Quality of Service-Driven Stream Mining

Marcel Karnstedt, Kai-Uwe Sattler
Department of Computer Science and Automation,
TU Ilmenau, Germany
{marcel.karnstedt; kus}@tu-ilmenau.de

Dirk Habich, Wolfgang Lehner
Database Technology Group,
TU Dresden, Germany
dbinfo@mail.inf.tu-dresden.de

Abstract

Scalable stream processing systems have to continuously manage changing resources efficiently, which is usually achieved by applying best-effort approaches on the level of processing operations. Thus, several authors have recently dealt with the problem of resource-aware stream processing, proposing methods and techniques capable of adapting to changing resources, both on the system and operator level. In this paper, we argue that Quality-of-Service (QoS) requirements are as mandatory as resource awareness when creating stream processing systems applicable in a wide and general range of data stream applications. We discuss QoS requirements and QoS-driven stream mining techniques as building blocks of the proposed framework for resource- and quality-aware stream processing systems.

1. Introduction

A growing number of application domains, i.e. traffic control or environmental surveillance, is characterized by data which is periodically or continuously produced by sensors. For the purpose of processing this data online, Data Stream Management Systems (DSMS) have been developed in the past providing operations for filtering, aggregating and combining the data as well as query languages for specifying queries declaratively. Furthermore, the analysis of streaming data, i.e., the discovery of patterns and rules, by applying data mining techniques is often an important task. Here, the combination of mining and query operators provides the necessary flexibility to implement processing pipelines.

Main challenges of processing data streams lie in the (theoretical) infinity of the streams, the potential high arrival rate, and the processing restrictions regarding CPU time and memory. Thus, there are obviously two divergent objectives: the analysis results should be exact and comprehensive, but due to the stream characteristics and the resource restrictions, the analysis has to be performed on an approximation of the entire stream, i.e., a sketch / sample or a finite subset of the stream (window). However, relying on approximations affects the quality of the analysis: the derived mining model differs from the model we would get by analyzing the entire stream.

Thus, in order to get reliable instead of best-effort results, some kind of Quality-of-Service (QoS) guarantees are needed. QoS addresses two different levels: (i) the *system level* of the DSMS with the parameters like tuple rate, CPU time, or memory consumption, and (ii) the *mining operator level* referring to the quality of the resulting mining model (e.g., SSQ for clustering). The difference between these two parameter classes is that system level parameters are usually to be guaranteed by the system, whereas the mining level parameters either depend on the system level parameters or they are given by the user. In the latter case, the values of system-level parameters have to be determined based on the specified QoS requirements. In this way, the DSMS either allocates the required resources or rejects the query.

Based on this model, we investigate in this paper the relationship between these two QoS levels. For this purpose, we discuss relevant parameters both on the system level and on the mining level, and we develop a mapping between them. This leads to a framework generalizing the idea of resource-aware and QoS-driven stream processing. We exemplify the mining level by two clustering approaches, both representing an extension of the CLUStream algorithm [1].

The remainder of this paper is structured as follows. In Section 2 we present quality measurements for mining task and the mapping between both introduced QoS levels. Afterwards, we propose two resource-aware clustering approaches with special focus on the QoS requirements and challenges in Section 3. Experiments for evaluating the proposed mining approaches can be found in Section 4. We conclude our work with related work in Section 5 and a summary in Section 6.

2. QoS-Driven Mining Operators

As available resources and queries managed by the DSMS may vary over time, operators have to be resource-aware and they should be able to dynamically adapt to changing situations, always aiming for the most efficient utilization of available resources. In the context of QoS, the opposite direction becomes equipollently important as well. Operators have to provide quality guarantees to the DSMS

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.90

under a certain amount of provided resources. The general goal is to provide both directions: *(i)* determine quality guarantees under provided resources, and *(ii)* calculate needed resources in order to achieve a provided minimum quality.

Depending on *(i)* specific QoS and resource requirements on the system level of a DSMS (memory, time, ...), *(ii)* the kind of adaptation process applied to actual operators, and *(iii)* the user's quality preferences (exact approximations, time granularity, ...), we have to handle different affected quality measures. In [4], we introduced several quality measures that apply to stream mining techniques in general. We identified two main classes of quality: one Q_M refers to the methodical quality of a mining technique, e.g., the *Sum of Square Distances (SSQ)* for clustering techniques, the other refers to time sensitiveness Q_T.

The focus of this work is to answer the question of how clustering algorithms have to interact with stream processing systems in order to achieve QoS-driven stream clustering. In such environments, changes in the output quality of an applied clustering algorithm are due to changing resources provided by the stream processing system. Thus, we will introduce different approaches for adapting dynamically to changing resources and analyze which of the quality measures are influenced and how. Figure 1 summarizes the concrete quality measures we investigate during the remainder of this paper.

Q	**Output**
Q_{Ma}	SSQ
Q_{Tr}	maximal "look back time"
Q_{Tg}	minimal granularity
Q_{Tc}	minimal time till detection of changes

Figure 1. Investigated Quality Measures

For creating scalable QoS-driven DSMS, two ways of putting resource and quality awareness into practice become mandatory: (1) claim specific quality requirements and deduce the needed resources to achieve this quality, and (2) limit the maximal memory available for processing and deduce the achievable quality.

Thus, in principle, we have to determine two (theoretical) kinds of functions:

1. $r : Q \rightarrow R$ - maps claimed quality to the resources needed, and
2. $\overline{r} : R \rightarrow Q$ - maps provided resources to the best achievable quality, as an inverse function to r.

More detailed, r is a function $r(args_x, Q_x(args_x), args_y, Q_y(args_y), \dots)$ taking all claimed quality measures $Q_x, Q_y, \dots$ and their factors as input, but we write $r(Q)$ for short. In contrast, $\overline{r}$ represents a bundle of inverse functions $\overline{r_x}, \overline{r_y}, \dots$, each corresponding to one quality measure $Q_x, Q_y, \dots$ Moreover, as we do not state the distribution of the stream elements as input factor for any function, r and $\overline{r}$ differ with different stream characteristics.

3. QoS-Driven Clustering

In this section, we introduce two clustering approaches, both applicable for achieving QoS-driven stream mining as introduced before. As a first approach, we implemented a resource-aware clustering operator on the basis of CLUStream[1] which follows the same idea of applying a statical k-means algorithm on the q micro-clusters [1].

3.1. Adapting to Changing Resources

Before introducing the implemented clustering operators, we briefly describe how the adaptation to changes in the provided resources is achieved. Both techniques utilize a filling factor

$$f := \frac{\textit{actual memory usage}}{\textit{maximal memory provided}}.$$

Reacting to changing resources now means monitoring the filling factor and reacting accordingly. As a first idea, we distinguish between three different intervals for f:

1. $f < 0.85$: memory utilization is too bad.
2. $0.85 \leq f \leq 1.0$: memory utilization is fine.
3. $f > 1.0$: overfill situation.

For each situation, an individual reaction is processed, depending on the applied clustering technique. This technique is applicable to a wide range of mining tasks, not only to clustering as described in [4].

3.2. K-Means-Based Approach

As mentioned before, in [4], we proposed a first extension of CLUStream capable of dealing with changing resource constraints. The processing of the stream and the final clustering is analog to the original CLUStream but extended by a dynamic snapshot pyramid and resource-aware features. Thus, we omit details of the actual processing here. As this first solution is also based on the usage of a statical k-means algorithm, parameters that influence the amount of memory are: k: number of end-clusters, d: dimensionality of data, $[t_s, t_e]$: time interval for calculating the clusters, T: time elapsed since the beginning of the stream when the determination of the end-clusters starts, and Q_{Ma}: quality of clusters (in relation to the SSQ achievable with maximal q/k; equals 10 in our tests).

Resource awareness is implemented by reacting to the three different states of the filling factor introduced above as follows:

1. The width of the pyramidal time frame is increased by one.
2. The width of the pyramidal time frame remains unchanged.
3. The width of the pyramidal time frame is reduced, as long as f is above 1.0.

While in the first two cases, no deletion operations need to be processed, an overfill situation as in case three causes the deletion of snapshots. Reducing the actual amount of memory includes the following operations:

1. Decrease the pyramidal time frame width by one.
2. If an order i of the pyramid has more snapshots than the given width, delete the oldest snapshots until the width is equal to the number of snapshots.
3. Check the actual filling factor: if $f > 1$, go to 1, otherwise exit.

The procedure for adjusting the width in combination with the filling factor f is processed every time a snapshot is added to the pyramidal time frame.

To summarize, we identified q/k and the width w of the pyramidal time frame as main factors of resource consumption and resulting quality. The deduced formulas for $r(Q)$ and $\overline{r}(R)$ are listed below – for details, we refer to [4].

$$r(Q) := q{\cdot}(2{\cdot}d{+}1{+}c_{LRU}){+}w{\cdot}\lceil log_\alpha(T)\rceil{\cdot}(q{\cdot}(2{\cdot}d{+}1){+}1)$$

The first formula approximates the number of floating point numbers (#FPN) needed in order to achieve the desired mining quality Q. This can easily be mapped to actual memory consumption in number of bytes. Q is represented by the values of q and w, d and T describe the current characteristics of the data stream, and α as well as c_{LRU} are constants provided by the CLUStream algorithm. Note that k influences the mining quality but has no direct impact on the memory consumption of the algorithm's online component. However, in order to achieve a certain cluster quality, the ratio q/k must be kept in corresponding intervals, which reflects q again.

$$\overline{r}(R) := w = \frac{R - q \cdot (2 \cdot d + 1 + c_{LRU})}{\lceil log_\alpha(T)\rceil \cdot (q \cdot (2 \cdot d + 1) + 1)}$$

The proposed algorithm adapts to changing resource limits R (again, in #FPN) by adjusting w, and thus the snapshot pyramid, accordingly. Hence, the formula for determining the achievable cluster quality computes the maximal w applicable under a provided resource limit R. On this basis, the following heuristic is applied: by varying the number of micro-clusters q from $10 \cdot k$ down to k, the corresponding average width w of the pyramidal time frame is calculated. If w is above 2 for a good choice of q ($3 \cdot k$ to $10 \cdot k$), acceptable clustering quality can be achieved. Otherwise, w or q/k are too small to guarantee near-optimal results. The user specifies QoS requirements by providing a certain quality level, where each level is mapped to a specific combination of quality parameters q/k and w – see Section 4 for a corresponding list of quality levels. The heuristic is used in order to ensure QoS guarantees during stream processing. The quality mapping is analogously used for approximating resource requirements based on a user-specified quality level as well as the formula for $r(Q)$ above.

3.3. Density-Based Approach

To detect cluster formations of arbitrary shape (spherical, drawn-out, linear, elongated etc.), we developed a temporal-density-based stream clustering approach.

Temporal CF Tree Our temporal-density-based stream clustering algorithm is a temporal extension of DBSCAN [3]. First, we construct a temporal CF tree (TCF tree) to compress the data stream elements according to their spatial as well as temporal closeness. The TCF tree is based on the concept of the already introduced micro-clusters [1] by Aggarwal et al. In contrast to the well-known CF vector of the CF tree, the micro-clusters store summarizing temporal information on the data stream elements. The CF Additivity Theorem [10] is also valid for the TCF vector.

Thus, our TCF tree is a height-balanced tree similar to the CF tree with the following three parameters: (i) branching factor B, (ii) spatial threshold S and (iii) temporal threshold T. The temporal threshold T is an additional parameter compared to the CF tree, which is absolutely necessary in the data stream environment. The leaf and non-leaf nodes are similar to the CF tree but all entries in a leaf node have to satisfy the following conditions: (i) the spatial threshold requirement with respect to the spatial threshold S: the spatial diameter or radius has to be smaller than S, and (ii) the temporal threshold requirement with respect to the temporal threshold T: the temporal standard deviation has to be smaller than T. The TCF tree will be built up dynamically as new data stream elements arrive. The insertion algorithm of the CF tree is slightly adapted [8].

Temporal-Density-Based Clustering Since not every leaf node corresponds to a temporally and spatially dense cluster, we require a clustering algorithm determining the final temporally and spatially dense clusters from the leaf entries. The leaf entries of the TCF tree represent subclusters, where the data stream elements are grouped together according to the spatial as well as the temporal closeness of the streaming elements with regard to the threshold parameters S and T. The set of subclusters $SC = \{s_1, \ldots, s_l\}$ is now the basis for the determination of the final clusters of the data stream using a temporal extension of DBSCAN [3].

For each subcluster s_i, the centroid $\mu_i^{spatial}$, the radius $r_i^{spatial}$, the temporal centroid $\mu_i^{temporal}$, the temporal deviation $\sigma_i^{temporal}$ and the number n_i of absorbed original data stream elements are known or can be computed from the corresponding TCF vector (micro-cluster). Furthermore, we know that each radius $r_i^{spatial}$ is equal to or smaller than the spatial threshold S. Moreover, we know that each subcluster s_i represents data stream elements, whose number is equal to or smaller than an upper boundary. This upper boundary is defined by the temporal threshold T of the TCF tree. This observation implies that a temporally and spatially dense region in the data stream chunk is represented by a set of overlapping subclusters. Another reason for high overlapping of subclusters is that in a data stream, a spatially dense

region can occur at different points in time. In this case, the same spatial subclusters can exist but with different time information. From this observation, we can determine the final temporally and spatially dense clusters with a temporal extension of DBSCAN. According to DBSCAN [3], we can define core subclusters as follows:

Definition 1 (core subcluster) A subcluster $s_i \in SC$ is a core subcluster when $n_i \geq MinPts$, where n_i is the number of absorbed data stream elements in the subcluster s_i.

A core subcluster results only from the number of absorbed data stream elements. Therefore, we can use the $MinPts$ parameter to determine the subclusters to become core subclusters. For each subcluster s_i, we know the number of absorbed data stream elements n_i. A good heuristic for $MinPts$ is:

$$MinPts = \frac{\sum_{i=1}^{l} n_i}{l},$$

where l is the number of subclusters. The advantage of this heuristic is that the $MinPts$ parameter is adjusted to the result of each TCF tree. The $MinPts$ parameter may be different for different data stream stages. Next, we extend the definition of density reachability, direct density reachability and density connection [3] so that the temporal relationship can be regarded.

Definition 2 (direct time-density reachability) A subcluster $s_i \in SC$ is directly time-density-reachable from a subcluster $s_j \in SC$ if

1. $\mu_i^{spatial} \in N_S(\mu_j^{spatial})$,
2. s_j is a core subcluster, and
3. $|\mu_i^{temporal} - \mu_j^{spatial}| \leq maxTimeGap$

In other words, a subcluster s_i is directly time-density-reachable if the centroid $\mu_i^{spatial}$ lies in the S-neighborhood ($N_S()$) of the subcluster s_j with centroid $\mu_j^{spatial}$ if s_j is a core subcluster, and if the time distance between the subclusters is equal to or smaller than an application-specific $maxTimeGap$. We use the S-neighborhood instead of the r_j-neighborhood because of the expected high overlapping and to establish a parameter linkage with the TCF tree. The TCF tree parameter determines the spatial density.

The *time-density reachability* and the *time-density connection* are the transitive extensions of the direct time-density reachability. Now, we can define our time-, space- and density-based notion of a cluster. A cluster is defined to be a set of time-density-connected subclusters which is maximal regarding the time-density reachability. Noise will be defined relative to a given set of clusters. Subclusters which do not belong to any clusters are considered noise.

Definition 5 (temporally and spatially dense clusters) Let SC be a set of subclusters. A temporally and spatially dense cluster C with regard to S, $MinPts$, $maxTimeGap$ is a non-empty subset of SC satisfying the following conditions:

1. $\forall s_i, s_j$: if $s_i \in C$ and s_j is time-density-reachable from s_i regarding S, $MinPts$, $maxTimeGap$, then $s_j \in C$ (Maximality).

2. $\forall s_i, s_j \in C$: s_i is time-density-connected to s_j regarding S, $MinPts$, $maxTimeGap$ (Connectivity).

The output of the clustering algorithm is a set of clusters $C = \{C_1, \ldots, C_m\}$, where the number of clusters may be different. Each cluster C_i is represented by a set of subclusters $C_i = \{sc_i^1, \ldots, sc_i^{k_i}\}$, where k_i is the number of subclusters for C_i.

QoS-Driven Processing Next, we discuss the processing of data streams in combination with QoS considerations in more detail. During the data stream processing, one TCF tree is kept in memory as a compact representation of the data stream elements. The final clustering results can be derived from this tree at any time with our presented algorithm. Thus, parameters that influence the amount of memory are the parameters of the TCF tree and data parameters: the branching factor B of the TCF tree, the spatial threshold S, the temporal threshold T and the dimensionality d of data.

Resource awareness is ensured by reacting to the three different stages of the filling factor f introduced in Section 3.1.

1. In case the memory utilization is too bad, we adjust the temporal and/or the spatial threshold. At the moment, we increase each parameter slightly until f is in the right interval.

2. If the memory utilization is fine, we do nothing.

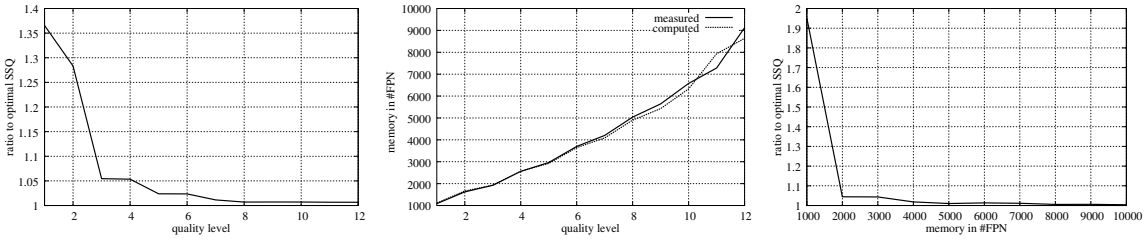
3. If there is an overfill situation, three different strategies are possible, which are described next.

The most important situation for a reaction is the overfill stage, where the filling factor f is larger than 1. In this case, the size of the TCF tree has to be resized in order to collect new data stream elements. From our point of view, three different reaction strategies arise.

1. *TCF tree compression* with ajusted spatial and temporal thresholds which nfluences the approximate quality of the clustering result Q_{Ma}. Therefore, it can only be applied as long as the user-specified quality is not violated.

2. *Delete some entries from TCF tree* which influences the time range quality metric Q_{Tr} because we reduce the captured time horizon of the data stream in the TCF tree.

3. *Store TCF tree on disk and create a completely new TCF tree.* In this case, the quality Q_{Tc} will decrease because disk access is much more expensive than memory access.



(a) SSQ for Applied Quality Levels (b) Memory Consumption (c) Memory Effects on Quality

Figure 2. Quality of Service in the k-Means-Based Approach

The necessary values for $r(Q)$ and $\overline{r}(R)$ can be approximated with a straightforward heuristic. Currently, we are developing a concrete formula for computing the size of the TCF tree. The heuristics applied on this formula follow the idea of the heuristics used in the k-means approach with an extension to respect the tree structure.

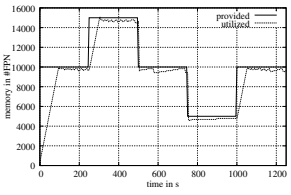
4. Evaluation

In this section, we evaluate main quality aspects of the proposed mining operators and the provided resource and quality awareness as main QoS requirements. The evaluation of the k-means-based approach is extended compared to prior works, whereas the density-based technique is still in the final development phase. Thus, this method is evaluated rather preliminary.

The k-means-based approach was evaluated on generated 2-dimensional data sets. We defined 5 cluster centers and a randomized procedure generated 25,000 points following a Gaussian distribution around these cluster centers. Additionally, in order to simulate evolving streams, we changed the centers in mean and variance every 5,000th point. Each test was repeated 5 times on such generated data and average values were measured. Internal parameters were set to $\alpha = 2$, $k = 5$, initPoints=1,000, snapshot saving interval=1s and input rate=200 points/s. In order to evaluate the resulting mining quality of the stream algorithm, a static clustering (LBGU) was run in order to compute a reference result and the corresponding SSQ SSQ_R. In the following figures, Q_{M_a} is represented by the ratio SSQ_S/SSQ_R, where SSQ_S stands for the SSQ resulting from the clusters identified during stream processing.

In a first series of tests, we wanted to show the method's resource awareness by proving its ability to dynamically adjust to changing resources. We set $q = 20$, started the processing, and changed the provided resource limit every 250 seconds. As Figure 3(a) reveals, the algorithm perfectly adapts to meet the current memory limit provided, whether it is raised or lowered. The memory utilization is always at the upper limit and changed limits are met quickly and accurately.

The second series of tests was run in order to analyze the QoS aspects introduced in this work. We use several levels



(a) Resource Adaptation

w / q/k	2	2.5	3	3.5	4	4.5	5
2	1						
3	2	3					
4		4	5				
5			6	7			
6				8	9		
7					10		
8						11	12

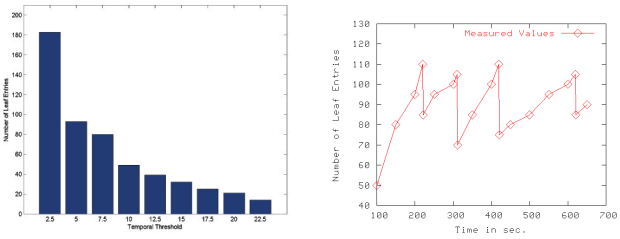
(b) Applied Levels of Quality

Figure 3. Evaluation of k-Means Approach

of quality, which are mapped to a specific combination of quality parameters q/k and w. The mapping is illustrated in Figure 3(b). These levels are used according to the heuristic responsible for meeting QoS requirements introduced in Section 3.2.

In these tests, the interval from points 11,000-14,000 was clustered after the arrival of all 25,000 points. Figure 2(a) illustrates the exact SSQ ratios the different quality levels result in. Figure 2(b) shows the almost linear correlation between memory consumption and quality level. Moreover, the figure also shows the plot for the computed $r(Q)$ according to the formula introduced in Section 3.2. The approximation is satisfyingly accurate, which represents a mandatory prerequisite for providing QoS guarantees to the system and/or the user. The correlation between provided memory and resulting mining quality is shown in Figure 2(c). The algorithm was run under particular memory limits, and the heuristic to adjust the internal quality parameters during stream processing was applied. As expected, higher memory limits result in higher mining quality. It is striking that a very low memory limit results in very bad quality, but there is an early point from which on all results are in an acceptable range. Very high memory limits do not affect the output quality significantly; rather, the temporal qualities are affected because fewer snapshots are deleted from the pyramid. The plot for $\overline{r}(R)$ cannot be shown here, as the formula from Section 3.2 computes only w and is used in the introduced heuristic.

Due to the current early phase of our time-density-based stream clustering approach regarding resource awareness, we are only able to show some preliminary evaluation results. Figure 4 (a) illustrates the number of leaf entries for



(a) Number of Leaf Entries for Different Temporal Thresholds

(b) Adaptation with Delete and Merge Operation

Figure 4. Evaluation of Time-Density-Based Clustering Algorithm

a data stream chunk against different temporal thresholds. The number of leaf entries is an indicator for the size of the TCF tree. The branching factor B and the spatial threshold S were fixed in the experiments. The number of all nodes can be approximated from the number of leaf nodes and the branching factor. As we can see, the larger the temporal threshold, the smaller the number of leaf entries (subclusters). With a large temporal threshold, a leaf entry can absorb more data stream elements in opposition to a small temporal threshold. The effect of the spatial threshold is identical and is already presented in [3]. This experiment indicates that the TCF tree can be reduced in size when the temporal and/or spatial threshold is increased.

In the second experiment, we evaluated our *delete* and *merge* operations as adaptation strategy. In this experiment, we set the maximal size of the TCF to 100 leaf nodes. If the actual number of leaf entries during the stream processing exceeded this maximal size, we deleted the oldest leaf entries until the number of leaf nodes had been reduced to 80% of the maximal size. To avoid degeneration, we then conducted our merge operation on the reduced TCF tree. As we can see in Figure 4(b), we are able to adjust the size of the TCF tree around the maximal size and thus, we can suitably react to resource limitations.

5. Related Work

Mining data streams has been an active research area for more than five years [7, 1]. Recently, the idea of processing streams while adapting to resource and quality constraints has gained more attention. Several recent works deal with quality-aware online query processing applications in centralized and distributed systems, e.g., [2], but only few discuss concrete approaches and algorithms in stream mining scenarios, let alone the conjunction of resource adaptiveness and quality awareness.

In [5], the authors propose a resource-adaptive mining approach based on similar goals as our work. They suggest the adaptation to memory requirements, time constraints and data stream rate by solely adapting the produced output granularity. The paper also focuses on clustering but the authors do not consider quality aspects explicitly. In succeeding works, e.g., [6], they extend their approach by adding resource adaptiveness to the input rate and the actual data mining algorithm as well. However, the approach still lacks providing quality guarantees for the mining results.

As another example for resource awareness in stream mining, the algorithm RAM-DS proposed in [9] uses a wavelet-based approach to control the resource requirements. Although the proposed algorithm for mining temporal patterns is resource-aware, it is not resource-adaptive and does not provide guarantees for the quality of the mining results.

6. Conclusion

In this paper, we show that quality-of-service requirements are as essential as resource awareness. Those QoS requirements are mandatory to get reliable and not just best-effort results. Based on a proposed QoS model, we investigate the relationship between different QoS levels. For this purpose, we discuss relevant parameters for the various QoS levels and develop a mapping between them. In our description, we focus on resource-aware and QoS-driven data stream mining techniques, in particular, on data stream clustering.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Clustering Evolving Data Streams. In *VLDB*, 2003.

[2] L. Berti-Équille. Contributions to Quality-Aware Online Query Processing. *IEEE Data Eng. Bull.*, 29(2):32–42, 2006.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, 1996.

[4] C. Franke, M. Hartung, M. Karnstedt, and K. Sattler. Quality-Aware Mining of Data Streams. In *IQ*, 2005.

[5] M. M. Gaber, S. Krishnaswamy, and A. Zaslavsky. Adaptive mining techniques for data streams using algorithm output granularity. In *The Australasian Data Mining Workshop*, 2003.

[6] M. M. Gaber and P. S. Yu. A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. In *SAC*, 2006.

[7] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[8] M. Karnstedt, K.-U. Sattler, D. Habich, and W. Lehner. Quality of service-driven stream mining (available under http://wwwdb.inf.tu-dresden.de/papers/kshl07.pdf), 2007.

[9] W.-G. Teng, M.-S. Chen, and P. S. Yu. Resource-aware mining with variable granularities in data streams. In *SDM 2004*.

[10] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD 1996, Montreal, Canada*, pages 103–114, 1996.