Quality and Performance Optimization of Sensor Data Stream Processing

Anja Klein
SAP Research Center Dresden, SAP AG
Chemnitzer Str. 48
01187 Dresden, Germany
anja.klein@sap.com

Wolfgang Lehner
Database Technology Group, TU Dresden
Helmholtzstrae 10
01069 Dresden, Germany
wolfgang.lehner@tu-dresden.de

Abstract—**Intelligent sensor devices together with data stream management systems allow the automatic recording and processing of huge data volumes to guide any kind of process control or business decision. However, a crucial problem is posed by data quality deficiencies due to imprecise sensors, environmental influences, transfer failures, etc. If not handled carefully, they misguide decisions and lead to inappropriate reactions. In this paper, we present the quality-driven optimization of stream processing that improves the resulting quality of data and service. After an introduction to data quality management in data streams, we define the targeted optimization problem comprising the optimization objectives and parameters that configure the required stream processing operators. Based on the generic optimization framework, we discuss and evaluate the optimization execution in batch and continuous mode. Further, we shed light on the crucial definition of the stream partition length used for the optimization that significantly influences the optimization performance. Finally, we provide a detailed validation of the proposed optimization strategies as well as the scalability of the overall approach not only at artificial data streams, but also using the real-world example of contact lens production monitoring.**

Keywords - Data Quality; Data Stream Processing; Query Optimization; Heuristic Optimization

I. Introduction

Data stream management systems have been developed to process continuous data flows of high data rate and volume. For example, turnover values or sales volume may be streamed from distributed affiliations to the central controlling department to derive management strategies. Further, data streams are recorded in sensors networks to control manufacturing processes or maintenance activities. In this paper we illustrate the quality control in contact lens manufacturing, where lens thickness and axial difference are measured to derive a quality indicator for the production line.

In most applications data stream systems encounter restricted resources, such as limited memory capacity, data transfer capability and computational power. To meet these constraints, data stream volume has to be reduced by processing the streamed information. Data reduction always goes along with a loss of information. Data processing results such as aggregations can only be approximated, so that answers are incorrect or incomplete with respect to the true outcome. Moreover, most data stream sources suffer from limited data quality in multiple dimensions from the beginning. The correctness is decreased for example by restricted sensor precisions, by typos in text elements, or by RFID readers failing to scan an item properly. The completeness of a data stream is reduced whenever a true world event is missed due to sensor or system malfunction. Information and decisions derived from such falsified sensor data are likely to be faulty, too. Therefore, data quality problems have to be handled carefully.

Data quality information, that describe data quality deficiencies due to data sources and/or data processing, can be recorded and transfered in the data stream, e.g., by the quality propagation model (QPM) presented in [1]. It provides a comprehensive framework for data quality management in streaming environments. Only then, data quality information are provided to the user to enable the comprehensive evaluation of imprecise and/or incomplete data stream processing results. Faulty information can be detected to prevent from incorrect decisions. Furthermore, quality information may identify processed data stream results as too insecure to derive any suitable decision. In that case, the data quality has to be improved to re-enable the confident decision-making.

This paper details and extends the quality-driven optimization of sensor data stream processing to improve the resulting data quality, while complying to the given system constraints, that we first published in [2]. Based on data quality information provided in the stream, the data stream operators are configured to maximize the quality outcome to meet user-defined quality requirements. The online tuning is performed continuously in parallel to the traditional data stream processing. The optimal operator configuration is adapted to varying data stream characteristics and changing user-defined requirements on resulting data quality.

For the field of data quality improvement and optimization our contributions are as follows.

- We present the definition of streaming data quality and discuss the conflicts between the embraced data quality dimensions. Further, we identify candidates for the data quality improvement out of a comprehensive set of data stream operators. We discuss derived parameters and their impact on the data quality-driven improvement of the stream processing in Section II.

- In Section III, we propose the optimization framework that adapts the stream processing for data quality improvement. We discuss the optimization execution in batch and continuous mode that supports the determination of the applied stream partition length. Further, we present the data quality improvement task as multi-objective optimization problem and range it in the traditional operations research classification.
- We discuss the specific components of the heuristic optimization algorithm Quality-driven evolution strategy (QES) to solve the data quality optimization problem in Section IV.
- We present a comprehensive evaluation of the presented algorithm in Section V. We analyze the practicability of the proposed optimization framework, validate the influences of different optimization strategies, compare the performance of QES to further optimization heuristics and evaluate the capability at the example of the contact lens production control.

We complete this paper with a discussion of related work in the field of data stream quality, quality improvement and optimization in Section VI and concluding remarks in Section VII.

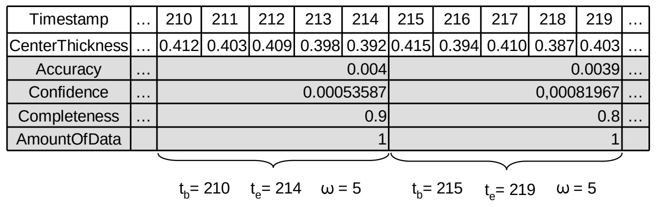
II. Problem Analysis

In this section, we first define the data quality in data streams to derive objectives for the data-quality driven optimization, which are then discussed in detail. Afterwards, the candidates for data quality improvement are described. First, the sampling rate configuration is illustrated, followed by the interpolation configuration. Then, the configuration impact of group size for aggregation, frequency analysis and filtering as well as of data quality window size is depicted.

A. Data Quality in Data Streams

A data stream comprises a continuous stream of m tuples τ, consisting of n attribute values $A_i (1 \leq i \leq n)$ and the represented time interval $[t_b, t_e]$. To allow for the efficient management of data quality in data streams, we adopt the data quality window approach introduced in [1]. DQ information is not forwarded for each single data item, but aggregated over ω_i data items independent for each stream attribute A_i. The stream is partitioned into κ_i consecutive, non-overlapping jumping data quality windows $w(k)$ $(1 \leq k \leq \kappa_i)$, each of which is identified by its starting point tw_b, its end point tw_e, the window size ω_i and the corresponding attribute A_i as illustrated in Figure 1. Beyond the data stream items $x(j)(tw_b \leq j \leq tw_e)$, the window contains $|Q|$ data quality information q_w, each obtained by averaging the tuple-wise DQ information over the window.

Furthermore, the executed data processing steps have to be tracked in the quality propagation model (QPM) to not lose data quality information. When data streams are aggregated or joined, their data quality information have to be summarized, too. Only then, the data quality path through the processing can be monitored and data quality deficiencies introduced during data stream processing are captured.



Timestamp	…	210	211	212	213	214	215	216	217	218	219	…
CenterThickness	…	0.412	0.403	0.409	0.398	0.392	0.415	0.394	0.410	0.387	0.403	…
Accuracy	…					0.004					0.0039	…
Confidence	…					0.00053587					0,00081967	…
Completeness	…					0.9					0.8	…
AmountOfData						1					1	

t_b= 210 t_e= 214 ω = 5
t_b= 215 t_e= 219 ω = 5

Fig. 1. Data stream sample

The probably best-known/most referenced, comprehensive and balanced definition of data quality was presented 1996 by Wang et al. [3]. They distinguish four data quality categories of 15 data quality dimensions incorporating data quality aspects of raw data, data sets as well as user requirements. We define data quality in data streams based on their findings on intrinsic (accuracy, believability, objectivity, reputation) and contextual data quality dimensions (value-added, relevancy, timeliness, completeness, amount of data). Due to high data stream volume and rate, a manual evaluation of data quality is not possible, so that subjective dimensions of representation and accessibility cannot be measured to describe data streams.

While the accuracy describes the systematic error of data stream values resulting from deficiencies in data sources, the believability in context of data streams can be equated with the confidence describing random errors produced by unforeseeable influences (e.g., environmental affects in sensor networks). Objectivity and reputation of automatic data stream sources can be assumed as guaranteed and do not have to be monitored. As value-added and relevancy are aspects subjective to the respective user, the contextual dimensions reduce to timeliness or up-to-dateness, completeness defining the rate of originally measured stream items compared to interpolated ones and the amount of data d of raw data items represented by an aggregation result.

Definition: *The data quality Q of a data stream D is defined by the set of five data quality dimensions: accuracy a, confidence ϵ, completeness c, amount of data d and timeliness u.*

As stated in the introduction, the data quality-driven optimization configures the data stream processing to maximize the above defined stream data quality while guaranteeing the compliance to the restricted system resources. As metric for the resource load, we use the data stream volume V. The less volume is needed, the better the constraints are met. Hence, the minimization of the data volume is admitted to the optimization goals. Finally, from the user's point of view a high data volume also has positive effects. The more details are given in the data stream, the better decisions can be derived. To measure this data granularity, we select the timeframe T represented by one data stream tuple. While raw data depict one point in time, the result of a data stream aggregation represents a larger time interval. The wider the timeframe, the lower is the granularity. To support the detailed evaluation of streaming data the granularity has to be maximized.

Operator	Parameter	a	ϵ	c	d	u	V	T
Projection	—							
Selection	—							
Join	—							
Aggregation	GroupSize l				+		-	-
Sampling	Rate r_{sa}		-	-		+	+	
Frequ. an.	GroupSize l				+		-	-
Filter	GroupSize l							
Algebra	—							
Threshold	—							
	WindowSize ω					+	-	(-)

Accuracy a Completeness c Stream volume V
Confidence ϵ Amount of data d Granularity T
Timeliness u

TABLE I
OPTIMIZATION CANDIDATES FOR QUALITY IMPROVEMENT

Table I summarizes the optimization objectives composed of data quality dimensions, data stream volume, and granularity. To identify candidates for the data quality improvement, we analyze the operator repository of the QPM consisting of traditional data stream operators like join and selection, operators of the signal analysis, which are often applied to sensor data streams, and operators of the numerical algebra like addition or division as well as the threshold control. Besides the operator configurations, we present the size of the jumping data quality windows ω as interesting parameter. Table I shows the impact of a parameter increase: the quality values are either increased (+) or decreased (-).

B. Objectives

This section defines the fitness functions for each objective of the data quality-driven optimization. As the accuracy describes defects or imprecisions of data stream sources, it cannot be improved by any operator configuration. This data quality can be removed from the list of optimization objectives.

The objectives determined above span different value domains. For example, the window completeness constitutes values in the range $0 \leq c_w \leq 1$, while the absolute statistical error in the dimension confidence is unlimited $0 \leq \epsilon_w \leq \infty$. To allow the quantitative comparison of different objectives, we normalize the objective functions to the range $[0, 1]$.

1) Confidence: The confidence illustrates the statistical error ϵ due to random environmental interferences (e.g., vibrations, shocks) defining the interval $[v-\epsilon; v+\epsilon]$ around the data stream value v containing the true value with the confidence probability p. ϵ is defined by the $(1-p/2)$-quantile α and the data varianceσ^2 of each data quality window w. For example, for $p = 99\%$ the initial confidence of a data quality window including the lens thickness measurements

$$\{0.396mm, 0.428mm, 0.412mm, 0.379mm, 0.403mm\} \quad (1)$$

is set to $\epsilon_w = \alpha \cdot \sigma = 2.58 \cdot 0,000286mm = 0,0007224mm$.

The average statistical error over all data stream attributes has to be minimized to maximize the data quality confidence. The objective function is normalized by division with the maximal statistical confidence error ϵ_{max} in the stream. The objective f_ϵ is defined as follows.

$$f_\epsilon \quad : \quad min \qquad \frac{1}{n \cdot \epsilon_{max}} \sum_{i=1}^{n} \frac{1}{\kappa_i} \sum_{k=1}^{\kappa_i} \epsilon_w(k) \qquad (2)$$

2) Completeness: The completeness addresses the problem of missing values due to stream source failures or malfunctions. Multiple estimation or interpolation strategies exist to deal with missing values in ETL processes and data cleansing [4]. We apply the linear interpolation as compromise between the quality of value estimation and computational capacity. The data quality dimension completeness c is accordingly stated as the ratio of originally measured, not interpolated values compared to the size of the analyzed data quality window.

For example, the sensor for axial difference misses the lens at timestamp $t =' 237'$. To nevertheless derive the quality indicator, the missing value is computed as follows.

$$ax('237') = \frac{1}{2} \cdot (ax('236') + ax('238')) \qquad (3)$$

To note the sensor failure, the completeness of the data quality window $['230','239']$ is set to $c_w = 0.9$.

To conform with the objective above, the objective of maximal completeness is transformed to the minimizing problem f_c, which minimizes the ratio of interpolated data items. Here, no normalization is required as the domain $[0, 1]$ is already provided by the completeness definition.

$$f_c \quad : \quad min \qquad \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\kappa_i} \sum_{k=1}^{\kappa_i} (1 - c_w(k)) \qquad (4)$$

3) Amount of Data: The amount of data determines the set of raw data x used to derive a data stream tuple $y = f(x)$. The higher the amount of data, the more reliable is the processed information. To eliminate outliers to derive statistically stable information of the production line quality, the quality indicator is averaged over a certain set of contact lenses. Thereby, the aggregation group size $l = 20$ leading to $d = 20$ produces more reliable results than $l = 2 (d = 2)$.

To transform the objective of maximal amount of data to a minimization problem, we calculate the difference to the highest possible amount of data $d = m$, that comprises the complete data stream. The maximum m serves at the same time as normalization weight.

$$f_d \quad : \quad min \qquad \frac{1}{n \cdot m} \sum_{i=1}^{n} \frac{1}{\kappa_i} \sum_{k=1}^{\kappa_i} (m - d_w(k)) \qquad (5)$$

4) Timeliness: The timeliness is defined as difference between tuple timestamp $t(j)(1 \leq j \leq m)$ and current system time $clock$. For example, the timeliness of $ax('13:26:54')$ at $'13:28:10'$ is $u = 76s$. To maximize the data quality dimension timeliness, the average tuple age normalized by the maximum age u_{max} has to be minimized.

$$f_u : \quad min \qquad \frac{1}{m \cdot u_{max}} \sum_{j=1}^{m} u(j) \qquad (6)$$

$$= \qquad \frac{1}{clock - t_{min}} \cdot \left[clock - \frac{1}{m} \sum_{j=1}^{m} t(j) \right] \qquad (7)$$

5) Data Stream Volume: The data stream volume V defines the number of transfered data values in n stream attributes over m data tuples. Besides, the transfered data quality information have to be incorporated. The additional volume is computed based on the number of transfered data quality dimensions Q_i per attribute A_i and the average data quality window size $\bar{\omega}_i$.

$$V \quad = \quad m \cdot (n+1) + \sum_{i=1}^{n} \frac{m}{\bar{\omega}_i} \cdot |Q_i| \qquad (8)$$

The average volume of a data stream tuple of the contact lens stream described with $|Q_i| = 4$ dimensions and an average data quality window size of $\bar{\omega}_i = 20$ results in $V = (4+1) + 1/20 \cdot 4 = 5.2$. To normalize the stream volume to the data range $[0,1]$, we refer to the maximal stream length $m_{max} = r_{max}/r \cdot m$ determined by the current stream rate r and the maximal manageable rate r_{max}(e.g., $r_{max} = 1/ms$). The maximal data volume further depends on the maximal data quality window size $\omega = 1$, such that

$$V_{max} \quad = \quad m_{max} \cdot (n+1) + m_{max} \cdot \sum_{i=1}^{n} |Q_i|. \qquad (9)$$

To minimize the costs for data stream transfer and processing, the normalized data stream volume has to be minimized.

$$f_V \quad : \quad min \qquad \frac{V}{V_{max}} \qquad (10)$$

6) Granularity: The data stream granularity T is measured as the average timeframe $[t_e - t_b]$ of all data stream tuples. For example, the timeframe of the averaged quality indicator with $l = 20$ constitutes in $T = 20s$. For raw data items describing one point in time, the granularity equals 0, as $t_e = t_b$. To maximize the granularity, the average timeframe normalized by its maximum T_{max} has to be minimized.

$$f_T \quad : \quad min \qquad \frac{1}{m \cdot T_{max}} \sum_{j=1}^{m} t_e(j) - t_b(j) \qquad (11)$$

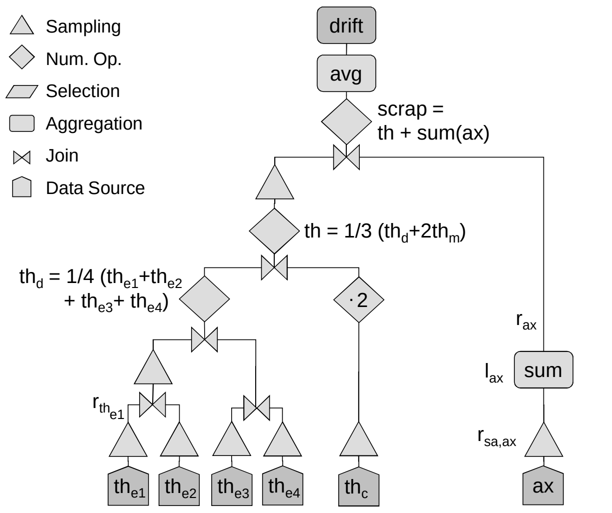


Fig. 2. Processing tree

C. Configuration Parameters

The analysis of typical data stream operators identified sampling, aggregation and frequency analysis as configuration candidates for the data quality-driven optimization. This section first defines the dimension of the optimization problem domain. Afterwards, the impact of the configuration parameters of the identified candidates are discussed.

Figure 2 shows the processing graph of the contact lens application. The goal is to monitor the overall production quality, i.e., the fraction of contact lenses that have to be removed from the production line as scrap, to predict the next maintenance date. Therefore, four thickness measurements th_e at the lens edge are summarized and added to the weighted center thickness th_c. All sensor streams are sampled up $r_{sa} > 1$ or down $r_{sa} < 1$ to reduce the overall data volume and adjust the data stream rates in case of missing data tuples. Then, the axial difference stream ax is aggregated to sum up potential errors. The measurements are combined to determine the scrap indicator $scrap$. If $scrap < 0.5$ the contact lens does not fulfill the required quality levels and has to be removed from the production. To produce stable monitoring results, individual scrap indicators are averaged in sliding windows. Finally, the drift of the scrap fraction, i.e the drift of the lens production quality, can be used to guide the maintenance planning of the production line.

To optimize the resulting data quality, all sampling rates and group sizes can be modified. For example, the contact lens scenario comprises $|sam| = 8$ sampling operators and $|agg| = 3$ aggregations. If applied in the data stream processing, also the group size of performed frequency analyses fre can be optimized. The size of the data quality windows constitutes the last optimization parameter, that can be determined independently for each data source $s \in S$. The contact lens quality requires $|S| = 6$ sensors, adding 6 parameters to the optimization problem. The overall problem optimizes $dim = |sam| + |agg| + |fre| + |S|$ parameters, e.g., $dim = 8 + 3 + 0 + 6 = 17$, defining the dimension of the problem domain.

1) Sampling Rate: The down-sampling reduces the data stream volume by randomly skipping a given set of data items defined by the sampling rate $r_{sa} < 1$. The information loss provoked by down-sampling represents a statistical error and has to be captured in the dimension confidence. Reconsider the data quality window presented above in Equation 1 with the true average $avg = 0.4mm$ sampled with $r_{sa} = 0.5$. The sample average ranges from $avg = 0.386mm$ to $avg = 0.414mm$ corresponding to a of $e^{+} = 0,0143mm$.

Definition: *The statistical error due to down-sampling can be estimated based on the confidence interval defined by Haas in [5], such that*

$$\epsilon_{+} = \frac{\alpha \cdot \sigma(w)}{\sqrt{\omega \cdot r_{sa}}} \cdot \sqrt{1 - r_{sa}}, \tag{12}$$

where $\sigma^2(w)$ states the variance of data stream values in the respective DQ window and α describes the confidence probability p as the $(1 - p/2)$-quantile.

Low sampling rates skip large data stream parts resulting in a high statistical error due to higher information loss. The confidence error rises with decreased sampling rate. On the contrary, low down-sampling rates reduce the data stream volume. The objectives of minimized confidence ϵ and minimized stream volume V conflict with each other. As timeliness and data stream volume are aligned due to higher processing times for large tuple numbers, the objectives of minimal timeliness u and minimal confidence ϵ conflict with each other in the same manner.

The up-sampling ($r_{sa} > 1$) inserts data items into the data stream. For example, a linear data interpolation is executed with the rate $r_{sa} = 2$, which doubles the data stream length. Hence, the up-sampling increases the fraction of interpolated data and has to be tracked by updating the DQ dimension completeness.

Definition: *During up-sampling, the window completeness c_w is divided by the sampling rate $r_{sa} > 1$, such that the completeness is stated as $c'_w = c_w / r_{sa}$.*

The higher the up-sampling rate, the more data items are generated. The lower is the resulting completeness and the higher the data stream volume. Hence, high up-sampling rates have a negative impact both on completeness c and data stream volume V.

As described above, the relation of stream rates has to remain constant for each processing tree level. For example, the up-sampling in data stream th_c increases the down-sampling rate for th_{e1-e4}. The up-sampling has an indirect positive impact on the statistical error in the data quality dimension confidence. The objective of maximized completeness c achieved by low sampling rates r_{sa} contradicts with the minimization of the confidence ϵ resulting from high r_{sa}.

2) Group Size: First and foremost, the group size l constitutes a parameter in the definition of data processing queries. For example during the calculation of the turnover, it is essential whether the sales volumes of one day ($l = 1d$) or one month ($l = 30d$) are summed up. However, during the processing of high-volume data streams situations can arise, where the group size must not be defined strictly.

The group size of aggregation and frequency analysis[1] share the same impact on the optimization objectives, as they summarize stream tuples to a smaller data volume. The larger their group size is defined, the more the data stream volume is reduced.

The more tuples are aggregated or serve as basis for frequency analysis, the larger is the amount of data of outcoming results. As the aggregation reduces the data volume, maximal amount of data goes along with minimal data stream volume. However, the summarization of information has a negative impact on the data stream granularity, as one processing result represents the timeframe spanned by all incoming data tuples in the respective group. The larger the group size, the larger is the resulting timeframe. During the configuration of the group size l positive effects on data stream volume V and amount of data d oppose negative effects on the granularity T.

The objective conflict can be resolved by configuring the data stream rate with the help of sampling and/or interpolation. During group sizes increase, the same stream rate increase leads to consistent amount of data. However, the consistency of optimal stream volume V, amount of data d and granularity T interferes with the objectives of minimized confidence ϵ and maximized completeness c as declared above.

3) Window Size: The definition of the size of data quality windows constitutes a compromise between fine granularity and high volume of transfered data quality information.

The wider the data quality window, the lower is the overhead for the data quality transfer and, thus, the lower is the overall stream volume. On the contrary, the larger the data quality window is defined, the more tuple-wise DQ information have to be aggregated in one window-wise quality value balancing out meaningful data quality peaks.

Therefore, we add the window size ω itself as optimization parameter. The objective function f_ω defining the averaged window size $\bar{\omega}_i$ over all data quality windows and stream attributes normalized with the maximal possible window size $\omega = m$ has to be minimized.

$$f_\omega \quad : \quad min \qquad \frac{1}{n \cdot m} \sum_{i=1}^{n} \bar{\omega}_i \tag{13}$$

As more data quality information have to be transfered and processed, the objective of low window sizes ω contradicts with minimal data stream volume V and maximal timeliness u.

[1]The frequency analysis computes amplitude and phase of all stream inherent frequency bands.

	ϵ	c	d	u	V	T	ω
ϵ	-	r_{sa}		r_{sa}	r_{sa}, l	r_{sa}, l	
c	r_{sa}	-	r_{sa}	r_{sa}	r_{sa}, l	r_{sa}, l	
d		r_{sa}	-			r_{sa}, l	
u	r_{sa}	r_{sa}		-		l	ω
V	r_{sa}, l	r_{sa}, l			-	l	ω
T	r_{sa}, l	r_{sa}, l	r_{sa}, l	l	l	-	
ω				ω	ω		-

TABLE II
OBJECTIVE CONFLICTS

Table II summarizes the conflicts between the optimization objectives. The cells indicate the configuration parameter leading to the respective conflict.

III. Quality-Driven Optimization

In this section, we present the framework architecture for DQ-driven optimization consisting of satisfiability checks and optimization component. To solve the defined problem, we propose the quality-driven evolution strategy.

A. Optimization Framework

The data quality-driven optimization is executed continuously to tune the data stream processing during system runtime. As soon as an optimal parameter set is found and deployed, it has to be checked against the currently processed data stream. The online tuning allows the seamless adaptation to varying stream rates, measurement values and data quality requirements.

First, the system evaluates by means of static information like maximal sensor stream rate or sensor precision, if the user-defined quality requirements can be accomplished or conflicts exclude a realization of all sub-objectives. In the latter case, the conflict is reported to the user. To check the satisfiability of DQ requirements, no access to streaming data is needed.

Heuristic optimization algorithms approximate the optimal problem solution by iteratively improving the achieved fitness. Different solution individuals have to be applied, evaluated and compared.

As the optimization must not interfere with the ongoing data stream processing, it is separated on an independent system component as illustrated in Figure 3. To execute the optimization in parallel with the traditional data stream processing, the processing path with all its operators is copied in the optimization component. In each iteration of the optimization algorithm the evaluated solution individual determines the specific path configuration of sampling and interpolation rates as well as group and window sizes.

Each solution is evaluated by directing a representative data stream partition through the configured processing path. We propose two approaches for the partition selection in Section III-C. As soon as the partition is completely processed, the average data quality result for each dimension, the average granularity and the used data stream volume are computed

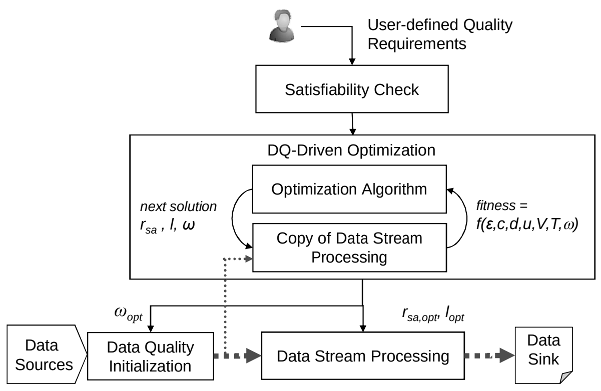

User-defined Quality Requirements
Satisfiability Check
DQ-Driven Optimization
Optimization Algorithm
next solution r_{sa}, l, ω
fitness = $f(\epsilon,c,d,u,V,T,\omega)$
Copy of Data Stream Processing
ω_{opt}
$r_{sa,opt}$, l_{opt}
Data Sources
Data Quality Initialization
Data Stream Processing
Data Sink

Fig. 3. Optimization process

and returned to the optimization algorithm to calculate the fitness of the tested configuration. If the user-defined quality requirements are not met, the fitness guides the determination of the next solution individual to iteratively improve the achieved fitness.

As soon as the requirements are accomplished, the optimization problem is solved. The new parameter setting is applied to the original processing path. The sampling operators are updated with the optimized sampling rates $r_{sa,opt}$. The frequency analyses and aggregations are updated with the determined group sizes l_{opt}. Finally, the data quality initialization at the sensor nodes is re-configured with the new window sizes ω_{opt}. After deploying the found parameter set, the next optimization run is performed to adapt the processing to dynamic streams characteristics.

The logical distinction between optimization and processing enables also the physical separation, for example on a distinct server node. Thus, the optimization task has no negative impact on the performance of the traditional data stream processing.

B. Satisfiability Check

Four quality checks have to be performed to prevent from optimizing against unsatisfiable user-requirements as shown in Algorithm 1.

First, the desired confidence error is evaluated. The best possible confidence is achieved, when no down-sampling is performed. Only initial statistical errors of the sensors reduce the confidence. Thus, the requested confidence objective ϵ_{req} must not be smaller than the square root of all initial confidence errors ϵ_{S_i} (line 1).

Second, only completeness deficiencies due to up-sampling can be reduced by DQ-driven optimization. The completeness objective c_{req} must not exceed the average initial completeness c_{S_i} provided by the sensors (line 2).

The conflicting objectives of minimal data stream volume V, minimal window size ω and maximal amount of data d are compared based on the resulting stream length m. As the amount of data cannot exceed the stream length, it must hold that $d_{req} \leq m$. By applying Equation 8, we derive the satisfiability check shown in line 3.

Finally, the conflict between amount of data d and granularity T is controlled based on the maximal stream rate r_{max} introduced in Section II-B in line 4. For example, the required amount of data of $d_{req} = 100$ and the maximal rate of $r_{max} = 1/ms$ leads to the minimal achievable granularity of $T_{min} = 100ms$.

Algorithm 1: Satisfiability Check

```
Input: $\epsilon_{req}, c_{req}, V_{req}$ user-defined requirements
Output: $sat$=FALSE satisfiability
1 if $\epsilon_{req} \geq \sqrt{\sum_{i=1}^{|S|} \epsilon_{S_i}^2}$
2 $\wedge c_{req} \leq \frac{1}{|S|} \sum_{i=1}^{|S|} c_{S_i}$
3 $\wedge d_{req} \leq \frac{V_{req}}{n+1+\sum_{i=1}^{n} \frac{1}{\omega_{req}} |Q_i|}$
4 $\wedge T_{req} \geq \frac{d_{req}}{r_{max}}$
5 then
6     $sat$ = TRUE;
```

C. Batch vs. Continuous Optimization

The stream partition for optimization constitute a data stream window of ζ data tuples. It may either be selected in batch-mode at the beginning of each optimization run and used in each iteration without changes. At the other hand, the window can be updated for each iteration with current tuples to reflect the dynamic progression of the data stream and allow the continuous optimization.

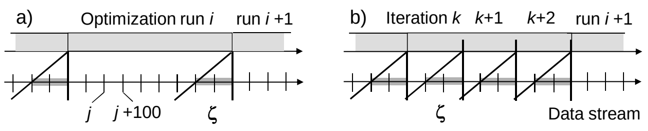


Fig. 4. Batch & continuous optimization

The batch-mode selects a constant stream partition of the last ζ tuples for one complete optimization run as shown in Figure 4a. The parameter ζ depicts the trade-off between representative partition length and duration of one optimization run. Besides, the length is restricted by limited memory capacity in most streaming environments. It only represents a small, static data stream window.

In contrast, the continuous optimization approach follows the dynamic stream behavior by selecting a new stream partition for each iteration of the applied optimization algorithm. As shown in Figure 4b, a larger stream fraction is used. However, the continuous mode exchanges the base data for optimization in each iteration. A good solution derived from the fitness in iteration k on partition k may perform poor if applied to base data $k+1$ in iteration $k+1$. The fitness will not increase monotonically, but may diverge. To guarantee the algorithm termination, additional criteria like allowed duration or number of fitness evaluations have to be defined.

The static optimization in batch-mode guarantees the converging of the fitness function and, thus, the accomplishment of user-defined quality requirements. However, the computed optimal solution only holds for the processed short stream partition, so that a permanent control with subsequent optimization runs is necessary. The continuous approach allows the inclusion of dynamic data stream alterations during the optimization process. Thereby, divergences of the fitness function may arise that must be encountered by supplementary terminations rules.

D. Optimization Classification

The operations research combines mathematics and formal science to find and define methods and algorithms to arrive at optimal or near optimal solutions to complex problems. Optimization problems are classified according to their domain, their objective function and respective problem constraints. In this section, we will range the optimization problem of data quality-driven stream processing in this classification scheme.

The problem analysis revealed seven objectives, which partly conflict with each other and define a multi-objective optimization problem. There exist various strategies to solve these problems. On the one hand, the Pareto optimization allows the finding of a set of optimal compromises, that dominate all other possible solutions. That is, the improvement of one sub-objective is only possible with a decline of one or more of the other sub-objectives. In Section IV we define the multi-objective quality-driven evolution strategy (MO-QES), which optimizes the Pareto front in the problem search domain by solving the objective f_{multi}.

Definition: *The multi-objective fitness function* $f_{multi} : \mathbb{R}^{dim} \mapsto \mathbb{R}^7$ *defines the fitness of a solution individual as the Pareto-dominance of the achieved sub-objective* $f_i (i \in \{\epsilon, c, d, u, V, T, \delta\})$.

On the other hand, sub-objectives can be summarized in one optimization function f_{single}, which will be minimized or maximized by the single-objective quality-driven evolution strategy (SO-QES). Cost weights defined for each objective determine the order and importance of optimization. However, the user-defined weighing may restrict the search space resulting in minor optimization solutions. Multiple optimization runs are required to determine optimal cost weights.

Definition: *The single-objective fitness function* $f_{single} : \mathbb{R}^{dim} \mapsto \mathbb{R}$ *calculates the overall fitness of a solution individual as weighted sum of the obtained sub-objectives, such as*

$$f_{single} = \sum_{i \in \{\epsilon, c, d, u, V, T, \delta\}} c_i \cdot f_i \qquad (14)$$

The data quality dimensions completeness, amount of data and timeliness are defined using linear computation functions, so that f_c, f_d and f_u pose linear optimization problems.

The same holds for the data quality window size f_ω and the granularity f_T. Due to the square root definition of the statistical error computation, the objective function f_ϵ gives a non-linear optimization problem. A second non-linear problem is posed by f_V, which minimizes the stream volume based on the window size ω.

While the parameters l and ω as well as fitness of the sub-objectives f_d, f_u, f_V, f_T and f_ω allow only discrete values, the domains of the parameters r_{sa} and the objective functions f_ϵ and f_c propose continuous optimization problems.

Moreover, the search space is restricted by side conditions. To guarantee join partners, the data stream rates in one tree level have to stay in constant relation to each other. For example, the duplication of $r_{th_{e1}}$ (see Figure 2) requires i.a. the duplication of r_{ax}. Either, the rate of the sampling operator $r_{sa,ax}$ has to be doubled, or the group size l_{ax} has to be halved. Thus, the dependence of sampling rates and group sizes defines side constraints for each processing tree level, such that

$$\forall\ level\ l \quad \forall\ r_i, r_j \in l : r_i = r_j. \tag{15}$$

As in multi-objective optimization problems the most complex sub-objective defines the complexity of the overall problem, the quality-driven optimization is defined as follows.

Definition: *The quality-driven process optimization constitutes a multi-objective, non-linear, continuous optimization problem with side conditions.*

There exists no deterministic algorithm to solve optimization problems of this complexity in reasonable time. However, the stream processing optimization shall be executed on-the-fly without interrupting the data flow. It is essential to provide good solutions in an acceptable timeframe. Further, the optimal solution is not required in most cases. Rather, the user or data consuming application defines data quality levels, which have to be met. Heuristic algorithms offer an appropriate answer to such optimization problems. They provide fast results by approximating the optimal solution.

Heuristic optimization algorithms range from simple approaches like the Monte-Carlo-Search over more sophisticated strategies such as Hill Climbing and Simulated Annealing to complex Evolutionary Algorithms. Evolutionary Algorithms comprise genetic algorithms working on binary data and the evolution strategy supporting real parameters and objectives. As only the evolution strategy can solve real-valued multi-objective as well as single-objective problems, we apply this heuristic to solve the DQ-driven optimization.

IV. Evolution Strategy

The evolution strategy constitutes a stochastic, population-based search heuristic inspired by the principles of natural evolution. It can be applied to arbitrary optimization problems and requires nothing but the objective function(s) of the optimization problem to guide its search. A population of possible solutions is iteratively recombined and mutated to select the best population individual as approximated global optima.

To improve the data quality via configuration of the data processing, we have to adapt the generic algorithm structure of the evolution strategy to the defined quality-driven optimization problem. In this section, we present the specification of the quality-driven evolution strategy (QES) including specific functions for recombination, mutation and selection.

Algorithm 2: QES

```
Input: $domain$ of possible inputs,
$DQ$ user-defined requirements
Output: $P$ population of optimal solutions
1 $t$ = 0;
2 initialize($P(t)$, $domain$);
3 while $DQ.notAchieved()$ do
4     $P_c(t)$ = recombine($P(t)$);
5     mutate($P_c(t)$);
6     $P(t+1)$ = selectNextGeneration($P_c(t), P(t)$);
7     $t$ = $t$ + 1;
8 end
```

Algorithm 2 shows the overall structure of QES. The first population $P(0)$ is randomly initialized in line 2 to cover the complete search $domain$. The first step of the repeated iteration process recombines the solution individuals of the current population $P(t)$ in line 4 to build new solution candidates $P_c(t)$. They are randomly mutated to allow new solutions to enter the population in line 5. Finally, they are evaluated with the help of the objective function f_{single} or f_{multi}, respectively, to select the best individuals of $P_c(t)$ to build the next generation $P(t+1)$ in line 6. The quality-driven evolution strategy terminates, when all user-defined data quality requirements are met (line 3). Other termination criteria are the allowed number of performed solution evaluations or the planned execution time.

The *recombination* is designed to hand down and combine positive traits of different solutions individuals. First, parent solutions are chosen randomly from the current generation. Then, the dim parameter configurations of parent pairs are combined. The children's parameters are determined by averaging each of the parents' parameters. As group and window sizes only allow integer values, they are rounded up.

Algorithm 3 illustrates the *mutation* of the recombined solution candidates. Due to different value domains, specific mutation steps sizes are applied to each parameter type: sampling and interpolation rates $\Delta r_{sa/in}$ in line 4, group size Δl (line 6) and window size $\Delta\omega$ (line 8). To ease the application of the QES and to allow for the automatic adaptation, the specific step sizes are considered as additional optimization parameters. The parameter vector is extended by three variables: $dim' = dim + 3$. The step sizes themselves are mutated by as shown in line 10.

The mutation is executed individually for each solution of

Algorithm 3: mutate()

```
Input: $P$ current population
Output: $P_m$ mutated population
1  forall $a \in P$ do
2      $index = random(1, dim + 3)$;
3      $oldValue = a$.getParameterAt($index$);
4      if type(index) = sampling || interpolation then
5          $newValue = oldValue \pm \Delta r_{sa/in}$;
6      else if type(index) = groupSize then
7          $newValue = oldValue \pm \Delta l$;
8      else if type(index) = windowSize then
9          $newValue = oldValue \pm \Delta\omega$;
10     else
11         $newValue = oldValue \pm 0.1 \cdot oldValue$ ;
12     $a$.setParameterAt($index$, $newValue$);
13 end
```

the current population. The parameter to mutate is selected randomly from the configuration set (line 1 & 2) and mutated according to the respective step size (lines 3-10). The new solution individual b is created by exchanging the mutated parameter in the current solution a (line 11).

The last algorithm step, the *selection*, evaluates the new solution candidates to form the next population generation. The quality-driven evolution strategy follows the $(\mu + \lambda)$-approach, that produces a monotonically nondecreasing fitness curve. One population consists of μ elements, which are used to produce λ candidates. The fitness of all parent and child solutions is calculated with the help of the objective function. The μ fittest solution individuals build the new generation as starting point for the next algorithm iteration.

The single-objective quality-driven evolution strategy (SO-QES) uses the cost-weighted objective function f_{single}. The optimization problem defined in f_{multi} is solved by the multi-objective evolution strategy (MO-QES). The comparison of optimization results allows conclusions on the impact of different cost settings. Furthermore, the following section evaluates the performance of the two optimization approaches.

V. Evaluation

In this evaluation, we examine data quality-driven optimization of the data stream processing at real-world data streams to empirically answer the following questions.

1) Which impact has the cost weighing on the optimal configuration of the data stream processing?
2) What are the benefits of batch and continuous optimization?
3) How do single- and multi-objective optimization compete with each other?
4) Do the presented algorithms scale for the complex data stream processing with high sensor numbers?

We have implemented the quality-driven optimization described in this paper using the data stream management system PIPES [6] and the Java-based optimization framework JavaEva [7].

A. Experimental Setting

We ran our experiments on a dual-core 2x2GHz Centrino Duo processor with 2GB of main memory, running Microsoft Windows XP Professional 2002. All Java-based systems were executed using JRE Version 6.

We use an artificial dataset to analyze the effects of the optimization modes and test the performance of the designed algorithms. Therefore, we generated data streams subject to the standard normal distribution ($\mu = 0, \sigma = 1$) with randomly varying stream rates in the range of $1/ms \leq r \leq 100/ms$. We simulated queries over 2 to 128 of such generated streams joined in pairs of two. Each data stream is sampled in each query tree level to find one-to-one-join partners; aggregations are spread randomly.

Further, we applied the real-world dataset of contact lens manufacturing available at [8] to analyze the impact of cost weights and to examine the practicability of the presented algorithm. It consists of measurements of the thickness of lens center and edge and the axial difference. As described in Section II-C, the production quality is monitored to predict the optimal maintenance planning for re-calibrating the production line.

For both datasets, we assumed a systematic error of optimistic 1%, while the statistical measurement error was derived from the measurements' variance using the confidence probability $p = 99\%$. To simulate sensor failures in the artificial dataset, we randomly skipped 2% of the data tuples. To initialize the lens data completeness, we identified missing measurements by comparing the recorded timestamps to the planned sensor rates.

B. Impact of Weights

This section answers the first of the above questions at the sample objectives of maximal completeness and minimal confidence (compare Table I). Figure 5 shows the Pareto front of the multi-objective optimization. Minimal statistical confidence errors produced by high sampling rates are only achieved at the expense of high values of incompleteness and vice versa. The optimal compromises represented by the Pareto front can be re-produced by the single-objective optimization using sophisticated weighing.

Points A,B and C in Figure 5 show exemplary single-objective results. The higher the weight was determined, the better the proposed optimal configuration suits the respective sub-objective. If the cost for incomplete data tuples exceeds the confidence weight, low sampling and interpolation rates are proposed (point C). High costs for statistical errors in the DQ dimensions confidence (point A) lead to high sampling rates resulting in less data loss. Similar cost weights result in a well-balanced compromise between completeness and confidence as given in point B.

Figure 6 illustrates the Pareto front and cost impact of the conflicting objectives maximal amount of data and maximal

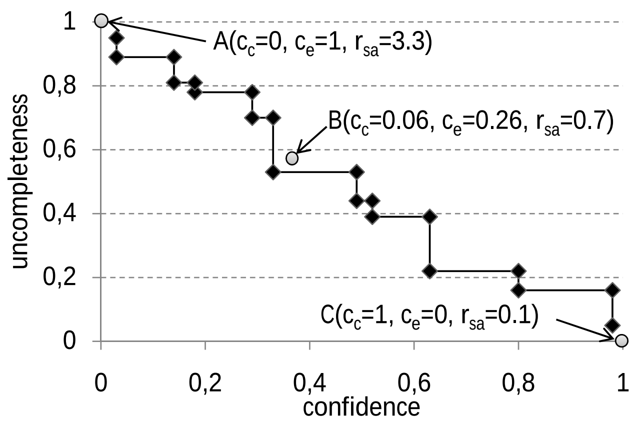


Fig. 5. Confidence vs. completeness

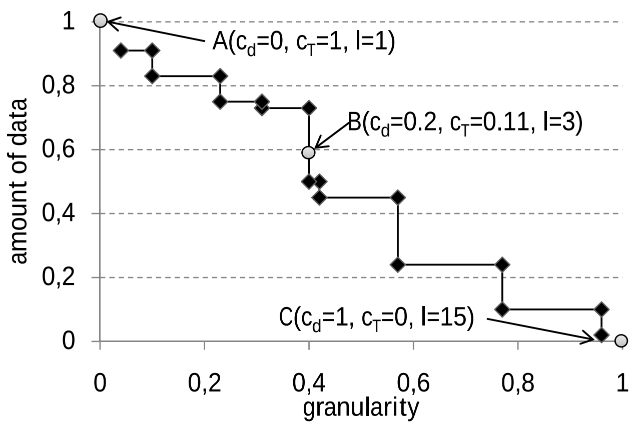


Fig. 6. Amount of data vs. granularity

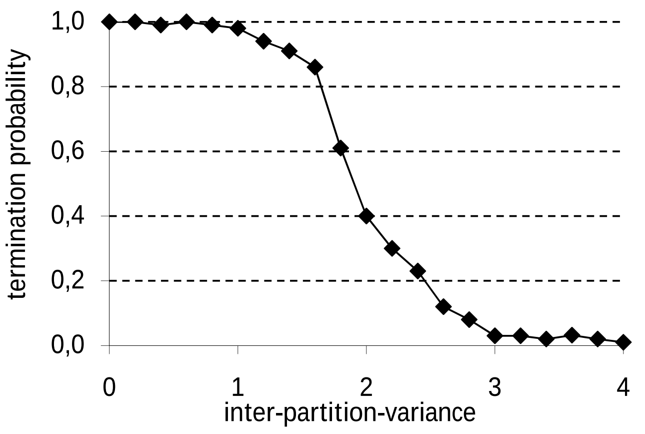


Fig. 7. Termination probability

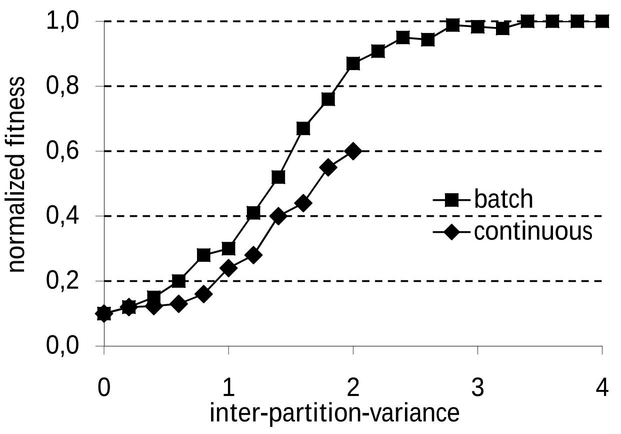


Fig. 8. Normalized fitness

granularity. High amount of data leads long timeframes represented by each tuples, i.e., low granularity. The higher the cost weight for the amount of data exceeds the cost for fine granularity, the higher is the average group size in the proposed optimization solution and vice versa (compare weights of points A, B and C in Figure 6).

Finally, we evaluated the conflicting objectives of minimal data stream volume and minimal data quality window size. Similar to the above evaluations, higher weighing of the stream volume results in low window sizes and vice versa.

C. Comparison of Optimization Modes

This section answers the second question. First, we check the termination probability of the continuous optimization, which selects a new data stream partition as basis for each optimization iteration. Therefore, we applied the artificial data set described above to the single-objective optimization. We kept the intra-partition-variance ($\sigma = 1$), but introduced an inter-partition-variance $\breve{\sigma}$ by modifying the mean μ for each partition.

Figure 7 shows the termination probability of the single- and multi-objective optimization for an increasing inter-partition-variance. For $\breve{\sigma} < 1$, the continuous optimization is likely to terminate successfully. The termination probability decreases for $1 \leq \breve{\sigma} \leq 3$ and converges to 0 for $\breve{\sigma} > 3$. The termination probability is independent from the applied partition length.

To compare the quality of the optimization results provided by batch and continuous mode, we apply the computed optimization results, i.e., the operator configurations, to the ongoing stream and compare the achieved overall quality. Figure 8 shows the normalized fitness for an increasing inter-partition-variance. The batch mode allows good results for low variances $\breve{\sigma} \leq 0.5$. The continuous mode adapts better to changing situations and thus provides appropriate fitness values also for a higher inter-partition-variance. However, the increasing termination probability limits the application of the continuous mode to the bound $\breve{\sigma} \leq 2$. Here, a small DQ improvement is only possible by using batch mode again.

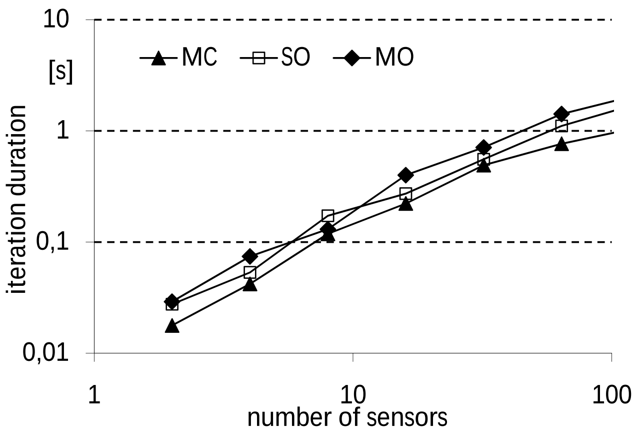


Fig. 9. Iteration duration vs. number of sensors

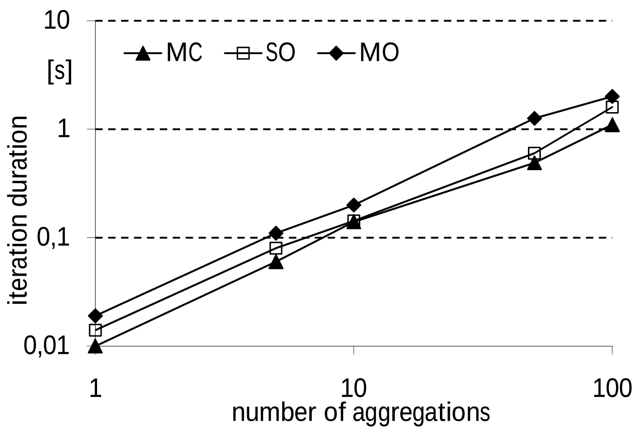


Fig. 10. Iteration duration vs. number of aggregations

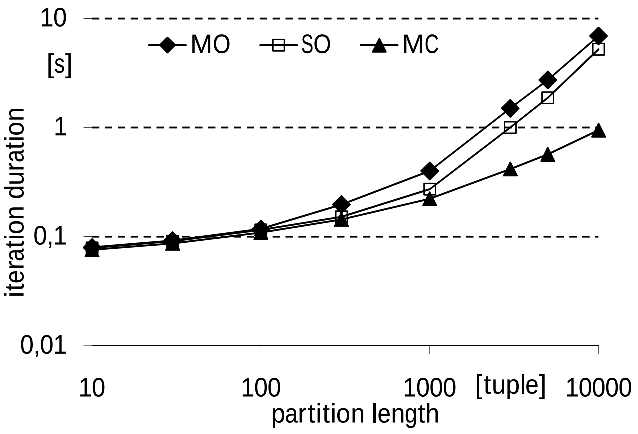


Fig. 11. Iteration duration vs. partition length

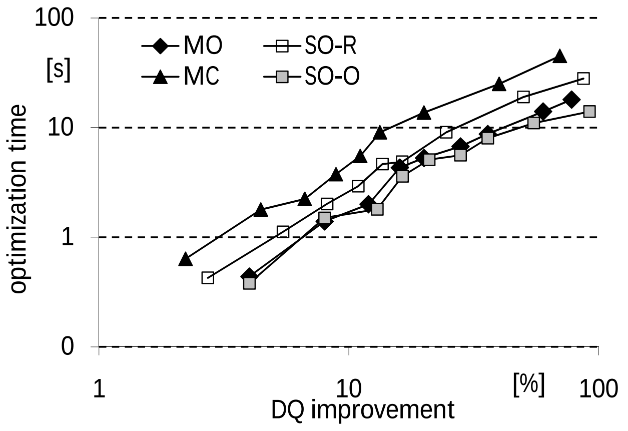


Fig. 12. Duration of DQ improvement

D. Performance Tests

In this section, we show the scalability of the presented algorithms by means of the artificial data set and complex query structure defined in Section V-A. As no inter-partition-variance was introduced, the evaluation tests were performed in batch-mode. We first analyze the performance of the proposed algorithms with regard to increasing numbers of data sources (sensors), applied aggregations and frequency analyses, respectively. As the number of sampling operators is defined by the executed joins and, thus, by the number of sensors, an individual scalability test for that operator class is not required. Then, we evaluate the impact of the length of the data stream partition used for optimization. Finally, we compare the optimization time required by the single- and multi-objective optimization strategy to improve the overall quality.

The Monte-Carlo-Search (MC) performs a random search over the problem domain and serves as reference value defining the lower performance bound [9]. The single-objective optimization is executed with randomly chosen weights (SO-R) as well as optimal weights (SO-O), which determine a well-balanced objective compromise. As the iteration duration of the single-objective optimization does not depend on the weights, these performance test results of SO-R and SO-O have been summarized to SO. Finally, the multi-objective optimization (MO) approximates the Pareto front of all optimal compromises.

Figure 9 illustrates the impact of the sensor number on the time required for one iteration of the respective optimization algorithm executed with a partition length of 1000 tuples. The more complex the algorithm, the longer takes one iteration. The performance difference between single- and multi-objective optimization is caused by the complex Pareto front computation. For all tested algorithms, the iteration duration increases linearly with the sensor number.

Figure 10 shows the scalability for increasing numbers of aggregations and frequency analyses. Again, the time required for one iteration rises linearly.

Figure 11 shows the iteration duration (in seconds) for 16

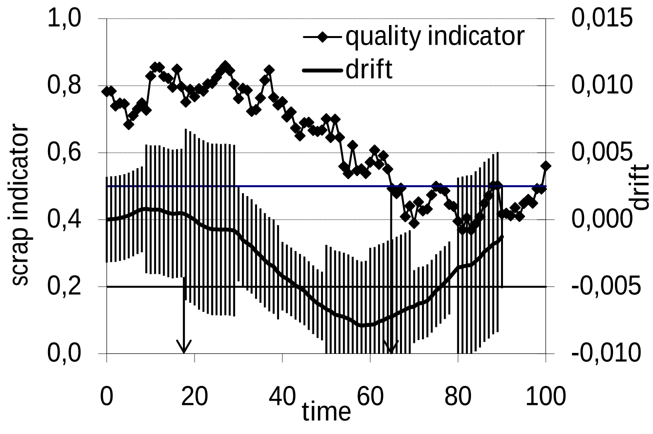


Fig. 13. Lens quality control before optimization

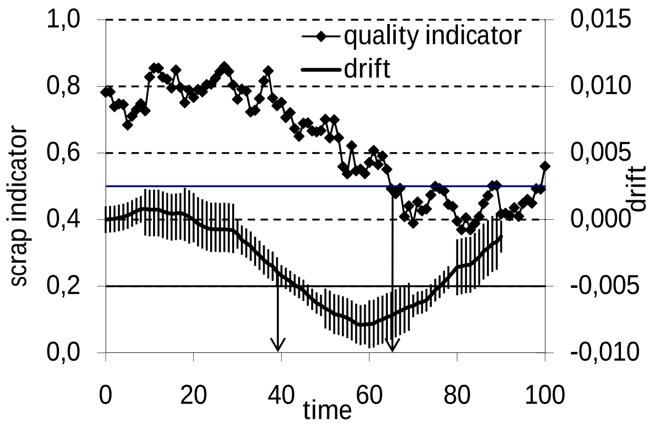


Fig. 14. Lens quality control with MO

sensors and 10 aggregations for increasing partition lengths (in data tuples). The processing time rises nearly linearly for small to medium partition lengths of 100 to 1000 stream tuples. Only for very large stream partitions, the iteration duration exhibits an exponential character.

Figure 12 compares the overall time performance of single- and multi-objective optimization with respect to the achieved quality improvement for 16 sensor data sources and 10 randomly inserted aggregations. The quality improvement is expressed as percentage value $(q_{before} - q_{after})/q_{before}$. The Monte-Carlo-Search performs worth followed by the randomly initiated single-objective optimization (SO-R), requiring 5.2 and 2.9 seconds, respectively, for a DQ improvement of 10%. The single-objective optimization executed with well-balanced weights (SO-O) performs best (1.6s for 10%). However, the definition of these weights requires multiple optimization runs and has to be adapted as soon as stream characteristics or user requirements change. The multi-objective optimization (MO) is a little slower (1.9s) due to the complex computation of the Pareto front. However, the result comprises the complete set of all optimal compromises and no pre-processing to determine optimal weights is necessary.

The evaluation showed, that the designed quality-driven optimization provides good scalability with regard the applied stream partition length as well as increasing complexity of the data stream processing. Data quality and quality of service could be improved within few seconds. Further, we deduce that the single-objective optimization in batch mode is the best choice for constant user requirements and steady data streams. If streaming data values present high fluctuations or user requirements are often adjusted, the multi-objective optimization constitutes the better option. Here, the inter-partition-variance has to be analyzed to estimate the termination probability. While the continuous mode provides better results for medium variances, the batch mode has to be applied for highly varying data streams to prevent from divergent fitness functions.

E. Lens Production Optimization

To approve the suitability of the designed algorithms, we refer to the application scenario of scrap monitoring for predictive maintenance in the contact lens production introduced in Section II-C. Due to numerical errors and missing sensor measurements, the predicted maintenance date deviates from the optimal point in time. The maintenance is scheduled either too early (the calculated quality drift exceeds the true value), or too late (the drift of the computed quality indicator is too low).

To improve the reliability of the determined maintenance planing, we aim to optimize the underlying data stream processing. The first optimization parameters are the sampling rates on the data streams of the lens thicknesses and axial difference. The group sizes of the axial difference summation and sliding average aggregation of the quality identifier determine the compromise of detailed information and outlier balancing. The last optimization criteria is given by the group size of the drift calculation that provides statistically stable short- or long-term variations.

We start the optimization process with arbitrary settings of the optimization parameters. Figure 13 illustrates the scrap indicator of the contact lenses and the derived drift of the production quality. To guarantee correct maintenance activities so that no lens under the scrap threshold of 0.5 remain in the production line, the drift is monitored against the drift threshold of -0,005. As the calculated drift suffers from measurement errors and uncertainties due to sensor failures, the lowest possible bound of the drift must be taken into account. Therefore, the absolute measurement error (the sum of systematic and statistical error) is illustrated as error bars of the drift function. The arrow indicates the resulting maintenance time at $t = 19$. However, due to the high measurement error before the optimization, this maintenance date is too early considering the actual scrap indicator, that falls below 0.5 at $t = 64$.

Figure 14 shows the same situation after 10 iteration of the multi-objective optimization of the data stream processing has been executed. The absolute measurement errors have been decreased significantly, such that the "'lost'" production time could be reduced by starting the maintenance only at $t = 39$.

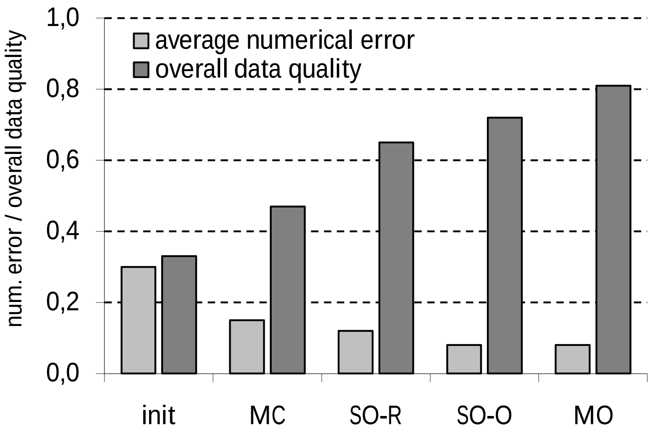


Fig. 15. Improvement of overall DQ & num. error

Figure 15 shows the overall data quality and the numerical error after 100 iterations for each of the presented optimization heuristics, averaged over 10 optimization runs. Already the simple random Monte-Carlo-Search halves the numerical error defining the uncertain range of the quality drift. The randomly initiated single-objective optimization (SO-R) further decreases the error by 30%. The best results are obtained by SO-O and MO. SO-O achieves an overall error reduction of 73%. MO provides a set of optimal compromise solutions: as all sub-objectives shall be improved, Figure 15 illustrates the error reduction (74%) for the well-balanced compromise of weights: $c_c = 0.06, c_\epsilon = 0.26, c_d = 0.2, c_T = 0.11, c_V = 0.2, c_\omega = 0.11$ (compare point B in Figure 5 and 6).

After the data quality-driven optimization, the numerical errors are decreased leading to a narrower uncertainty range. The determined maintenance planning approximates the optimal point in time with higher confidence. Premature maintenance, that unnecessarily interrupts the contact lens production, as well as too late activities, that risk loss of sales due to low production quality, are both prevented.

VI. Related Work

In this section, we will discuss related work in the fields of data quality management and optimization methods, especially focusing on multi-objective optimization. This paper gave an extended view over the quality-driven optimization of sensor data stream processing, that we first published in [2]. Besides detailed definitions of the objective functions as well as optimization parameters, we added the discussion of the batch and continuous optimization strategy as well as the crucial definition of the stream partition length used for the optimization. Further, the evaluation was deepened to show the influences of different optimization techniques as well as the scalability of the overall approach not only at artificial data streams, but also using the real-world example of contact lens production monitoring.

Traditional approaches of query optimization in database and data stream systems aim at minimal processing time and maximal data throughput. The quality of service is improved to provide fast processing results to the user. In this paper, we address the opposite problem: we improve the quality of data. We maximize the data quality of processing results under consideration of restricted system resources, so that the quality of service remains in acceptable ranges.

Multiple publications underline benefits of the data quality management in data warehouses and databases [10][11]. To define the term data quality, different sets of data quality dimensions are discussed i.a. in [12] and [3]. While there are different approaches to structure data quality metadata in databases [13][14], [1] presents the first data quality model suitable for data streaming environments using so called jumping data quality windows.

Data quality improvement in the context of data warehouse and information systems is achieved by data cleaning [15][16]. For example, the Total Data Quality Management (TDQM) provides tools to analyze the data quality in information systems and suggests data cleansing techniques for DQ improvement [17]. However, prior work in this domain suffers from the major drawback, that either an active participation of users or domain experts in data quality improvement is necessary or the presented approaches refer to a (set of) reference data source(s) providing better or optimal data quality. It is obvious that in case of sensor data, the manual subsequent data quality correction for each measurement item is not feasible, and a high-quality reference for comparison is not present.

Instead, the data and data quality processing has to be configured to reduce the error amplification. Based on the classification of the quality-driven optimization problem, we identified the heuristic evolution strategy as appropriate tool to approximate the optimal problem solution in an acceptable timeframe. For the evaluation of different implementations of the evolution strategy we rely on the comprehensive study and experimental analysis provided in prior work and focus on the SPEA as a well-studied algorithm with high rankings in multiple test cases [18][19].

The application to multi-objective optimization problems is described in [20]. Empirical studies showed the practicability and superiority of the multi-objective evolution strategy (MOES) [21][22]. [23] discusses the benefits of parallel execution of evolutionary algorithms, which would further improve the performance of the quality-driven evolution strategy.

VII. Conclusion

In this paper, we presented the quality-driven optimization of sensor stream processing. On the one hand, the data quality of processing results, expressed by the DQ dimensions accuracy, confidence, completeness, amount of data and timeliness, were improved. On the other hand, the quality of service was increased by minimizing the data stream volume to comply with resource constraints in data streaming environments.

To identify candidates for the data quality improvement, we analyzed the operator repository of the quality propagation model presented in [1] and extracted sampling rate and group size as configuration parameters. Furthermore, the size of jumping data quality windows was detected as promising parameter for the data quality optimization. Based on these insights, we defined the multi-objective, non-linear, continuous optimization problem with side conditions.

To solve the problem of quality-driven optimization, we presented the generic optimization framework that can be instantiated with any optimization algorithm. Optimization time and quality were improved by satisfiability checks and two optimization modes for changing stream characteristics. Evolutionary algorithms represent the most promising optimization strategies for the defined complex optimization problem. Thus, we developed the quality-driven evolution strategy QES as sample instantiation of the generic framework.

Finally, we evaluated the proposed optimization strategy with the help of artificial data streams as well as real-world data from the contact lens production control. We showed, that QES solves the optimization problem in a reasonable timeframe and provides good scalability for complex data stream processing queries. The maintenance prediction for contact lens production could be determined more precisely by improving the data quality of the calculated maintenance date.

VIII. Acknowledgment

This work has been partially supported by the European Commission under the contract FP7-ICT-224282 (GINSENG).

References

[1] A. Klein, "Incorporating quality aspects in sensor data streams," in *Proceedings of the 1st ACM Ph.D. Workshop in CIKM (PIKM)*, 2007, pp. 77–84.

[2] A. Klein and W. Lehner, "How to optimize the quality of sensor data streams," in *ICCGI '09: Proceedings of the 2009 Fourth International Multi-Conference on Computing in the Global Information Technology*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 13–19.

[3] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, 1996.

[4] M.-L. Lee, T. W. Ling, H. Lu, and Y. T. Ko, "Cleansing data for mining and warehousing," in *DEXA*, 1999, pp. 751–760.

[5] P. J. Haas, "Large-sample and deterministic confidence intervals for online aggregation," in *SSDM*, 1997, pp. 51–63.

[6] J. Kraemer and B. Seeger, "Pipes - a public infrastructure for processing and exploring streams," in *SIGMOD*, G. Weikum, A. C. Koenig, and S. Deloch, Eds., 2004, pp. 925–926.

[7] A. Zell, "Javaeva: A java based framework for evolutionary algorithms," 2009, http://www.ra.cs.uni-tuebingen.de/software/EvA2, last access: 14.06.2010.

[8] R. L. Edgeman and S. B. Athey, "Digidot plots for process surveillance," *Quality Progress*, vol. 23, no. 5, pp. 66–68, 1990.

[9] N. R. Patel, R. L. Smith, and Z. B. Zabinsky, "Pure adaptive search in monte carlo optimization," *Mathematical Programing*, vol. 43, no. 3, pp. 317–328, 1989.

[10] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Springer-Verlag, 2006.

[11] J. M. Juran, *Juran's quality control handbook*, F. M. Gryna, Ed. McGraw-Hill, 1988.

[12] F. Naumann and C. Rolker, "Assessment methods for information quality criteria," in *ICIQ*, 2000, pp. 148–162.

[13] V. C. Storey and R. Y. Wang, "An analysis of quality requirements in database design," in *ICIQ*, 1998, pp. 64–87.

[14] D. M. Strong, Y. W. Lee, and R. Y. Wang, "Data quality in context," *Communications of the ACM*, vol. 40, no. 5, pp. 103–110, 1997.

[15] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining Knowledge Discovery*, vol. 2, no. 1, pp. 9–37, 1998.

[16] R. C. Morey, "Estimating and improving the quality of information in a mis," *Communications of the ACM*, vol. 25, no. 5, pp. 337–342, 1982.

[17] J. M. Pearson, C. S. McCahon, and R. T. Hightower, "Total quality management: are information systems managers ready?" *Information Management*, vol. 29, no. 5, pp. 251–263, 1995.

[18] Z. Michalewicz, *Genetic Algorithms Plus Data Structures Equals Evolution Programs*. Springer-Verlag, 1994.

[19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[20] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications (Advanced Information and Knowledge Processing)*. Springer-Verlag, 2005.

[21] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computing*, vol. 8, no. 2, pp. 173–195, 2000.

[22] T. Hanne, "Global multiobjective optimization using evolutionary algorithms," *Journal of Heuristics*, vol. 6, no. 3, pp. 347–360, 2000.

[23] D. A. V. Veldhuizen, J. B. Zydallis, and G. B. Lamont, "Issues in parallelizing multiobjective evolutionary algorithms for real world applications," in *ACM Symposium on Applied Computing*, 2002, pp. 595–602.