# Incremental Stream Mining for Intelligent 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

short but meaningful!

## 1 Introduction

In modern buildings single technical objects like heating, cooling and ventilation systems are connected to a sensor network and combined by a factory master control system that in principle offers the possibility to coordinate individual components to reach a desired global state of comfort and safety. State-of-the-art controllers include functions for controlling the cooperation of different devices. However, the components are often adjusted and optimized manually. Because of the complexity of system functionalities and correlations, not only between the single objects but also between the system and external influences such as the weather, expert knowledge is required to change system settings properly. In many cases systems are run suboptimally with factory or initial settings made at installation time. This not only effects operation costs but also the environmental impact of the system. With rising resource costs and legal requirements regarding environmental issues, there is a growing potential and a need for optimization in this area. The different aspects like operating costs, the desired user comfort, operability, human sensitivity, and environmental sustainability have to be considered while planning, operating, and using building automation systems.

A comprehensive characterization of the facility management problem to be solved is beyond the scope of this paper. We focus on the optimization issue while operating a building automation system, we especially want to focus on handling information collected from sensors, actuators, and from users interacting with the system within buildings. We propose the application of data mining and knowledge discovery algorithms and techniques (i) to handle data streaming in from different facilities and buildings and (ii) to create association rules describing states and corresponding actions. Thus, the collaboration of different technical systems is characterized and may be used to optimize the entire facility management in a (semi) automatic manner. Additionally, the information obtained is useful for forecasting the consumption of resources and expected costs.

This paper is organized as follows. **[aufbau beschreiben, wenn halbwegs fertig ;-)]**

## 2 Intelligent Facility Management [Daniel/Katja/Jana/Kai]

This section gives a brief overview of the main aspects of the intelligent facility management project. At first, we outline the problem scenario that this project aims to solve. After we have elaborated on the system architecture we discuss the benefits that we gain from applying data mining techniques.

### 2.1 The Idea Behind

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 usually there is great potential for optimization in nearly all facilities. 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 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 from sunrise on it is going to be a rather warm day, then maybe heating 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 increasing 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, a building automation system consists of 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 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 or consider additional information, coherences and dependencies cannot be identified and consequently an optimal solution cannot be found.

Another problem in facility management is to reconcile the requirements of all interest groups involved. For example, users in office buildings are not interested in minimizing costs. Thus, for them as long as they feel comfortable in their offices it means no problem at all when systems work against each other (air-conditioning and heating). However, for the operating company it is very important to minimize costs (electricity, heating oil, water, etc.). Thus, heating and cooling air at the same time should strictly be avoided. Of course, ecological aspects also play an important role. Consequently, a common goal should be to reduce energy consumption whenever possible.

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)

In order to achieve these goals we need to process a variety of sensor readings and process data. By detecting correlations we can optimize operation and cooperation of installed systems while minimizing user interaction.

## 2.2 System Architecture

Our intelligent facility management system consists of a large number of sensors/actuators, a controller and the facility management system. The general system architecture is shown in Figure 1.

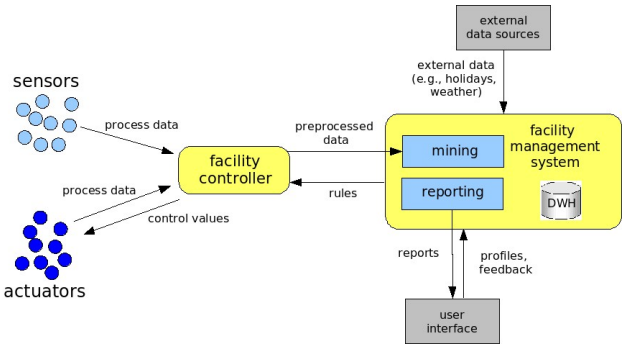



Figure 1: System architecture

As stated above, current buildings consist of different autonomous installations, like heating or air conditioning systems. In general, installations may have actuators as well as sensors. Sensors record, like temperature sensors, movement sensors, light sensors, process data like the current flow and room temperature. Actuators are mechanical devices that finally control the installations. In addition to the installation own sensors, installation independent sensors (for instance movement sensors or light sensors organized in a sensor network ) are required to get extra information about building specific parameters like air humidity, lightning, real room temperature or the number of persons within room. Both sensors and actuators produce a continuous data stream.

The facility controller collects these values, preprocess the data and forward the results to the facility management system. Within the preprocessing the controller tests the sensor and actuator process data for plausibility and notifies the facility management system about possibly defective sensors. Each building has its own facility controller. In opposition the facility management system is not building specific. Several controller can communicate (for instance via the internet) with one global facility management system. In order to reduce transmission costs, only changes measured by the sensors and actuators, detected by the facility controller, are send to the facility management system. A data item is triple $(s, v, t)$, where $s$ denotes the sensor resp. actuator id, $v$ denotes the measured value and $t$ denotes a time stamp, when the values is 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. Additionally to the output rules the knowledge management system generates reports for the different types of users. Those reports may serve as a basis for decision-making regarding comfort settings and costs as well as an information about where costs come from. Both, the high number of sensors and actuators and the continuous data stream result in a large-scale data set that is too bulky for storing it within a data base. Thus, we apply mining algorithms for association, classification and clustering directly to the data stream. Association rules can be used to detect new coherences between sensors, actuators and user preferences resp. maintenance costs. Classification strategies allow us for instance to define classes of actuators, which should not be combined (e.g., air condition systems and heating systems). We can also use the classifier to identify substitution

classes. Clustered sensors within a room possibly indicate that the sensor concentration within the room is too high.

Finally, the controller can use the rules produced by the facility management system to generate control values for the actuators.

### 2.3 Mining Tasks

In this paper, we restrict our considerations only to the aspect of stream mining. The applications of data mining in intelligent facility management are manifold. We need classificators as well as 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 use 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 raise an alarm since this could mean fire.

We assume the input to our system to adhere to the following form $[s_1, s_2, \ldots, s_l, a_1, a_2, \ldots, a_m, e_1, e_2, \ldots, e_n]$. These tuples consist of readings for $l$ sensors, $m$ actuators, and $n$ energy consumption values. These readings are used to determine frequent itemsets. Based on these itemsets a number of 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). Suddenly, 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 bind has been lowered??? – bestimmt: that blinds have been lowered.]** Consequently, the change in the frequent itemsets needs to be detected such that the association rules can be adapted as well.

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 Related Work [who ever can provide something!]

## 4 Detecting Frequent Patterns in Data Streams [Marcel]

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) 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 $\sigma \cdot W$.

In [Franke *et al.*, 2005; 2006] we presented a modified version of this algorithm, which was designed to be resource- and quality-aware in parallel. The main idea...

The focus of this paper is the application of our frequent itemset mining algorithm 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 section. Doing this, we will highlight the specialties in the context of intelligent facility management.

### 4.1 Incremental Change Detection [Marcel]

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. 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. 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). 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 intelligent 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. 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]. A further disadvantage of CDM is that we have to deal with *change candidates*, because not each modification will lead to a actual change in the itemsets. 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.

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.

# 5 Evaluation [Marcel/Fauth]

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 ...[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 [IBM, 2005]. For the purpose of this evaluation, it is sufficient to know the main characteristics of the produced data streams:

From all tests run, we present and discuss selected results which are partcularly representative .

In the first series of experiments, we compare the three approaches mutually pairwise. Figure2 illustrates differences between CDM and CDFISM.

In Figure3 we compare the CDM and CT methods.

In the series of comparisons, we finally illustrate the relationship between CDFISM and CT in Figure4.

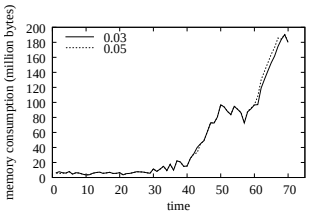



Figure 5: Memory FPStream

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. Figure5 shows the memory consumption of the actual pattern tree for different support values on the ... data stream, Figure6 shows the same for the CT method based on a separate table. Note that the memory for the second approach is needed in addition to this of the pattern tree.

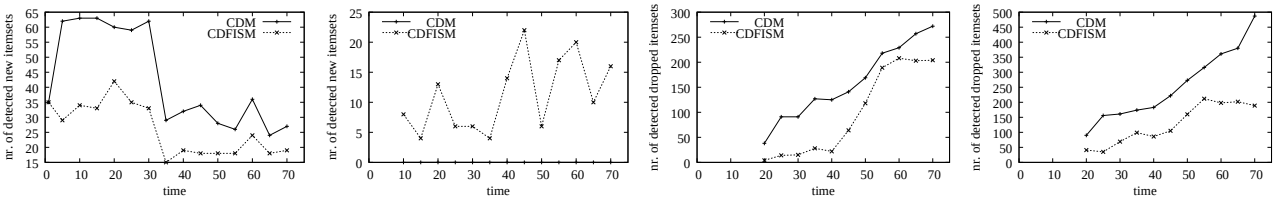
Last but no least, we compare the detection capability of all three methods on different data streams in Figure4.

Summarizing, ...

# 6 Conclusion [anyone should as well!]

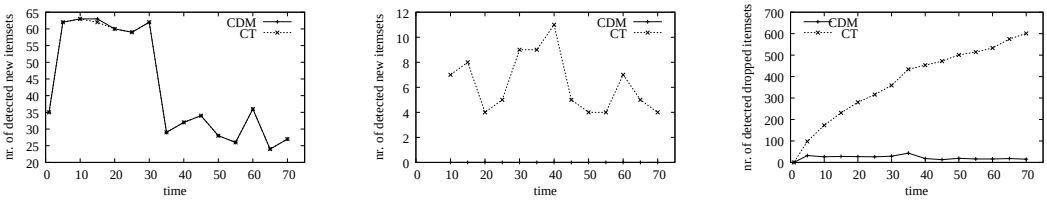
# Acknowledgments

We would like to thank Matthias Fauth, who implemented and evaluated the proposed algorithms for change detection




(a) Good conformance (b) Bad conformance (c) Good conformance (d) Bad conformance

Figure 2: Comparison of CDM and CDFISM




(a) Good conformance (b) Bad conformance (c) Bad conformance

Figure 3: Comparison of CDM and CT

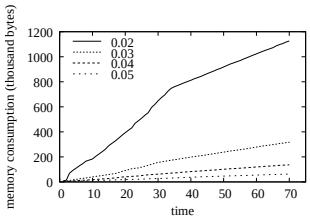



Figure 6: Memory change table

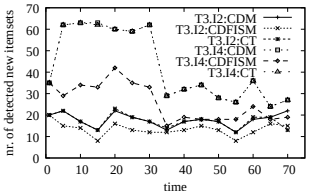



Figure 7: Varying input data

in the context of his diploma thesis at the TU Ilmenau.

## References

[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.

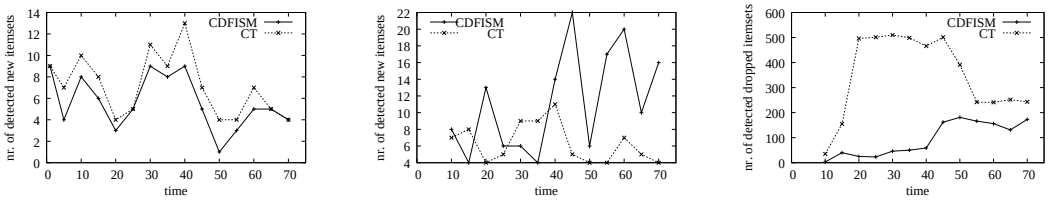
[Franke *et al.*, 2006] C. Franke, M. Karnstedt, and K. Sattler. Mining Data Streams under Dynamicly Changing Resource Constraints. In *KDML: Knowledge Discovery, Data Mining, and Machine Learning*, pages 262–269, 2006.

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

[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.

[IBM, 2005] IBM. Quest Synthetic Data Generation Code by IBM, 2005. available at ...




(a) Good conformance (b) Bad conformance (c) Bad conformance

Figure 4: Comparison of CDFISM and CT