Incremental Mining for Facility Management*

Katja Hose Marcel Karnstedt
Daniel Klan Kai-Uwe Sattler
Department of Computer Science and Automation,
TU Ilmenau, Germany

Jana Quasebarth
Research and Development,
NT Neue Technologie AG, Germany

Abstract

Modern buildings are equipped with high-tech systems that take care of several fundamental aspects, e.g., air-conditioning, heating and water supply. The requirements posed on facility management by such buildings are challenging. Modern techniques implement adaptive control systems to achieve this, in which decisions are preferably based on the results of (multiple correlated) mining tasks on recently gathered sensor data. In this work, we discuss the general relationship between such control systems and the underlying mining tasks. We exemplary choose change detection in the context of pattern analysis as a representative, because this mining task involves general requirements known from stream processing like the need for incremental algorithms, but also poses specific challenges like in-time detection. We present three concrete approaches for this and an according evaluation.

1 Introduction

Because of the growing number of installed systems within modern office buildings configuring all these systems manually is difficult or even impossible. The interaction between those systems is rarely considered. State-of-the-art controllers provide primitive functions for controlling the cooperation of different devices. However, the components are often adjusted and optimized manually. In many cases systems run suboptimally with factory or initial settings made at installation time. Because of the complexity of system functionalities and correlations, not only between the system components themselves but also between the system and external factors such as the weather, expert knowledge is required to change system settings properly. With respect to rising energy costs and pollution control, there is a growing interest in (automatically) optimizing the operation of buildings.

Finding a configuration that reconciles the goals of all concerned parties is rather complicated and depends on many influencing factors. Consequently, a solution that does this (semi-)automatically and reacts upon events such as changes in the price of electricity is highly desirable. In summary, the main objectives of an intelligent facility management are

- minimization of maintenance and operation costs
- minimization of ecological costs
- improving or at least preserving user convenience
- forecasting of operation costs and the environmental impact of the system under different premises (what-if-analyses)

As modern buildings are equipped with many installations, e.g., heating, air-conditioning, etc, there are many sensors and actuators that continuously produce streams of readings. A good idea in this context is to apply methods known from data mining and signal processing. This may involve burst detection, classification as well as detecting correlations between different sensors. As sensor data is usually generated continuously at a rapid rate, approaches from stream mining are becoming the focus of interest. Especially for discovering correlations which are unknown or even never supposed to exist, methods for pattern recognition promise to be very effective. An item can be seen as a combination of a sensor ID and the (optionally discretized) sensor value, an itemset as a set of items collected at (again, optionally discretized) times. For instance, $\{s1, x20, y18\}$ may encode the sensor information "sun is shining, temperatur x is 20, temperatur y is 18". Applying this scheme, different itemsets may be formed by combining different sensors (multiple times), depending on application specific conditions. If multiple sensor data are combined and encoded to form itemsets, frequent itemset mining is applied as a basis for frequent pattern detection and association rule mining. A pattern (resp. itemset) is frequent in a stream of transactions, if it occurs in more than σ percent of the actual stream. Here, σ is called the required *support* of the itemset. With, usually approximating, stream mining methods, the actual support of itemsets declared as frequent may vary by a provided *error bound* ϵ. We use the term transaction to refer to a set of itemsets, which merely stems from the popular application field of customer basket analysis. An outstanding requirement of automated facility management solutions is the capability of in-time detection of changes and according reaction. This also holds for the detection of changes in correlations symbolized by frequent patterns. Thus, efficient and in-time change detection is a field of high interest in this sector, but almost unexplored. The chosen field of application implies the incremental character of appropriate methods as a major requirement. As a second main requirement for automated facility management, we expect any algorithm to support querying arbitrary time intervals. Beside the fast reaction to changing situations, this allows for analyzing and reasoning about sensor correlations belatedly, e.g., in order to explain problems that occurred or to optimize settings for the future. In this work, we present three different approaches for solving this ex-

*This work was supported by the BMBF under grant 03WKBD2B.

emplary task.

This paper is organized as follows. In Section 2 we describe the idea of automated facility management and discuss the importance of data mining in this context, as well as a general architecture for such systems. In Section 3 we briefly present the frequent itemset mining algorithm applied for pattern detection, and afterwards propose three concrete methods for change detection on these frequent patterns. Related work is covered in Section 4, whereas Section 5 includes exemplary results from an extended evaluation. Finally, we conclude in Section 6.

2 Automated Facility Management

This section gives an overview of the main challenges and application scenarios in automated facility management. Furthermore, we sketch the architecture of such an environment and point out in what aspects data mining is beneficial.

2.1 Application Scenarios

Nowadays, facility management is no longer a simple matter. First, various autonomous systems have to be calibrated individually. Second, these systems should be configured in a way that enables them to work together efficiently. A worst case scenario for example would be a situation where the air-conditioning cools the air while radiators are warming it. However, the usual case is not that extreme but there is great potential for optimization in nearly all buildings. As initial situation we assume each building to have a number of autonomous installations (air-conditioning, heating, automatic rolling shutters, lights, water, etc.). In general, these systems are programmed with primitive functions without consideration of much additional information.

Optimizing a single installation already may be challenging. Let us take a conventional boiler heating system as an example. Heating installations can be programmed with primitive functions such as "reduce operation at night" or "do not operate the radiator heating circuit in summer, just heat drinking water". Unfortunately, transition from winter to summer operation mode and vice versa is either initiated by the user or determined by a fixed date disregarding the actual weather conditions. A more intriguing solution would be to consider weather forecast information. However, given the information that it is going to be a rather warm day, maybe heating the rooms will not be necessary at all. Such considerations are particularly interesting for transition times from one season to another and are closely related to the consumption of resources and to costs.

Second, users mostly have to specify the times for starting reduced or standard operation instead of specifying the times of utilization. Central heating installations are rather slow systems. Thus, they have to start to increase their flow temperature and turn on the pumps supplying the radiator heating circuit sometimes over two hours in advance in order to provide the set temperature for the time of utilization. This often results in a long-term manual trial-and-error approach until the desired comfort is reached and the right start and end times are found.

Third, conventional heating installations consider outdoor temperature using sensors and adapt the flow temperature according to the outdoor temperature. This is done by a simple characteristic curve relating the outdoor temperature to a specific flow temperature and a controller regulating the flow temperature. This curve can be shifted by a parallel translation and its inclination can be changed. Settings in this context depend on the building characteristics, but they have impact on comfort and costs. However, such functions are usually static in conventional installations – once manually defined by the installer and never adapted.

Altogether, an automated facility management system receives input from a considerable quantity of sensors (flow temperature of the heating installation, inside and outside temperatures, etc.) and actuators (burner, pumps, valves, ventilators, etc.) providing a vast amount of *process data* describing current states and actions. However, current installations mostly work completely independent from each other. Thus, air-conditioning and heating do not know from each other. Besides, since the air-conditioner is in most cases also capable of warming air instead of only cooling it, it might be interesting to combine this heating capacity with the actual heating installation. Especially with respect to energy costs (electricity, heating oil) the economic optimum might change frequently and is not easy to find. Furthermore, aspects such as physical well-being, room properties, and condition (room isolation, room size, etc.) are not considered. As long as the systems do not interact with each other or consider additional information, coherences and dependencies cannot be identified and consequently an optimal solution cannot be found.

2.2 System Architecture

The architecture for automated facility management that we propose consists of a large number of sensors/actuators, a controller, and the facility management system. It is shown in Figure 1.

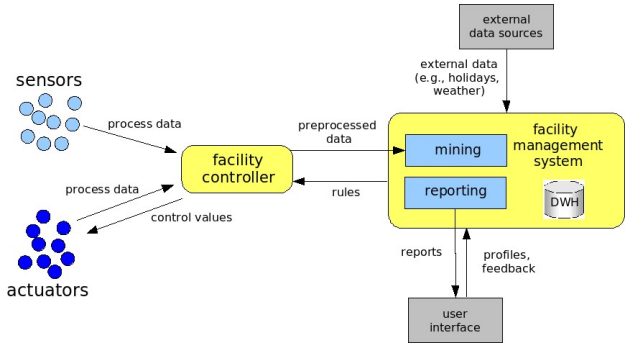


Figure 1: System architecture

As stated above, current buildings accommodate different autonomous installations such as heating or air-conditioning systems. In general, installations may have actuators as well as sensors. These sensors record process data such as the current flow temperature of the heating installation. In addition to the sensors integrated into these installations, further sensors are used to obtain additional information about parameters specific to a particular building, e.g., air humidity, current room temperature, movement, and light. Both types of sensors as well as actuators produce a continuous data stream.

The facility controller collects these readings, preprocesses them and forwards the results to the facility management system. Within the preprocessing step the controller already runs a plausibility test on the input sensor and actuator readings and notifies the facility management system about possibly defective sensors. Each building has its own facility controller. On the contrary, facility management systems are not specific to individual buildings.

Multiple controllers can communicate with one global facility management system. In order to reduce transmission costs only changes in the actuator and sensor readings that are detected by the facility controller are sent to the facility management system. Such a data item is a triple (s, v, t), where s denotes the sensor resp. actuator id, v denotes the measured value, and t denotes a time stamp that indicates when the values were measured.

The facility management system generates rules for the controller based on the current process data, user profiles, and external metadata such as holidays, the daily weather report, or the current tariffs for electricity, water, or gas. Finally, the controller can use these rules to generate control values for the actuators. In addition to the output rules the knowledge management system generates reports [Motegi and Piette, 2002] for the users. Such reports represent information about which installation causes what costs and may serve as a basis for decision-making with respect to comfort settings and costs. The high number of sensors and actuators results in a continuous data. Storing all these tuples into a database is impossible.

2.3 Mining Tasks

The applications of data mining in automated facility management are manifold. We need classification and clustering as well as algorithms for creating association rules.

An example for the application of classificators is to decide whether the current room temperature conforms to the user well-being. Sometimes this is rather simple, 15°C in an office where people used to work is definitely too cold. However, that temperature is still acceptable for an office that is momentarily unused (e.g., because of holidays). Furthermore, in cases of sudden increase of temperature and heat the system should detect that change and for example raise an alarm since this could mean fire. Furthermore, classification strategies allow us for instance to define classes of actuator readings whose combination should be avoided (e.g., air-condition systems and heating systems running at the same time). We can also use the classifier to identify substitution classes. Then, the rule generator can use the substitution classes to define different configurations with the same effect but different costs. Clustered sensor values within a room possibly indicate that the sensor concentration within the room is too high.

Association rules can be used to detect new coherences between sensors, actuators, and user preferences resp. maintenance costs. Based on the sensor readings that have been sent to the facility management system, buffer resp. window techniques are applied and a stream of itemsets is formed. On this stream frequent itemsets are determined. Based on these itemsets association rules are determined, e.g., if the sun shines the temperature sensor that is located at the window of a room measures 2 degrees more than the temperature sensor at the door (located several meters away from the window). Assume there is an event that changes this coherency such that the difference between the two sensors is only 1 degree. The reason might be that blind has been lowered.

Operating facility management systems when knowing about all correlations of dimensions (sensor/actuator reading, energy consumption) is straightforward. However, it is a challenging task to detect correlations between arbitrary dimensions that are not obvious and thus not known at the initial start-up of the facility management system. In order to detect such correlations we propose the use of frequent itemset mining. However, it is not enough to detect such correlations since they can change over time. In some cases it is necessary to react immediately upon such changes. This is the problem that we focus on in this paper: (incremental) change detection of frequent itemsets.

3 Detecting Frequent Patterns in Data Streams

Frequent itemset mining deals with the problem of identifying sets of items occurring together in so-called transactions frequently. Basically, two classes of algorithms can be distinguished: approaches with candidate generation (e.g., the famous apriori algorithm [Agrawal *et al.*, 1993; Agrawal and Srikant, 1994; Manku and Motwani, 2002]) as well as without candidate generation. Here, only the latter ones are suitable for stream mining. Usually, these approaches are based on a prefix-tree-like structure. In this tree – the frequent pattern (FP) tree – each path represents an itemset in which multiple transactions are merged into one path or at least share a common prefix if they share an identical frequent itemset [Han *et al.*, 2000]. For this purpose, items are sorted as part of their transaction in their frequency descending order and are inserted into the tree accordingly. Again, the FP tree is used as a compact data summary structure for the actual (offline) frequent pattern mining (the FP growth algorithm).

In order to mine streaming data in a time-sensitive way an extension of this approach was proposed [Giannella *et al.*, 2003]. Here, so-called tilted time window tables are added to the nodes representing window-based counts of the itemsets. The tilted windows allow to maintain summaries of frequency information of recent transactions in the data at a finer granularity than older transactions. The extended FP tree, called pattern tree, is updated in a batch-like manner: incoming transactions are accumulated until enough transactions of the stream have arrived. Then, the transactions of the batch are inserted into the tree. For mining the tree a modification of the FP growth algorithm is used taking the tilted window table into account. The original approach assumes that there is enough memory available to deliver results in the given quality and no way is described how to proceed if the algorithm runs out of memory.

As frequent itemset mining algorithms on data streams usually produce approximate results, there may be some false positives in the resulting output. Therefore, we need an algorithm that guarantees an error threshold. Additionally, the approach has to be time-sensitive. The FP-Stream approach in [Giannella *et al.*, 2003] is capable to satisfy these requirements. Asked for the frequent itemsets of a time period $[t_s, t_e]$, FP-Stream guarantees that it delivers all frequent itemsets in $[t_{s'}, t_{e'}]$ with frequency $\geq \sigma \cdot W$, where W is the number of transactions the time period $[t_{s'}, t_{e'}]$ contains. $t_{s'}$ and $t_{e'}$ are the time stamps of the used tilted time window table (TTWT) that correspond to t_s and t_e, depending on Q_{Tg}. The result may also contain some itemsets whose frequency is between $(\sigma - \varepsilon) \cdot W$ and $\sigma \cdot W$.

[Franke *et al.*, 2005; 2006] presents a modified version of this algorithm, which was designed to be resource- and quality-aware in parallel. The main contribution of this work is the identification of parameters the resource consumption of the algorithm is sensitive to. Based on this, it proposes when and how to change these parameters in order to meet given resource limits, while adhering to specific

output quality requirements.

The focus of this paper is the application of the frequent itemset mining algorithm from [Franke *et al.*, 2005; 2006] with priority on a in-time (subsequent or parallel) change detection. The incremental approaches for such a change detection are presented in detail in the next subsection.

3.1 Incremental Change Detection

In [Franke *et al.*, 2006] we already discussed general approaches for incremental change detection on data streams, particularly in the context of frequent itemset mining. We reasoned about the challenges and inferential requirement for this task, and identified two general approaches: *(i)* approaches on separate data structures, and *(ii)* approaches operating directly on the pattern tree. Beside particular pros and cons for each, which will be discussed along with the corresponding algorithms later on in this section, they are characterized by a major difference: methods following the first approach will be represented by separate operators in a complex mining task, they are processed on the output of a preceding frequent itemset operator. Methods of the second class are integrated in these operators, i.e., change detection is processed in parallel to the mining for actual frequent itemsets.

The first method, called *CT* (*C*hange *T*able), stores all frequent itemsets produced so far together with additional information (e.g., temporal) in a table. Each row of that table represents one frequent itemset. This is a naive, but effective approach. The content of that table can be queried for changes in arbitrary time intervals. This allows for detecting a wide variety of changes, even temporal ones. Unfortunately, the task of detecting changes in one specific itemset (e.g., itemset $\{a1, a3, c2\}$ changes to $\{a1, a3\}$ or $\{a1, a3, b2\}$) is complicated, because there is no information about the location of itemsets in the pattern tree if they are registered after being output from the frequent itemset operator. Thus, all itemsets in that table must be compared, which may consume a lot of computing time. Of course, this data structure allocates memory in parallel to the frequent itemset operator, thus, resources must be shared between both. However, the resource-adaptive techniques introduced in [Franke *et al.*, 2006] could be applied to this data structure in order to meet given resource limits. Other approaches on separate data structures have not been implemented yet, as they promise similar characteristics and challenges in their handling. The CT method is simple, but effective, and thus, representative for this class of change detection methods. The incremental processing of input data is implied in this case, by means of the output frequency of the preceding mining operator. Each time a new set of frequent itemsets is output and inserted in the table, we can detect simple as well as sophisticated (see [Franke *et al.*, 2006] for a more detailed explanation) changes for each single itemset – if new or already inserted before. This is incremental, but with rising table size very expensive.

A maybe more intuitive idea is to try to detect changes directly from the pattern tree used in the frequent itemset mining operator. Beside the expected improvement in the reaction time, this would allocate no extra memory. We implemented two methods based on this idea: *CDM* (*C*hange *D*etection during *M*anipulation) detects changes as soon as the pattern tree is modified, and *CDFISM* (*C*hange *D*etection during *F*requent *I*temset *M*ining) detect changes after executing the FP-growth algorithm on the pattern tree (when looking for the actual frequent itemsets after each batch). We have to consider three different modifications:

- a new node is inserted
- a node is deleted
- the TTWT in a node is changed

Moreover, we use the information stored in the TTWTs in order to detect changes over arbitrary time intervals. Note that the insertion of a node does not urgently reflect in a change of the frequent itemset. This happens not until the required support is achieved, which is signalized by modifying the according TTWT entries.

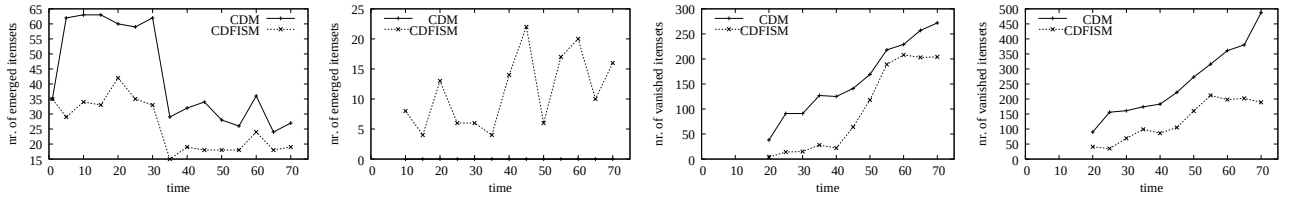
CDM is the method which promises the best reaction time, which is intuitively a major requirement for change detection in data streams, especially for the aspired automated facility management. But, we can only detect changes in itemsets that are modified in the current batch run. If a change in an itemset is only recognized during the run of FP-growth, the CDM method will not detect this change, but the CDFISM method will. If the TTWT entries of an existing node are modified, we always have to compare the current frequency with the former ones. In analogy, the deletion of a node does not urgently reflect in the drop of a frequent itemset. Again, more detailed comparisons have to be made. Due to its nature, querying time intervals which do not start at the current point in time is senseless, as manipulations in former time steps cannot be reconstructed. With the CDM method we may signalize false positives (so called *change candidates*), because changes are detected before a batch is finally processed. This incremental character allows, in parallel, for a very fast change detection.

The disadvantage of CDFISM is that it can only be run after processing a whole batch and has to completely traverse the tree – which leads to worse reaction time and less information about new or deleted nodes. Thus, incremental processing is bounded to the size of gathered batches. But inserting single transactions into the pattern tree is not suitable, due to the principles of the FP-Stream algorithm. For details we refer to [Giannella *et al.*, 2003; Franke *et al.*, 2005; Franke, 2006]. Note that nevertheless the size of a batch may vary, as we do not have to collect a sudden amount of transactions to complete a batch, but may also refer to a sudden interval of time. The CDFISM method is also suited for querying time intervals that do not start at the current point in time, because it does not depend on the currently modified parts of the pattern tree. Additionally, there are no change candidates. In contrast, the method cannot detect nodes deleted from the pattern tree Advantages of both approaches, in contrast to those on separate data structures, are low runtime, easy detection of temporal changes (the TTWTs containing temporal information can be analyzed directly) and no extra memory consumption. A common disadvantage is the limited temporal quality, because it depends on the pruning steps based on the TTWTs.

After a theoretical analysis of the algorithms, we expected the actual choice of algorithm depending on the goals actually desired by the user and the characteristics of the data stream. These expectations have been substantiated by the experiences of an evaluation. These results are presented in Section 5.

4 Related Work

Optimization of building automation systems and decrease of resource consumption are of interest for politics, re-



(a) $\sigma = 0.03$, time 0,1 (b) $\sigma = 0.05$, time 6,12 (c) $\sigma = 0.03$, time 7,20 (d) $\sigma = 0.05$, time 7,20

Figure 2: Comparison of CDM and CDFISM

search and development since the last decade.

In [Garces *et al.*, 2006; Österlind *et al.*, 2007] wireless sensor networks are proposed as a feasible solution to gain additional information for building automation systems. By means of such sensor networks room status and changes can be monitored continuously. In fact, this is a precondition for a continuous optimization of user convenience and reduction of costs and resource consumption.

Tagging fault sensor data is a problem to deal with in a preprocessing step before actually analyzing the data. In [Kolokotsa *et al.*, 2005] the detection of fault sensor data is especially investigated for building energy management systems, while [Klein *et al.*, 2007] follows a more general approach for streaming or static data.

A very good survey over models and problems in the field of data stream mining provides [Babcock *et al.*, 2002]. This work lists several other contributions dealing with specific as well as general mining tasks on data streams. Change detection is not an explicit focus of this survey, but is covered by other proposals. [Chakrabarti *et al.*, 1998; Ganti *et al.*, 2001; 2002] refer to so-called "evolving" data and discuss the detection of changes in these. [Chawathe *et al.*, 1998] deals with change detection on semi-structured data. The efficient detection of bursts is discussed in [Zhu and Shasha, 2003; Kleinberg, 2003]. [Zhu and Shasha, 2003] apply a wavelet-based approach for change detection on data streams, [Kleinberg, 2003] deals with so-called "word bursts", i.e., the occurrence of frequent words. In [Aggarwal *et al.*, 2003; Aggarwal, 2003] a framework for change detection is proposed which uses spatial distance estimations and places boosted attention on heuristics to detect trends, rather than applying formal statistical models. [Kifer *et al.*, 2004] is one of the few works explicitly discussing the detection of sophisticated changes and how to present them to the user in a preferably intuitive way.

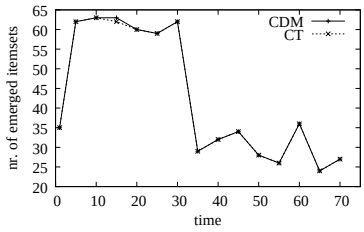
Beside the approaches developed by database researchers, there are several proposals from the machine learning community related to our work. [Widmer, 1997] deals with incremental (on-line) meta-learning and applies different classifiers in order to detect changing contexts of concepts. The guesses for concept changes based on these contexts changes are related to association rule mining. This work is based on the notion of concept drift, closely related to what we identify as changes, which occurred first in [Schlimmer and R. H. Granger, 1986]. The authors of the rather recent work [Scholz and Klinkenberg, 2007] propose boosting classifiers taking drifting concepts in account. Change detection in the context of regression is researched as well, e.g., by [Herbster and Warmuth, 1998] which deals with the problem of finding best regressors by lifting static bounds to shifting bounds. Two works covering change detection in the context of pattern analysis are [Rozsypal and Kubat, 2005] and [M.-Ch. Chena and Chang, 2005]. While [M.-Ch. Chena and Chang, 2005] focuses on the application of change detection for customer relationship management, [Rozsypal and Kubat, 2005] presents an apriori-based algorithm working on windows and blocks which we see as unqualified for our needs. Particularly, the authors aim for detecting context changes on the basis of single basic changes in the itemsets, rather than signalizing each single change, which may be basic or sophisticated.

Most of the existing works focus on detecting one specific kind of change, preferably on detecting changes in the distribution of the stream elements. Rather often they are based on statistical models and functions. The detection of changes over different time intervals or sophisticated changes in sets of elements is rather unexplored. Especially in the specific field of pattern mining we do not know about any proposals providing approaches for detecting basic and sophisticated changes, neither in parallel to the actual mining nor as a subsequent processing step. We extend existing ideas by these aspects and include time-sensitiveness, change presentation and the crucial question of *what* changed in detail (in contrast to only signalizing the pure *occurrence* of changes) into our considerations. Moreover, to the best of our knowledge, we are the first applying change detection in the context of facility management. Last but not least, our whole framework is especially designed for achieving quality- and resource-aware stream mining.

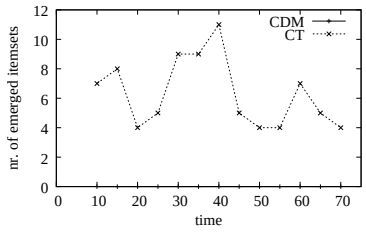
5 Evaluation

In this section we compare the three introduced approaches for incremental change detection. The following results are taken from [Fauth, 2005], which was finished under the chair of the database group at TU Ilmenau. The main purpose of this evaluation is to substantiate the advantages and disadvantages of each method, and the dependence of these on the specific requirements of the user as well as the characteristics of the data stream. Moreover, we five first directions for choosing the appropriate approach in the right situation.

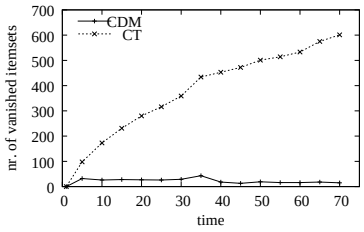
Due to the interim lack of representative facility data, we ran our tests on data generated from IBM's popular pattern generator Quest [IBM, 2005]. Currently, we are preparing the collection and processing of such data in a multifaceted project cooperation with industrial as well as scientific institutions. For details about the used data generator, we refer to [Ramesh *et al.*, 2005]. For the purpose of this evaluation, it is sufficient to know the main characteristics of the produced data streams: As proposed by [Ramesh *et al.*, 2005], the generated data follows a Poisson distribution. For representing the used data sets, we apply the same notion as generated by the IBM tool: $Tx.Iy.Dz$ symbolizes a data stream with z transactions of average size x and av-



(a) $\sigma = 0.03$, time 0,1

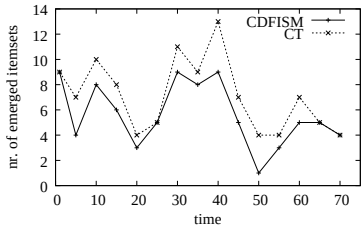


(b) $\sigma = 0.05$, time 6,12

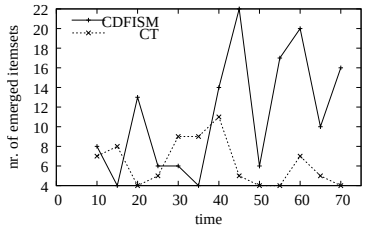


(c) $\sigma = 0.03$, time 0,1

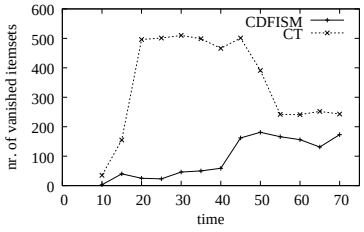
Figure 4: Comparison of CDM and CT



(a) $\sigma = 0.05$, time 0,1



(b) $\sigma = 0.05$, time 6,12



(c) $\sigma = 0.03$, time 6,12

Figure 5: Comparison of CDFISM and CT

q_s	q_e
0	1
0	10
6	12
7	20
10	30

(a) Queried times

Support σ	**Error** ϵ
0.02	0.002
	0.008
0.03	0.003
	0.012
0.04	0.004
	0.016
0.05	0.005
	0.02

(b) Values for support and error bound

Figure 3: Parameter values used in tests

erage length of maximal frequent itemsets y. For instance, $T3.I4.D1000K$ refers to a data stream containing one million transactions with 3 elements per transaction in average, the average size of the maximal frequent itemsets is 4. In the presented experiments, every five seconds one transaction is generated.

The main goals of this evaluation are to analyze:

1. the completeness of detected changes
   - with reference to different queried time intervals
   - with reference to different values for the support σ and the error bound ϵ
   - with reference to different data streams
2. the exactness with reference to false positives resp. change candidates
3. the memory consumption

We chose to query changes from time intervals $[t_s, t_e]$ shown in Figure 3(a). Interval $[0, 10]$ means we query all changes between the current point in time (0) and 10 seconds before. Figure 3(b) shows the values for σ and ϵ used in each interval. As illustrated, we switch between an ϵ value of 10% and 40% of the support value. We used generated data streams $T3.I2.D1000K$, $T3.I4.D1000K$ and $T4.I2.D1000K$. From all tests run, we present and discuss selected results which are particularly representative. We spotlight on directly comparing the introduced algorithms with respect to their ability of detecting basic changes. In the following figures, the number of emerged itemsets refers to those itemsets detected as being additional compared to prior sets. In analogy, the number of vanished itemsets refers to those itemsets the particular algorithm detected as being missing compared to prior sets.

In the first series of experiments, we compare the three approaches mutually pairwise. The following results are all gathered from runs on the $T3.I4.D1000K$ data stream using an error bound ϵ of 10%. Figure 2 illustrates differences between CDM and CDFISM. In Figure 2(a) both approaches on the pattern tree conform closely in the number of detected new itemsets. CDFISM detects slightly less itemsets, because new inserted nodes cannot be determined as new. Figure 2(b) reveals significant differences. This shows that the CDM method is only suited for detecting current changes. Analyzing past time intervals is rather poorly supported by this method. Figure 2(c) and Figure 2(d) however exemplary show close conformance for different support values when detecting dropped itemsets. In all cases, the CDM method is capable of finding a larger amount of itemsets. This is due to the possibility to recognize the deletion of nodes in the pattern tree, which is not possible using the CDFISM method.

In further tests both methods revealed to perform rather bad when querying large time intervals. This is primarily due to the summarizing character of TTWTs in the pattern tree, which also results in false positives with both methods.

In Figure 4 we compare the CDM and CT methods. Figure 4(a) shows a case where both methods behave almost identical, whereas in Figure 4(b) we illustrate a case where only the CT method is suited to continuously detect changes. Figure 4(c) shows that the number of dropped itemsets detected by the CT method increases linearly, while that of the CDM method stays constant. A case where both methods conform close in the number of detected dropped itemsets could not be found during all our tests.

In the series of comparisons, we finally illustrate the re-

lationship between CDFISM and CT in Figure 5. Again, Figure 5(a) and Figure 5(b) each show a case of good and bad conformance. Remarkable are the peaks of the CDFISM method in Figure 5(b). Figure 5(c) illustrates the capabilities of detecting dropped itemsets. Obviously, the CT method is capable of detecting more dropped itemsets, particularly in the time between 20 and 45. From time 50 upward both plots slowly approach each other again. Similar to the CDM method, we could not find any case where the CDFISM method was suited for detecting a similar amount of dropped itemsets as the CT method.

The results gained using the CT method show the improvements gained by affording more memory, which are as expected. By storing and analyzing all itemsets found so far, it is capable of detecting more changes than the methods on the pattern tree. Of course, without pruning this method could easily break resource limits. Due to the detection of changes by directly comparing (sub)sets, the CT method implicitly signalizes more changes, even subsets that were never contained in the pattern tree. Ignoring these itemsets, the CT method still detects slightly more changes than the CDM and CDFISM methods..

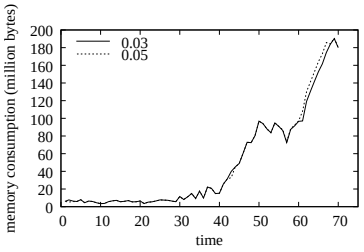


Figure 6: Memory FPStream for different σ

Beside the exposed differences in the particular capabilities of detecting different kinds of changes in different situations, the methods also differ in performance aspects and resource requirements. Figure 6 shows the memory consumption of the actual pattern tree for different support values, Figure 7 shows the same for the CT method. Note that the memory for the second approach is needed in addition to that allocated for the pattern tree. This memory requirement is almost equal for both support values, and significantly increases in the later time steps. Figure 7 reveals constantly increasing memory requirements of the CT method, depending on the support value

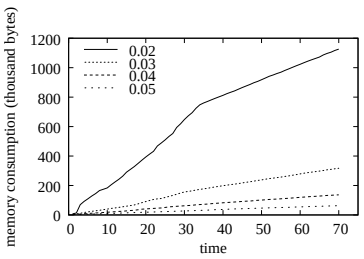


Figure 7: Memory change table for different σ

Last but no least, we compare the detection capability of all three methods on different data streams in Figure 8. These tests compare the results on the $T3.I2.D1000K$ and $T3.I4.D1000K$ data streams using a support value of $\epsilon = 0.03$ and querying time 0,1. The CDM and CT method behave very similar, whereas the CDFISM method always detects a slightly smaller amount of new itemsets.

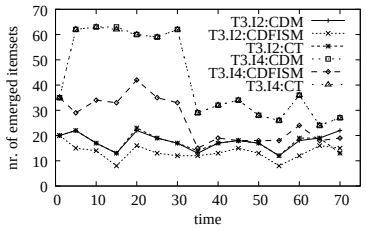


Figure 8: Varying input data

Summarizing, we can state that the CT method is suited for detecting the most changes. This goes along with high costs, especially with increasing time if no pruning technique is applied. Negative is the need for another processing step in order to eliminate wrong candidates. The methods on the pattern tree work especially good in recent time intervals, which is fine for implementing a signal and control mechanism that depends on fast reactions. All methods show no significant dependence on the actual stream characteristics or specific parameter values. Our ongoing work is to analyze the algorithms with reference to their real-time capabilities and CPU time consumption.

6 Conclusion

Facility management imposes a wide variety of challenges if it is aimed to be automated and efficient. In a setting proposed in this work, data mining approaches can be applied in an incremental manner in order to support the detection of sophisticated correlations and fast reactions. Especially for supporting fast reactions, change detection is an important field of interest. We introduced three basic methods for fulfilling this task in the context of pattern recognition. We showed that each method reveals advantages and disadvantages, and that each of the method should be preferred in specific situations. Further, we gave first directions on the essential factors influencing the optimal choice. The achieved results suggest to combine the different approaches in order to implement a reliable and flexible change detection. In future work we will extend the introduced framework for facility management and highlight further aspects. This covers other mining tasks as well as management and application challenges. Change detection will be an integral part of any such implementation. Before, we will extend the evaluation presented in this work.

Acknowledgments

We would like to thank Matthias Fauth and Conny Franke, who did great work in implementing and evaluating algorithms introduced in this work in the context of their diploma theses at the TU Ilmenau.

References

[Aggarwal *et al.*, 2003] Ch. C. Aggarwal, J. Han, J. Wang, and Ph. S. Yu. A Framework for Clustering Evolving Data Streams. In *VLDB 2003, Berlin, Germany*, pages 81–92, 2003.

[Aggarwal, 2003] Ch. C. Aggarwal. A Framework for Diagnosing Changes in Evolving Data Streams. In *SIGMOD 2003, San Diego, USA*, pages 575–586, 2003.

[Agrawal and Srikant, 1994] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*, pages 487–499, 1994.

[Agrawal *et al.*, 1993] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD 1993, Washington, D.C., USA*, pages 207–216, 1993.

[Babcock *et al.*, 2002] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of PODS 2002, Madison, USA*, pages 1–16, 2002.

[Chakrabarti *et al.*, 1998] Soumen Chakrabarti, Sunita Sarawagi, and Byron Dom. Mining Surprising Patterns Using Temporal Description Length. In *VLDB*, pages 606–617, 1998.

[Chawathe *et al.*, 1998] S. S. Chawathe, S. Abiteboul, and J. Widom. Representing and Querying Changes in Semistructured Data. In *ICDE*, pages 4–13, 1998.

[Fauth, 2005] M. Fauth. Änderungserkennung in Datenströmen, 2005. Diploma Thesis at TU Ilmenau (available in German only).

[Franke *et al.*, 2005] C. Franke, M. Hartung, M. Karnstedt, and K. Sattler. Quality-Aware Mining of Data Streams. In *IQ*, pages 300–315, 2005.

[Franke *et al.*, 2006] C. Franke, M. Karnstedt, and K. Sattler. Mining Data Streams under Dynamicly Changing Resource Constraints. In *KDML: Knowledge Discovery, Data Mining, and Machine Learning*, pages 262–269, 2006.

[Franke, 2006] C. Franke. Ressourcen-Adaptives Frequent Itemset Mining in Datenströmen, 2006. Diploma Thesis at TU Ilmenau (available in German only).

[Ganti *et al.*, 2001] V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: Mining and Monitoring Evolving Data. *IEEE Trans. Knowl. Data Eng.*, 13(1):50–63, 2001.

[Ganti *et al.*, 2002] V. Ganti, J. Gehrke, R. Ramakrishnan, and W.-Y. Loh. A Framework for Measuring Differences in Data Characteristics. *J. Comput. Syst. Sci.*, 64(3):542–578, 2002.

[Garces *et al.*, 2006] D. Garces, A. Krohn, and O. Schoch. Energy Management in Buildings with Sensor Networks. In *EWSN 2006*, 2006.

[Giannella *et al.*, 2003] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In *Workshop on Next Generation Data Mining*, 2003.

[Han *et al.*, 2000] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *SIGMOD 2000, Dallas, USA*, pages 1–12, 2000.

[Herbster and Warmuth, 1998] M. Herbster and M. K. Warmuth. Tracking the Best Regressor. In *Computational Learing Theory*, pages 24–31, 1998.

[IBM, 2005] IBM. Quest Synthetic Data Generation Code by IBM, 2005. available at www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html#assocSynData.

[Kifer *et al.*, 2004] D. Kifer, Sh. Ben-David, and J. Gehrke. Detecting Change in Data Streams. In *VLDB*, pages 180–191, 2004.

[Klein *et al.*, 2007] A. Klein, H.-H. Do, G. Hackenbroich, M. Karnstedt, and W. Lehner. Representing Data Quality for Streaming and Static Data. In *ICDE Workshop on Ambient Intelligence, Media, and Sensing (AIMS07), Istanbul, Turkey*, pages 3–10, 2007.

[Kleinberg, 2003] J. M. Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7(4):373–397, 2003.

[Kolokotsa *et al.*, 2005] D. Kolokotsa, A. Pouliezos, and G. Stavrakakis. Sensor fault detection in building energy management systems. In *5th International Conference on Technology and Automation ICTA'05*, pages 282–287, 2005.

[M.-Ch. Chena and Chang, 2005] A.-L. Chiub M.-Ch. Chena and H.-H. Chang. Mining changes in customer behavior in retail marketing. *Expert Systems with Applications*, 28(4):773–781, 2005.

[Manku and Motwani, 2002] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB 2002, Hong Kong, China*, pages 346–357, 2002.

[Motegi and Piette, 2002] N. Motegi and M. Piette. Web-based Energy Information Systems for Large Commercial Buildings. In *National Conference on Building Commissioning*, 2002.

[Österlind *et al.*, 2007] F. Österlind, E. Pramsten, D. Roberthson, J. Eriksson, N. Finne, and T. Voigt. Integrating Building Automation Systems and Wireless Sensor Networks. Technical report, SICS, 2007.

[Ramesh *et al.*, 2005] G. Ramesh, M. J. Zaki, and W. Maniatty. Distribution-Based Synthetic Database Generation Techniques for Itemset Mining. In *IDEAS*, pages 307–316, 2005.

[Rozsypal and Kubat, 2005] A. Rozsypal and M. Kubat. Association mining in time-varying domains. *Intelligent Data Analysis (IDA)*, 9(3):273–288, 2005.

[Schlimmer and R. H. Granger, 1986] J. C. Schlimmer and Jr. R. H. Granger. Incremental Learning from Noisy Data. *Machine Learning*, 1(3):317–354, 1986.

[Scholz and Klinkenberg, 2007] M. Scholz and R. Klinkenberg. Boosting Classifiers for Drifting Concepts. *Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams*, 11(1):3–28, 2007.

[Widmer, 1997] G. Widmer. Tracking Context Changes through Meta-Learning. *Machine Learning*, 27(3):259–286, 1997.

[Zhu and Shasha, 2003] Y. Zhu and D. Shasha. Efficient Elastic Burst Detection in Data Streams. In *Proceedings of SIGKDD 2003, Washington, D.C., USA*, pages 336–345, 2003.