because it can only use the multi-pattern sharing technology to share the same prefix, suffix, or subpattern in its all pattern expressions, but it cannot share these subexpressions before decomposition, thus needing more automata states. However, since single pattern detection model cannot use the sharing technology in all the pattern expressions due to their unsharing characteristic and needs to independently construct every CED model to complete its detection for each pattern expressions, therefore requiring the most number of automata states.

Test migration edges. In this section, we mainly test the effectiveness of our detection model from the number

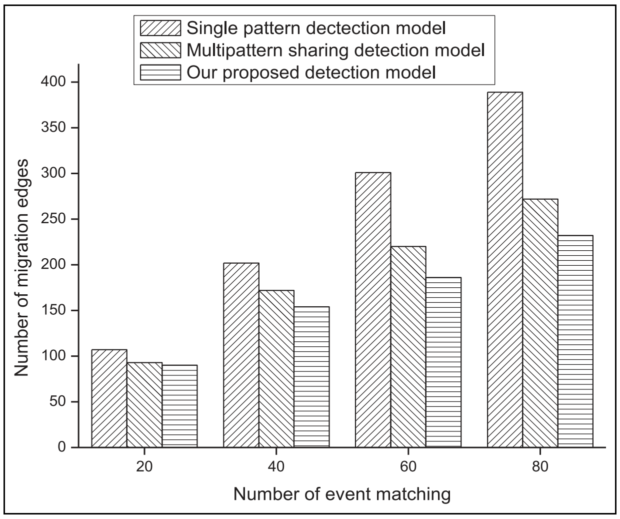

Figure 13. Comparison in number of migration edges.

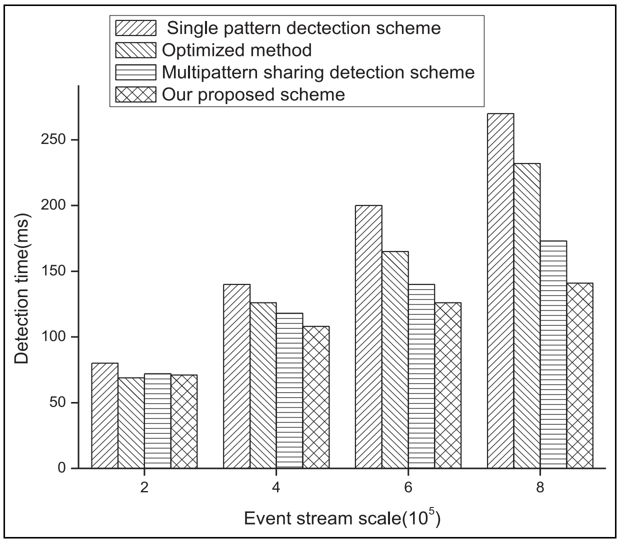

Figure 14. Comparison in detection time.

of migration edges with single pattern detection model and multi-pattern sharing detection model. The experimental result is shown in Figure 13.

Figure 13 shows that our detection model in this article needs less number of migration edges compared with multi-pattern sharing detection model and single pattern detection model under the same testing conditions. The reasons for that lies that, in our scheme, first decomposition sharing technology is used to decompose pattern expressions into many subexpressions, which can provide many sharing opportunities for subexpressions; second, merge sharing technology is used to merge sharing all the same prefix, suffix, or subpattern in all the pattern expressions, including decomposition subexpressions, which can reduce lots of redundant migration edges, therefore requiring less number of migration edges. Because multi-pattern sharing detection model can only realize the sharing detection for the same prefix, suffix, or subpattern in pattern expressions, but it cannot realize the sharing detection for these subexpressions before decomposition; therefore, it needs more migration edges compared with our proposed detection model. However, single pattern detection model needs the most migration edges lies in its unsharing detection characteristic in all the pattern detection expressions, including decomposable pattern expressions, and it needs to independently construct all CED models to complete their detection, therefore needing the most number of migration edges in three methods.

Test the effectiveness of our detection algorithm

Test detection time. In this section, we evaluate the effectiveness of our proposed detection algorithm from detection time. The experimental result is shown in Figure 14.

From Figure 14, we can see clearly that our proposed detection method shows the least detection time in four detection methods. Multi-pattern sharing detection method and optimized method followed. Single pattern detection method consumes the most detection time. The main reasons for that are that, in our suggested scheme, we use decomposition sharing technology to decompose pattern expressions into many subexpressions and use merge sharing technology to merge sharing the same prefix, suffix, or subpattern in all pattern expressions, including subexpressions after decomposing pattern expressions, which can reduce many building, storing, searching, and calculating operations for them, thus saving much detection time. Multi-pattern sharing detection method costs more detection time compared with our proposed method lies in the use of multi-pattern sharing among pattern expression in its scheme, which can eliminate many the same prefix, suffix, or subpattern in all its pattern expressions, but it cannot share the subexpressions before decomposition. Optimized method used series of optimized algorithms based on nondeterministic automata model to realize its detection function in its processing of complex event; therefore, it needs less detection time compared with single pattern detection method. Single pattern detection method costs the most detection time because it is unable to realize the sharing detection for all the pattern expressions, including the decomposable pattern expressions, and it needs to independently construct multiple different CED models to complete its detection function, therefore requiring most detection time in four detection methods.

Test memory consumption. In this section, we evaluate the effectiveness of our proposed detection algorithm from memory consumption. Figure 15 shows the experimental result.

Figure 15 reveals that our proposed detection scheme in this article has a significant improvement in saving memory consumption compared with other three detection methods. Multi-pattern sharing detection method and optimized method followed. Single pattern detection method shows the most memory consumption. The main reasons for that can be explained as follows: in our scheme, decomposition sharing technology is used to decompose many pattern expressions into many subexpressions, and merge sharing technology is used to merge sharing all the same prefix, suffix, or subpattern in all the pattern expressions, including subexpressions in decomposable pattern expressions, which can reduce many redundant building, storing, looking up, and calculating operations for them, thus saving more memory consumption. Multi-pattern sharing detection method mainly used multi-pattern sharing technology to reduce the same prefix, suffix, or subpattern in all pattern expressions; however, it cannot share detection for the subexpressions in decomposable pattern expressions, thus costing more memory consumption than our proposed scheme. Optimized method mainly used series of optimized algorithms based on nondeterministic automata model to realize its detection function during its detection of complex event, thus saving less memory consumption than single pattern detection method. Single pattern detection method costs the most memory consumption lies that it is unable to realize the sharing detection for all the pattern expressions, including decomposable pattern expressions, and it needs to independently construct multiple different CED models to complete its detection, therefore consuming the most memory consumption in four methods.

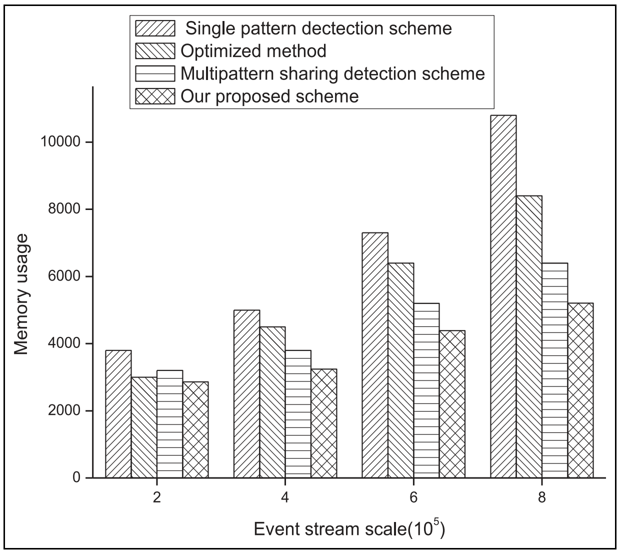

Figure 15. Comparison in memory consumption.

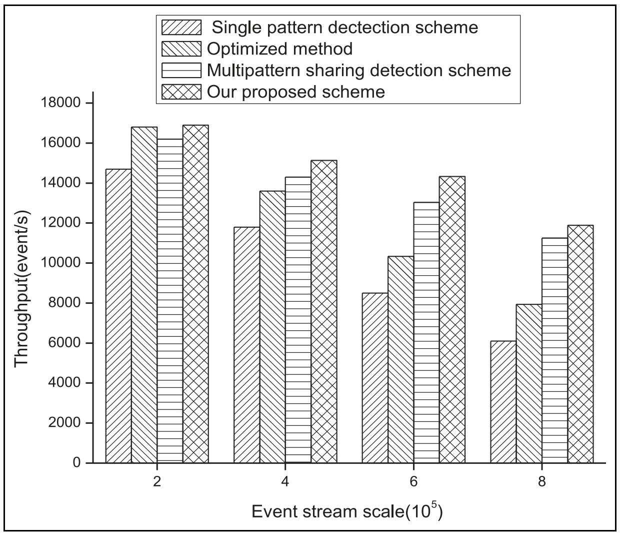

Figure 16. Comparison in event throughput.

Test event throughput. In this section, we evaluate the effectiveness of our proposed detection algorithm from event throughput. The experimental result is shown in Figure 16.

From Figure 16, we can observe that our proposed detection scheme shows the good event throughput processing capacity compared with other three methods. Multi-pattern sharing detection method and optimized method followed. Single pattern detection approach presents the worst processing performance. The main reason lies in the use of decomposition and merge sharing technology in our proposed scheme. Specially, in our method, we use decomposition sharing technology to decompose pattern expressions into multiple sharing subexpressions and use merge sharing technology to construct a general multiple pattern detection model of complex event by merging sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results, and then improve the event throughput of our scheme. Multi-pattern sharing detection method can improve its event throughput capacity lies in the use of multi-pattern sharing in its pattern expression, which can eliminate many the same prefix, suffix, or subpattern in them. However, it cannot share the subexpressions in decomposable pattern expressions. Optimized method shows a better processing capability compared with single pattern detection method because of the uses of series of optimized algorithms based on nondeterministic automata model in its detection scheme of complex event. Single pattern detection method presents the worst processing performance because it needs to repeatedly execute the building, storing, looking up, and calculating operations for all the pattern expressions due to its unsharing pattern, including the subexpressions in decomposable pattern expressions, which needs to execute many unnecessary detection operations, therefore leading to lowest event detection performance.

In addition, from Figures 14–16, we can also see that four detection methods show the similar detection performance in small event stream scale, but with the increasing of event stream scale, our proposed scheme is obviously superior to other three methods.

Conclusion

In this article, a multiple pattern CED scheme based on decomposition and merge sharing is presented for massive event stream. In our scheme, we first successfully use decomposition technology to decompose pattern expressions into multiple subexpressions, which can provide many sharing opportunities for the subexpressions. We then use merging sharing technology to construct a multiple pattern CED model by merging sharing all the same prefix, suffix, or subpattern into one based on the above decomposition results. As a result, our proposed detection method in this article can effectively solve the above problem. The simulation results show that our proposed scheme in this article has a great improvement in reducing automata states number and migration edges number, saving detection time, lowering memory access, and improving event throughput compared with some general detection methods from massive event streams.

Acknowledgements

The authors authentically thank associate professor Wang Tao, who offered us some valuable suggestions and help.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China (nos 61602187 and 6180405); the National Key Research and Development Plan (no. 2016YFD0200700); the Guangdong Science and Technology Projects (no. 2019B020219002); and the Guangdong Laboratory of Lingnan Modern Agriculture.

ORCID iD

Jianhua Wang https://orcid.org/0000-0001-9587-2845

References

1. Cheng L, Wang T, Hong X, et al. A study on the architecture of manufacturing internet of things. *Int J Model Ident Control* 2015; 23(1): 8–23.
2. Tao F, Cheng Y, Da Xu L, et al. CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. *IEEE Trans Ind Inform* 2014; 10(2): 1435–1442.
3. Bi Z, Da Xu L and Wang C. Internet of things for enterprise systems of modern manufacturing. *IEEE Trans Ind Inform* 2014; 10(2): 1537–1546.
4. Wang JHLUJ, Lan YB and Cheng LL. An efficient complex event detection algorithm based on NFA_HTS for massive RFID event streams. *J Electri Eng Technol* 2018; 13(2): 989–997.
5. Del G, Eugene W, Hee J, et al. SASE: complex event processing over streams. In: *Proceedings of the 3rd biennial conference on innovative data systems research*, Asilomar, CA, 7–10 January 2007, pp.407–411. https://arxiv.org/ftp/cs/papers/0612/0612128.pdf
6. Wang F, Liu S and Liu P. Bridging physical and virtual worlds: complex event processing for RFID data streams. In: *Processing of the 10th international conference on EDBT*, Munich, 26–31 March 2006, pp.588–607. Heidelberg: Springer.
7. Bai L, Lao S, Smeaton AF, et al. Semantic analysis of field sports video using a Petri-net of audio-visual concepts. *Computer* 2009; 52(7): 808–823.
8. Sun XW, Chen R and Du ZJ. Composite event detection based on automata. In: *Proceedings of 2009 IEEE international conference on intelligent human-machine systems and cybernetics*, Hangzhou, China, 26–27 August 2009, pp. 160–163. New York: IEEE.
9. Mei Y and Madden S. ZStream: a cost-based query processor for adaptively detecting composite events. In: *Proceedings of the 2009 SIGMOD, Providence, 28–29* June 2009, pp.193–206. New York: ACM.
10. Jin X, Lee X and Kong N. Efficient complex event processing over RFID data stream. In: *Proceedings of the 7th IEEE/ACIS international conference on computer and information science (ICIS)*, Portland, OR, 14–16 May 2008, pp.75–81. New York: IEEE.
11. Li Y, Wang J and Feng L. Accelerating sequence event detection through condensed composition. In: *Proceedings of the 5th international conference on ubiquitous information technologies & applications*, Sanya, China, 16–18 December 2010, pp.6–10. New York: IEEE.
12. Cao J, Wei X, Liu YQ, et al. LogCEP-complex event processing based on pushdown automaton. *Int J Hybrid Inform Technol* 2014; 7(6): 71–82.
13. Liu M, Rundensteiner E, Green field K, et al. E-cube: multi-dimensional event sequence analysis using hierarchical pattern query sharing. In: *Proceedings of the 2011 ACM SIGMOD international conference on management*

of data, Athens, Greece, 12–16 June 2011, pp.889–900. New York: ACM.
14. Ray M, Lei C and Rundensteiner EA. Scalable pattern sharing on event streams. In: *Proceedings of the 2016 international conference on management of data*, San Francisco, CA, 26 June–1 July 2016, pp.495–510. New York: ACM.
15. Zhang SH, Vo HT, Dahlmeier D, et al. Multi-query optimization for complex event processing in SAP ESP. In: *Proceedings of 2017 IEEE 33rd international conference on data engineering (ICDE)*, San Diego, CA, 19–22 April 2017, pp.1213–1224. New York: IEEE.
16. Kolchinsky I and Schuster A. Real-time multi-pattern detection over event streams. In: *Proceedings of the 2019 ACM international conference on management of data*, Amsterdam, 30 June–15 July 2019, pp.589–606. New York: ACM.
17. Chen J, Gong Z and Liu W. A nonparametric model for online topic discovery with word embeddings. *Inform Sci* 2019; 504: 32–47.
18. Hu J and Zheng W. Transformation-gated LSTM: efficient capture of short-term mutation dependencies for multivariate time series prediction tasks. In: *Proceedings of 2019 IEEE international joint conference on neural networks, Budapest*, 14–19 July 2019, pp.1–8. New York: IEEE.
19. Chen J, Gong Z and Liu W. A Dirichlet process biterm-based mixture model for short text stream clustering. *Appl Intell* 2020; 50: 1609–1619.
20. Hu J and Zheng W. Multistage attention network for multivariate time series prediction. *Neurocomputing* 2020; 383: 122–137.
21. Wang W, Chen J, Wang J, et al. Trust-enhanced collaborative filtering for personalized point of interests recommendation. *IEEE Trans Ind Inform* 2019; 16: 6124–6132.
22. Suhothayan S, Gajasinghe K, Narangoda I, et al. Siddhi: second look at complex event processing architectures. In: *Proceedings of the 2011 ACM workshop on gateway computing environments*, Seattle, WA, 14–19 November 2011, pp.43–50. New York: ACM.
23. Mayer R, Koldehofe B and Rothermel K. Predictable low-latency event detection with parallel complex event processing. *IEEE Internet Things J* 2015; 2(4): 274–286.
24. Kalyvianaki E, Wiesemann W, Vu QH, et al. SQPR: stream query planning with reuse. In: *Proceedings of IEEE international conference on data engineering (ICDE)*, Hannover, 11–16 April 2011, pp.840–851. New York: IEEE.
25. Ray M, Lei C and Rundensteiner EA. Scalable pattern sharing on event streams. In: *Proceedings of the 2016 ACM international conference on management of data*, San Francisco, CA, 14–16 June 2016, pp.495–510. New York: ACM.
26. Liu M, Rundwnsteiner E, Dougherty D, et al. High-performance nested CEP query processing over event streams. In: *Proceedings of IEEE international conference on data engineering (ICDE)*, Hannover, 11–16 April 2011, pp.123–134. New York: IEEE.
27. Ma M, Wang P, Chu CH and Liu L. Efficient multipattern event processing over high-speed train data streams. *IEEE Internet Things J* 2014; 2(4): 295–309.
28. Ma M and Wang P. Efficient event inference and context-awareness in internet of things edge systems. *IEEE Trans Big Data*. Epub ahead of print 29 March 2019. DOI: 10.1109/TBDATA.2019.2907978.
29. Giannikis G, Makreshanski D, Alonso G, et al. Shared workload optimization. *VLDB* 2014; 7(6): 429–440.
30. Zervakis L, Setty V, Tryfonopoulos C, et al. Efficient continuous multi-query processing over graph streams. *arXiv Preprint arXiv* 2019; 1902: 05134.
31. Schultz-Møller NP, Migliavacca M and Pietzuch PR. Distributed complex event processing with query rewriting. In: *Proceedings of the 3rd ACM international conference on distributed event-based systems*. Nashville, TN, 6–9 July 2009, pp.1–12. New York: ACM.
32. Xiao F, Zhan C, Lai H, et al. New parallel processing strategies in complex event processing systems with data streams. *Int J Distrib Sens N* 2017; 13(8): 1550147717728626.
33. Yuan L, Xu D, Ge G, et al. Study on distributed complex event processing in Internet of things based on query plan. In: *Proceedings of IEEE international conference on cyber technology in automation, control, and intelligent systems*, Shenyang, China, 8–12 June 2015, pp.666–670. New York: IEEE.
34. Xiao F. An intelligent complex event processing with D numbers under fuzzy environment. *Math Prob Eng* 2016; 2016(1): 1–10.
35. Abbasnejad I, Sridharan S, Denman S, et al. Complex event detection using joint max margin and semantic features. In: *Proceedings of IEEE international conference on digital image computing: techniques and applications, Gold Coast*, QLD, Australia, 30 November–2 December 2016, pp.1–8. New York: IEEE.
36. Zhang H, Diao Y and Immerman N. On complexity and optimization of expensive queries in complex event processing. In: *Proceedings of ACM SIGMOD international conference on management data*, Snowbird, UT, 22–27 June 2014, pp.217–228. New York: ACM.
37. Wang JH, Lan YB, Lu SL, et al. An efficient complex event processing algorithm based on multipattern sharing for massive manufacturing event streams. *KSII T Internet Info Syst* 2019; 13(3): 1385–1402.
38. Scanagatta M, Corani G, De Campos CP, et al. Approximate structure learning for large Bayesian networks. *Mach Learn* 2018; 107(8–10): 1209–1227.
39. Liu S and Liu J. CP-nets structure learning based on mRMCR principle. *IEEE Access* 2019; 7: 121482–121492.
40. Jiang M, Cui P, Beutel A, et al. Catching synchronized behaviors in large networks: a graph mining approach. *ACM T Knowl Discov D* 2016; 10(4): 1–27.