Observing the Pulse of a City: A Smart City Framework for Real-time Discovery, Federation, and Aggregation of Data Streams

Şefki Kolozali[1,4], Maria Bermudez-Edo[1,5], Nazli FarajiDavar[1,6], Payam Barnaghi[1], Feng Gao[2], Muhammad Intizar Ali[2], Alessandra Mileo[7], Marten Fischer[3], Thorben Iggena[3], Daniel Kuemper[3], and Ralf Tonjes[3]

[1]Institute for Communication Systems, Electronic Engineering Department, University of Surrey, UK
[2]Insight Centre for Data Analytics, National University of Ireland, Galway
[3]Faculty of Engineering and Computer Science, University of Applied Sciences Osnabrück, Germany
[4]MRC-PHE Centre for Environment & Health, King's College London
[5]School of Information Technology and Telecommunications, University of Granada
[6]Institute of Biomedical Engineering, University of Oxford
[7]Insight Centre for Data Analytics at Dublin City University, Dublin

Abstract—An increasing number of cities are confronted with challenges resulting from the rapid urbanisation and new demands that a rapidly growing digital economy imposes on current applications and information systems. Smart city applications enable city authorities to monitor, manage and provide plans for public resources and infrastructures in city environments, while offering citizens and businesses to develop and use intelligent services in cities. However, providing such smart city applications gives rise to several issues such as semantic heterogeneity and trustworthiness of data sources, and extracting up-to-date information in real time from large-scale dynamic data streams. In order to address these issues, we propose a novel framework with an efficient semantic data processing pipeline, allowing for real-time observation of the pulse of a city. The proposed framework enables efficient semantic integration of data streams and complex event processing on top of real-time data aggregation and quality analysis in a Semantic Web environment. To evaluate our system, we use real-time sensor observations that have been published via an open platform called Open Data Aarhus by the City of Aarhus. We examine the framework utilising Symbolic Aggregate Approximation to reduce the size of data streams, and perform quality analysis taking into account both single and multiple data streams. We also investigate the optimisation of the semantic data discovery and integration based on the proposed stream quality analysis and data aggregation techniques.

Index Terms—Smart Cities, Internet of Things, Time Series Analysis, Complex Event Processing, Quality Analysis

1 Introduction

Over centuries, cities have been a flourishing place for people on account of prosperity and socio-economic prospects. While citizens of a city form the key facet of city systems, city systems need to serve the demands of its inhabitants and elaborate interactions between different city applications. City authorities, however, encounter several difficulties in implementing, sustaining, and optimising operations and interactions among different city departments and services [1]. Therefore, smart cities seek to improve living standards and quality of life of its citizens through the utilisation of the advancements in the Internet of Things (IoT) to tackle common urban challenges such as reducing energy consumption, traffic congestion [2] and environmental pollution [3], [4]. Smart cities rely on data streams collected from various sensors (e.g. traffic congestion level, air quality and trash bin levels) to observe the pulse of cities. City of Aarhus provides an open data platform called Open Data Aarhus (ODAA)[1], which contains city related information generated by various sensors deployed within the city. Although such platforms allow city-related data to be published and shared, considering the fact that vast a number of sensors will be connected to the Internet in the future, leading consequent challenges in utilisation and analysis of IoT data, the interpretation of time-series data and discovery of city events remain among some of the most vital challenges for smooth operation of cities [5].

In smart cities, *collection*, *transmission*, and *processing* of observations in IoT environments should be efficient. To cope with large volumes of data, dimensionality reduction techniques can be applied to reduce the size of the data while maintaining essential information and patterns in aggregated data. This allows the communication overhead in Smart City Frameworks (SCF) to be reduced and facilitates more advance tasks to be performed in large scale, such as clustering, outlier detection and event detection. However, most of the existing approaches transmit raw sensor data constantly [6], [7], [8], [9], [10], [11], [12], [13]. Therefore, novel methods are required to provide aggregated data transmission to save processing power while distributing

1. http://www.odaa.dk

data in the whole network. In the meantime, semantic representation of the aggregations and abstractions are crucial to provide machine-interpretable observations for higher-level interpretations of the real world context.

Reliability, inconsistency and incompleteness are other challenges in smart cities. Since city data is provided by diverse sources, quality of data varies: it may include errors, missing values, and in some cases data collected from citizens (e.g. smart phone sensors or social media) can be biased. Determining and managing trustworthiness of information sources is an important step towards safe and secure operation of smart cities. Provenance information is one of the most important features to keep track of the source of information and assess trustworthiness of different information sources. However, although there are increased numbers of studies in this field [14], [15], [16], there remains a need to develop techniques for real-time monitoring of heterogeneous data streams to detect reliability violations and adapt data acquisition accordingly.

In addition to data aggregation, abstraction and Quality of Information (QoI) issues, smart city applications need to handle discovery and integration of heterogeneous data sources, and extract up-to-date information in real-time to enable further complex processing. Since current semantic service discovery and composition approaches (e.g. WSMO, OWL-S) are based on input, output, precondition and effect, they are not well suited to describe complex event processing services with event patterns and to cope with dynamic and resource constrained data streams.

In this study, we present a novel framework developed within the EU FP7 CityPulse project[2] for real-time data stream analysis using data aggregation, quality analysis, and optimisation techniques to improve the performance of real-time semantic integration of data streams and complex event processing. Using a real-world traffic dataset that is collected from the City of Aarhus, we evaluate the performance of the framework in three parts:

- data aggregation using Symbolic Aggregate Approximation (SAX) algorithm;
- quality analysis of data streams based on analysis of single data streams as well as evaluation of multiple data stream sources describing the city traffic flow.
- semantic integration of data using the estimation and assessment of Quality of Service (QoS) for event service compositions using a Genetic Algorithm (GA);

The remainder of the paper is organised as follows. Section 2 describes the related work. Section 3 demonstrates the proposed framework and semantic annotation of each stage including data federation, aggregation, abstraction, and reliability processing of data stream. Section 4 provides a use case scenario and evaluations of the proposed framework. Section 5 details a discussion and describes the future work.

2 Related Work

In this section, we present the existing smart city and IoT frameworks and we discuss the related work in the three main aspects of the study, namely, i) semantic annotation & data aggregation, ii) QoI, stream dependency and correlating information sources, and iii) semantic data integration and complex event processing.

2.1 Smart City Frameworks

Several IoT frameworks, and in particular smart city frameworks, have been developed in recent years, enabling access to sensory data and interaction with sensors. One of the pioneers in IoT platforms is Global Sensor Networks (GSN) [17] which offers a federation of sensor networks by means of XML-based deployment descriptors that homogenised the sensory data. More recently, the standard *de facto* Iot architecture, IoT-A [18], has provided a semantic description based on the concept sensing-as-a-service that other platforms have adopted, such as IoT-est. IoT-est [10] is a dynamic service and test framework, that allows the creation of dynamic composed services that provides access to heterogeneous sensory data. OpenIoT [9] is an instantiation of IoT-A [18], and similar to IoT.est, it uses the concept of sensing-as-a-service, and provides cloud-based IoT services including a middleware platform and tools for application development. It provides access to an enhanced version of GSN (X-GSN) and uses semantics for better interoperability, based on the IoT-A model. Some well-known examples of this architecture that are focused solely on providing a data platform, semantic modelling and/or semantic sensor discovery are Km4city [6], SmartSantander [7], Spitfire [8], OpenIoT [9], IoT.est [10], OpenCube [11], iCore [12]. StartCity [13] semantically annotates and aggregates traffic data to predict spatio-temporal traffic conditions. The ontology here is domain specific (traffic domain), without generic concepts that can be reused for different kinds of domains. The aggregation of the data is based on traditional aggregation methods (e.g. average, maximum, minimum). OneM2M[3] is working on a new standard for machine-to-machine communications, covering aspects such as protocols, security, services, and data management. However, data aggregation at the framework layer is not supplied, at least in the current version of OneM2M. This initiative is also studying the possibility of adding semantics to the standard through the first draft of SAREF [19]. In this current version SAREF only covers household appliances at a physical level. A meta model was built by [20] in order to annotate the sensory data. While this study introduced numerous metadata to represent sensory observations and devices, it is suffers from not having any links to one of the well-known W3C Semantic Sensor Network Ontology[4]. There has been a recent study such as [21] where authors tested the performance of their IoT middleware based on memory consumption and efficiency. It was particularly tested on smart phones. Although the purpose of this paper also includes efficient communication in IoT systems, developing a unified middleware application is beyond the scope of our study. Recently, there has been similar studies in the smart city domain, such as [22], [23]. However, while these studies contributed to the smart city systems in their

2. http://www.ict-citypulse.eu/page/

3. http://www.onem2m.org/

4. https://www.w3.org/TR/vocab-ssn/

own right, none of these studies tackle the semantic interoperability, data aggregation, complex event processing and quality of information all at the same time in the semantic web environment for smart city systems.

Nonetheless, none of these platforms involve ontologies to handle generic annotations for heterogeneous data streams nor an effective time-series data analysis approach for the sensory domain. Consequently, there still remains a need for efficient semantic knowledge representation and adaptive aggregation of sensory observations in dynamic environments such as smart cities that handles real-time or almost real time annotations.

2.2 Time-Series Data Aggregation and Abstraction

Regarding the time-series analysis, there is a need for much smaller storage space and faster processing due to increasing size and channels of available time-series data streams. Data reduction often serves as the first step in an effort to keep good-quality synopsis of data. *Lower bounding* principle assures preserving the meaning of data by keeping the distance between two time-series data streams in the reduced space (i.e. aggregated data) less than or equal to the true distance of time-series data (i.e. the original data). Some of the well-known algorithms for numerical representation of time-series analysis that accomplish good quality of data reduction and lower bounding principle are Discrete Fourier Transform (DFT) [24], Discrete Wavelet Transform (DWT) [25], [26], Singular Value Decomposition (SVD) [27], Piecewise Aggregate Approximation (PAA) [28]. DFT enables to transform time-series data into the frequency domain by obtaining a single complex number, called Fourier coefficient for each signal by the superposition of a finite number of sine (and/or cosine) waves. Once it has been transformed into the frequency domain, it allows the Fourier coefficients with high amplitude to be obtained and discards the low amplitude coefficients for data compression without much loss of information. However, although it satisfies the lower bounding principle by using *Parseval's Theorem*, the computational complexity (i.e. $C(n^2)$ time, or $C(n \log n)$ time with algorithm in [29]) and choice of the best number of coefficients are the main challenges of this algorithm [30]. While Fourier coefficients always represent global contributions of the data, DWT represent small, local segments of the data with a less computational complexity ($C(n)$) and satisfies lower bounding principle. Its limitation is the data length must be a power of two ($n = 2^m$). In parallel, SVN can also perform dimensionality reduction by optimally transforming a dataset into a new k-dimensional dataset based on the first ordered k-biggest singular values. However, it demands to use of an entire dataset prior to transformation to perform dimensionality reduction, and it cannot work incrementally since a new data insertion requires a new global computation. On the other hand, PAA is a very simple and strongly competitive method compared to more sophisticated transforms such as DFT and DWT. The computational complexity is low ($C(n)$), and supports lower bounding principle, which can be simply calculated using the distances on PAA representation.

Contrary to the numerical approaches, the discretisation of the original data into symbolic strings has not been considered in great detail. Even though it seems a straightforward solution, it comes with substantial advantages over existing algorithms and data structures that enable the efficient manipulations of symbolic representations in addition to allowing the framing of time. However, while general symbolic representation methods are not capable of calculating distance in symbolic space and supporting lower bounding at the same time [31], [32], SAX is the most known symbolic representation technique on time series data mining [33]. Due to the fact that it is based on PAA, it is not computationally expensive, and ensures both considerable dimensionality reduction and lower bounding support. Therefore, we use the SAX algorithm to obtain reduced space of time series data.

2.3 Quality of Information, Stream Dependency and Correlating Information Sources

When processing data streams from external sources an often underestimated part is the evaluation of the quality of the provided data. In [34] Wang and Strong describe the impacts of faulty information. Their work is focused on the utilisation of large scale databases that contain data from multiple data sources, where it has been pointed out that faulty, incorrect or incomplete data can cost billions of dollars. To describe possible issues affecting the utilisation of information, they defined four categories with a list of dimensions for data quality. These categories are used to describe the quality of fixed data sets (in comparison to smart city data streams) like database. In this study a (re)evaluation of the QoI for incoming sensor observations is envisaged, reflecting the dynamics of smart cities.

While the work by Wang and Strong is the theoretical cornerstone for the categorisation of quality metrics, Stvilla *et al.* presented a framework for the assessment of information quality in [35]. The framework is designed as an abstract model and can be extended for specific quality measurement settings. However, it requires a domain expert to define and implement quality measurement rules for each data set. Thus, it is not feasible in a smart city context, where new data streams need to be integrated fairly quickly and frequently. This requires a generic approach that automatically adapts to new data streams.

A development towards application or context independent quality measurement is described by Bisdikian *et al.* [36]. To describe both application dependent as well as application independent quality metrics, they divided the quality analysis into two main parts: namely QoI and Value of Information (VoI). A UML-based data model is introduced supporting both parts of the quality analysis in order to provide a general template for organising QoI/VoI meta-data allowing an effective exchange of QoI/VoI data in a repeatable, consistent and reproducible manner. However, it lacks machine interoperable data format.

2.4 Complex Event Processing and QoS-aware Event Service Composition

Event processing systems are crucial for smart city applications to extract high-level and complex situations from low-level information. Traditional event notification systems only allow filtering over primitive level events, where each

event is considered an individual entity [37]. As an advancement of the traditional event notification systems, complex event processing systems can filter and correlate multiple events by matching a specific pattern [38], [39]. Harnessing over the benefits of semantic technologies, ontology-based event specification has been presented in [40]. However, most of the existing complex event ontologies [41], [42] lack expressivity to describe a broader range of complex event operators. Moreover, these ontologies do not include the semantic description of non-functional properties within specifications of complex events, hence hinder the implementation of a quality-aware complex event composition.

QoS-aware service composition has been studied extensively. The first step of solving the QoS-aware service composition problem is to define a QoS model, a set of QoS aggregation rules and a utility function. Existing works have discussed these topics extensively, e.g., in [43], [44], [45]. However, the aggregation rules in existing works focus on conventional web services rather than complex event services, which need a different QoS aggregation schema. For example, the event engine also has an impact on the QoS aggregation, which is not considered in conventional QoS aggregation for services. Also, the aggregation rules for some QoS properties based on event composition patterns is different to those based on workflow patterns (as in [44]).

As a second step, different concrete service compositions are created and compared with regard to their QoS utilities to determine the optimal choice. To achieve this efficiently, various heuristic approaches are developed. There are two prominent strands: Integer Programming (IP) (e.g., [43], [46], [47]) and Genetic Algorithm (GA) (e.g., [48], [49], [50], [51]) based solutions.

In [46] the limitation of local optimization and the necessity of global planning are elaborated. The authors propose to address the complexity problem of global planning by introducing an IP-based solution with a Simple Additive Weighting (SAW) based utility function to determine the desirability of an execution plan. This approach is extended in [47] with more heuristics to promote efficiency. In [43] a hybrid approach of local and global optimization is proposed, in which global constraints are delegated to local tasks, and the constraint delegation is modeled as an IP-based optimization problem. The problem with IP-based solutions is that they require QoS metrics to be linear, and they do not address the service re-planning problem.

In [48] the chromosomes are encoded with binary bits representing whether a service is selected or not. The problem with this approach is that the readability of the genomes are poor and the chromosome length is not fixed during evolution. In [49] the authors use a different encoding approach, which leads to a fixed chromosome length. In [50] a two-dimensional genome encoding is proposed to express all execution paths while considering task relations, but crossover and mutation need validation. In [51], the authors use tree coding chromosomes, crossovers operate on sub-trees and mutation operate on leaf nodes to avoid invalid reproductions. In [52] the authors develop a GA-based approach that goes beyond QoS-aware composition and enables compliance-aware service composition.

The above GA-based approaches can only evaluate service composition plans with fixed sets of service tasks (abstract services) and cannot evaluate composition plans which are semantically equivalent but consists of different service tasks, i.e., service tasks on different granularity levels. A more recent work in [45] addresses this issue by presenting the concept of Generalized Component Services (GCS) and developing the GA encoding techniques and genetic operators based on GCS. Results in [45] indicate that up to 10% utility enhancement can be obtained by expanding the search space. Composing events on different granularity levels is also a desired feature for complex event service composition. However, the work described in [45] only caters for *Input*, *Output*, *Precondition* and *Effect* (IOPE) based service compositions, which creates composition plans as imperative workflows. Complex event service composition creates declarative event patterns (transformed from the requested event pattern with the same event semantics) as composition plans. It requires an *event pattern* based reuse mechanism [53] and as a result, different genetic encoding mechanisms and crossover operations are needed.

3 Large Scale Data Analysis with CityPulse Framework

In this section, we present a large-scale data analysis framework for smart cities. The proposed framework enables semantic annotation and analysis of IoT and social media data streams by taking into account dimensionality reduction and reliability processing. It involves the following units: a) virtualisation, b) federation, c) aggregation, and d) reliable information processing. While the first three units are part of the large scale data analysis component, the last unit has been divided into atomic and composite monitoring units. The atomic reliable information processing components are called Quality of Information (QoI) units assessing information quality in the first appearance of the data in the IoT system. Figure 1 depicts the main components and basic workflow of the framework.

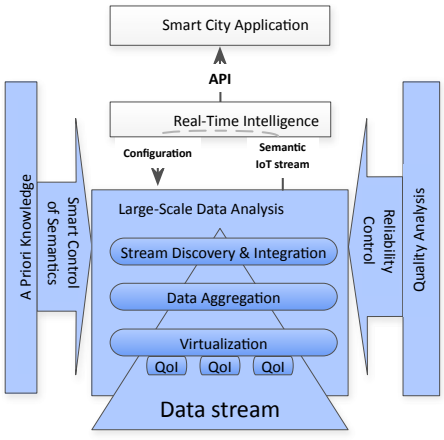


Fig. 1: Architectural overview of the CityPulse framework. The examined units have been illustrated in shaded areas.

Initially, the sensor nodes transmit either raw or aggregated data to the virtualisation component. After collecting the data, it forwards it to three components of the system,

namely, *semantic annotation*, *data federation*, and *reliable information processing*. The data federation component discovers and composes data streams to answer queries over multiple streams in real-time. In parallel, the pattern creation and discretisation process is applied in the data aggregation component. Afterwards, the event detection component performs abstraction process. The abstracted data is finally accessible through the middleware where different layers such as real-time intelligence layer or graphical user interface can access the data. In parallel, the reliable information processing aims at the annotation of information sources with QoI. Therefore, it conducts active evaluation of QoI for data sources and their steady adaption triggered by events that could be sent by the monitoring components and the event management components of other CityPulse framework modules. Figure 2 illustrates an example snapshot from the 3D visualisation tool of the CityPulse framework.

3.1 Virtualisation

The virtualisation component facilitates access to heterogeneous data sources and infrastructure concealing the technical facets of data streams such as location, storage structure, access format, and streaming technology. The system designates various wrappers to encompass a large number of input formats, while it provides a unified format as output (i.e. RDF or Turtle). Describing the obtained sensor data stream for interoperability or facilitated search is the core objective of this component. However, the amount of IoT data streams can be voluminous, while the details often provided by resource constrained devices with limited bandwidth, memory or power. Therefore, the information model that is being used by the IoT and smart city frameworks not only needs to explicitly represent the meaning and relationships of terms in vocabularies but also should be lightweight in order to reduce the traffic and processing time. In our experiments, we used CityPulse information models, namely, Stream Annotation Ontology[5], Quality Ontology[6], Complex Event Processing Ontology[7] for semantic representation of data aggregation, quality, and complex event services, which are extensions of existing ontologies such as W3C SSN and other data annotation frameworks to describe the streams and their resources as well as quality related features of the data. In the implementation stage, we used a semantic annotation library, called SAOPY[8], which involves the ontologies given above along with state-of-the-art IoT-related ontologies.

3.2 Data Aggregation and Abstraction

SAX algorithm [33] transforms a time-series into a discretised series of letters e.g. a word. It divides a time series data into equal segments and then creates a string representation for each segment. The algorithm involves 3 main steps, namely normalisation, PAA and discretising of the aggregated data. Initially, time series data is normalised to have a mean of zero and standard deviation of one before converting it to PAA. Afterwards, PAA divides the original data into desired number of windows and calculates the average amount of data falling into each window. This results in a reduction of data size. A shorter window length n results in a better reconstruction of the original data, however more data space is needed to store the data and eventually higher energy consumption by higher communication costs. The calculation of each window of PAA segments is given as:

$$\overline{Z_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} Z_j, \tag{1}$$

where $\overline{z_i}$ represents the ith element of a time series data, Z, of length n, and w represents the number of segments. This results in a reduction of data size from n to n/w data points. Once time series data transformed into PAA coefficients, symbolising the PAA representation into a discrete string is the final stage. Considering the fact that normalised time series data follows a Gaussian distribution, the discretisation phase allows symbolic representations of data to be obtained by mapping PAA coefficients to breakpoints that are produced according to the alphabet size 'a', which in turn determines equal-sized areas under a Gaussian curve. Table 1 shows the Gaussian breakpoints for values of alphabet size, 'a', from 3 to 10. The definition for breakpoints are given below:

Definition 1. (Breakpoints). breakpoints are a sorted list of numbers $B = \beta, ..., \beta_{a-1}$ such that the area under a N(0,1) Gaussian curve from β_i to $\beta_{i+1} = 1/a$, where β_0 and β_a are defined as $-\infty$ and ∞, respectively.

β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

TABLE 1: Breakpoints that divide a Gaussian distribution in an arbitrary number from 3 to 10 of equiprobable regions

The break lines are distributed vertically according to the Gaussian distribution, the first letter of the alphabet represents the smallest PAA coefficient, 'a', and the greatest PAA coefficient is represented by the last letter of the alphabet. The definition to obtain the symbolic representation is given below:

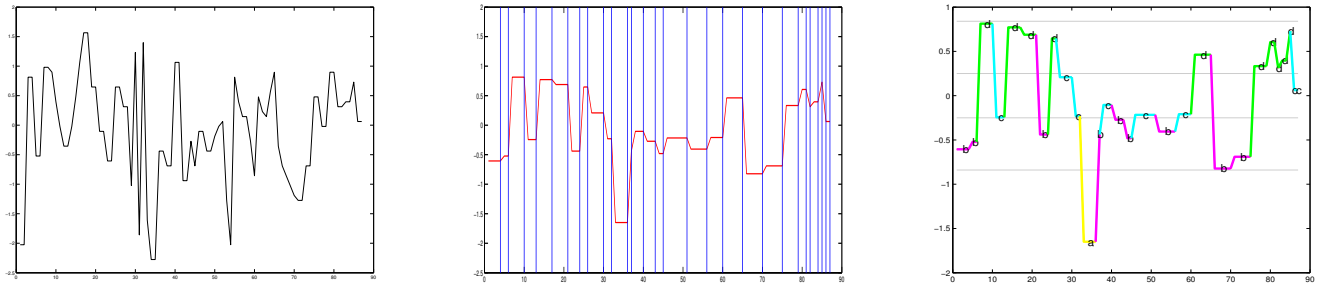
Definition 2. (Word). A subsequence C of length n can be represented as a word $\hat{C} = \hat{c}_1, ..., \hat{c}_w$ as follows. Let β_i denote the $i - th$ element of the alphabet, i.e., β_1 = a and β_2 = b. then the mapping from a PAA approximation $\bar{C}$ to a word $\hat{C}$ is obtained as follows:

$$\hat{c}_i = \beta_j, \iff \beta_{j-1} \leq \hat{c}_i < \beta_j \tag{2}$$

Example 1. Let's assume that we have a time-series data, time-series (c) $= \{2, 3, 4.5, 7.6\}$. Following the steps given above, we apply $z - transform$ and obtain time-series (z) $=\{-0.93, -0.52, 0.09, 1.36\}$. Here we use SAX with window size of '2' and alphabet size of '4': it leads to a set of PAA coefficients of $\{-0.72, 0.72\}$. Finally, we map the PAA

5. http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/sao
6. https://mobcom.ecs.hs-osnabrueck.de/cp_quality/
7. http://citypulse.insight-centre.org/ontology/ces/
8. http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/saopy.html

Fig. 2: An example snapshot from CityPulse 3D map, which illustrates the traffic events that are present on the travel route of a user in the city of Aarhus.



(a) Normalisation (b) PAA (c) Discretisation

Fig. 3: A real time data obtained for average speed from a pair of sensor points are normalised, discretised by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols. In the example above, with n=144, w=6 and a=5, the time series is mapped to the words "bbdcddbdcabcbbcbcdbbddddcc".

coefficients into SAX symbols by using the cut off ranges of β and β_{a-1}, given in Table 1, $\{-0.67, 0, 0.67\}$, and obtain corresponding SAX word $\{ad\}$. Given that the first PAA coefficient is smaller than -0.67 and the second coefficient is greater than 0.67, the former is assigned to 'a' and latter to 'd'.

3.2.1 Semantic Annotation of Aggregated Data

Using SAO, we can describe the time series data further to take into account aggregated features as well as temporal entities such as particular segments of a data stream. Figure 3 shows an exemplification of the creation of SAX patterns obtained from the real-time data — average speed from a pair of sensor points — which are initially normalised using $z - transform$ (Figure 3(a)), segmented using PAA (Figure 3(b)), and mapped to SAX symbols in discretisation phase (Figure 3(c)). Listings 1 depicts an example of describing a PAA segment from the data stream presented in Figure 3, which has been extracted by SAX algorithm with alphabet size of '5' and segmentation size of "2928". First, we identify a data stream, then we describe a timeline instance. This timeline is used to link the segment feature description with the time extent of a temporal entity representing the data stream. Thus, we can express a stream data as a time interval on the universal timeline, and also relate such an interval with the corresponding interval on the discrete timeline. This particular segment represents 30-minute time intervals and a window size of 6. The output of discretisation phase has also been provided with a SAX letter, 'c' for a PAA segment as well as a SAX word as an overall SAX output for the entire data stream using `sao:value` in the RDF document.

3.3 Quality Analysis

Due to the large amount of data it is not feasible to determine the QoI with respect to the application that processes the data. Thus, we use the application-independent approach introduced in [15]. To realise an application independent approach, a new ontology was developed to describe the quality of data sources. Some of the ambiguous definitions of quality concepts (as in section 2) required redefinition in the CityPulse context. Table **??** lists the categories along with their definitions to understand their utilisation in the framework and to avoid misunderstandings caused by different definitions in the referenced literature. It consists of typical quality categories like `Accuracy` or `Timeliness`. These categories may be further divided into sub-categories (e.g. `Completeness` and `Correctness` in the category `Accuracy`). The quality value of an upper-

```
@prefix ces: <http://www.insight-centre.org/ces#> .
@prefix ct: <http://www.insight-centre.org/ct#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix sao: <http://purl.oclc.org/NET/sao/sao#> .

http://unis/trafficData206369Property-0cfcabfd
      a "http://www.surrey.ac.uk/ics#Average_Speed"  ;
      ssn:isPropertyOf <http://unis/trafficData206369FoI-0f6046e9b> .
PAA-Average_Speed4d7d270f-e5e2-4bb9-9559-2a0e2719e63e a
sao:PiecewiseAggregateApproximation ;
      sao:time [       a tl:Instant      ;
                       tl:at "20-Aug-2014  07:29:00"^^xsd:date ;
                       tl:duration "PT30M"^^xsd:duration
                     ] ;
      sao:value "c"^^xsd:string   ;
      tl:onTimeLine <feature_timeline> ;
      prov:wasGeneratedBy <http://unis/trafficData206369Property-0cfcabfd> .
ics:Average_SpeedSaxWord-7d5374cd-d092-4d50-8bd3-ce48c569b7d0 a
sao:SymbolicAggregateApproximation ;
      sao:alphabetsize "5"^^xsd:int ;
      sao:segmentsize "2928"^^xsd:int ;
      sao:time [       a tl:Instant      ;
                       tl:at "01-Aug-2014  07:59:00"^^xsd:date ;
                       tl:duration "P2M"^^xsd:duration
                     ] ;
      sao:value "bbdcddbdccabcbbcbcdbbddddcc"^^xsd:string ;
          prov:wasGeneratedBy <http://unis/trafficData206369Property-0cfcabfd>
feature_timeline a tl:DiscreteTimeline .
feature_timeline_map
      a tl:UniformSamplingWindowMap      ;
      tl:rangeTimeLine <feature_timeline> ;
      tl:windowlength "6"^^xsd:int .
```

Listing 1: Summarised data expressed using the Stream Annotation Ontology.

level category is the combination of the contained lower-level QoI metrics. The intention is to determine the values for the sub-metrics by analysing sensor observations and provide a top-level QoI metrics as a tuple of lower level categories. The `Communication` category contains attributes to describe QoS characteristics. In addition, the categories `Cost` and `Security` contain attributes that describe non-functional static characteristics of the streams important for the end user. Throughout this paper we use `Completeness` and `Correctness` subcategories to represent the `Accuracy` of data streams, and `Frequency`, `Latency` and `Age` to represent the `Timeliness` of the data streams. The following list describes the upper-level categories while the lower-level categories for each of the upper-level categories can be found in Table 2.

- `Accuracy`: Metrics in the `Accuracy` category describe the degree to which delivered information is correct, precise and complete. To determine the `Resolution`, `Deviation`, `Completeness` and `Correctness`, and to allow a comparison with geo-spatial related data streams, a registration of sensor characteristics is necessary.
- `Communication`: As stated earlier the `Communication` category contains attributes related to typical QoS parameters of a data stream. With this category it is possible that an application receives only information from data sources that fulfil the QoS requirements e.g. have only a small packet loss. Example subcategories are: `Bandwidth`, `Latency` or `Throughput`.
- `Cost`: The `Cost` category is required to be able to describe the monetary, energy and network costs of a data stream. This will allow an application (depending on the user's preferences) using the CityPulse framework to potentially find a trade-off between cost-efficient and reliable data source.
- `Security`: This category contains metrics to describe the permission levels for provided data, e.g. if the data owner allows the data to be republished/distributed or prohibits any further usage. Furthermore metrics to describe the encryption and data integrity through signing mechanisms are included in this category.
- `Timeliness`: With the metrics in the `Timeliness` category it is possible to select data streams by their update frequency, the amount of time information is valid (`Volatility`) and the timespan between the data measurements and the publication time (`Frequency`).

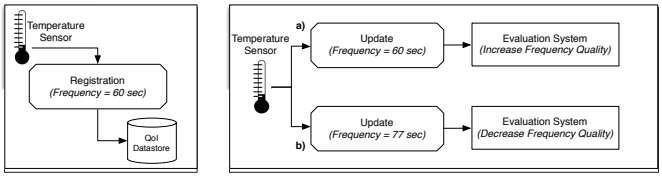


Fig. 4: Quality Analysis - General Approach

To use the full variety of information sources in city deployments, a vast number of heterogeneous interfaces and data formats have to be adapted. Furthermore, the varying information quality and uncertain reliability of multiple information providers has to be considered in the selection of the best available data stream. Therefore, the framework continuously monitors and calculates the QoI of incoming data streams and exploits this information to provide reliable applications.

3.3.1 General Approach

The quality analysis involves multiple quality metrics, which have been introduced above and in Table tab:qualitymetricdefinitions. Figure 4 shows the general approach in calculating the quality of a data source/stream for the update frequency of a simple temperature sensor. In the first step a temperature sensor (temperature data stream) is registered at the framework. Within the registration the frequency of the stream is annotated with '$60s$' which implies that the sensor should deliver an update every 60 seconds. The second part of the graphic shows the update mechanism. If the stream sends an update, the time difference between the current and the last update is calculated. If it fits the registered frequency (example a of Figure 4) the frequency quality increases or stays at the maximum level. The second update (example b) is received with a time difference of $77s$. This denotes that the registered frequency for the stream update is not fulfilled. In this case the frequency quality is decreased.

3.3.2 Correlating Information Sources

The goal of the quality analysis is to produce a metric which indicates the current performance of a sensor node

Category	Metric	Definition
Accuracy	Correctness	Probability that provided data is within the range of precision and completeness.
	Precision	
	Resolution	Resolution detail for the measured value.
	Deviation	The maximum percentage of deviation from the real value.
	Completeness	The ratio of attribute values compared to expected parameters.
Communication	Network Performance	
	Packet Loss	The probability that a set of data / a packet will not be transported correctly from the source to its sink.
	Bandwidth	Min/Avrg/Max amount of bandwidth that is required to transport the stream.
	Latency	Measurement of the time delay between the stream is sent and received in the virtualisation layer.
	Jitter	Deviation from true periodicity of an assumed periodic signal.
	Throughput	The amount of useful information sent by the network (ex: sensor data), without protocol information.
	Queuing	
	Queuing Type	Type of queuing, e.g. FIFO, LIFO, unordered.
	Ordered	Probability that datasets arrive in the defined order.
Cost	Energy Consumption	The amount of energy used to access the steam.
	Monetary Consumption	Is the usage of the stream free of charge or how much does it cost.
	Network Consumption	How much traffic is caused by usage of the data source.
Security	Confidentiality	
	License Definition	Reference to Licence class, e.g. http://creativecommons.org/ns#Licence.
	May Be Used	Reference to Permission class, e.g. http://creativecommons.org/ns#Permission.
	May Be Published	Reference to Permission class, e.g. http://creativecommons.org/ns#Permission.
	Encryption	Encryption method, authority for key management.
	Signing	
	Authority	Certificate authority.
	Public Key	Key to decrypt signatures.
Timeliness	Age	The time an information was created/measured/sensed.
	Frequency	Maximum timespan between two datasets.
	Volatility	The amount of time the information remains valid in the context of a particular activity.

TABLE 2: Quality Metrics Categorisation and Definition

compared with promised quality metrics annotated during the registration process. This metric can be either the absolute value of a specific QoI metric or a normalised value. An advantage of using a normalised value is a) that sensor nodes of different providers can be compared and b) different QoI metrics can be combined to derive an overall reputation metric for a sensor node or network. To calculate normalised QoI values the following algorithm was designed. The output of the algorithm is in the range of 0 and 1. The input is a discrete value and indicates if the last observation update satisfies the annotated quality metric. The output is increased if the input was positive with respect to the output value range, and vice versa. The algorithm considers past behaviour of the sensor node so that continuous reliability results in a higher output. The magnitude of a negative input decreases over time. At the same time it allows the system to fully recover from negative inputs such as sporadic outliers after a series of observation updates. The algorithm requires a constant amount of memory and processing power.

In our quality analysis, we have modified the reward and punishment algorithm introduced in [54]. The witness (i.e. a sensors) agrees if the new value is within the upper and lower bounds. Quality value has been computed based on the majority of agreed witnesses. In our approach we combine the reward and the punishment equations into one equation. It uses a sliding window over the last inputs with window length W. The quality metric is calculated as follows:

$$q(t) = |q(t-1) - 2 * Rd(t)| \tag{3}$$

where $q(t)$ is the quality metric at time t and $q(t-1)$ is the past quality metric. $Rd(t)$ denotes the reward to be added for the current input. The reward is calculated as:

$$Rd(t) = \frac{\alpha^{W-1}(t-1)}{W-1} - \frac{\alpha^{W-1}(t-1) + \alpha^{current}(t)}{W} \tag{4}$$

where α^{W-1} denotes the number of positive entries within the window and $\alpha^{current} \in \{0, 1\}$ the current input.

3.4 Semantic Sensors Stream Discovery & Integration

In this section, the ontology used for describing event services and event requests are presented, the discovery and integration mechanism for the sensor data streams are discussed. Notably, the discovery task given in Section 3.4.1 should not be confused with the integration task described in Section 3.4.2, where former matches most fine-grained and atomic service requests and responses, and the latter deals with service compositions.

3.4.1 Sensors Streams Discovery

The task of sensor stream discovery is to find candidate sensor services based on sensor service descriptions and request specifications. A sensor stream is an atomic unit in IoT stream discovery and integration. It is described both as a *PrimitiveEventService* in CES ontology, as well as a *Sensor* device in SSN ontology. The CES ontology is mainly used to describe the non-functional aspects of sensor service requests/descriptions, including sensor event types, quality parameters and sensor service groundings. SSN ontology is used to describe the functional aspects, including *ObservedProperties* and *FeatureOfInterest*.

A sensor service description is denoted as $s_d = (t_d, g, q_d, P_d, FoI_d, f_d)$, where t is the sensor event type, g is the service grounding, q_d is a QoS vector describing the QoS values, P_d is the set of *ObservedProperties*, FoI_d is the set of *FeatureOfInterests* and $f_d : P_d \rightarrow FoI_d$ is a

```
@prefix ces: <http://www.insight-centre.org/ces#> .
@prefix owls: <http://www.ai.sri.com/daml/services/owl-s/> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema> .

:sampleTrafficSensor a ssn:Sensor,ces:PrimitiveEventService;
    owls:presents :sampleProfile ;
    owls:supports :sampleGrounding;
    ssn:observes [ a ces:AverageSpeed;
                   ssn:isPropertyFor :FoI_1],
                 [ a ces:VehicleCount;
                   ssn:isPropertyFor :FoI_2],
                 [ a ces:EstimatedTime;
                   ssn:isPropertyFor :FoI_3].
:sampleProfile a ces:EventProfile ;
    owls:serviceCategory [ a ces:TrafficReportService ;
    owls:serviceCategoryName "traffic_report"^^xsd:string].
```

Listing 2: Traffic sensor service description

```
@prefix ces: <http://www.insight-centre.org/ces#> .
@prefix owls: <http://www.ai.sri.com/daml/services/owl-s/> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema> .

:sampleRequest a ssn:Sensor,ces:EventRequest;
    owls:presents :requestProfile ;
    ssn:observes [ a ces:EstimatedTime;
                   ssn:isPropertyFor :FoI_3].
:requestProfile a ces:EventProfile ;
    owls:serviceCategory [ a ces:TrafficReportService ;
    owls:serviceCategoryName "traffic_report"^^xsd:string].
```

Listing 3: Traffic sensor service request

```
@prefix ces: <http://www.insight-centre.org/ces#> .
@prefix owls: <http://www.ai.sri.com/daml/services/owl-s/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema> .

:SampleEventRequest a ces:EventRequest;
   owls:presents :SampleEventProfile.

:SampleEventProfile a owls:EventProfile;
   ces:hasPattern [ a ces:And, rdf:Bag;
                rdf:_1 :locationRequest;
                rdf:_2 :seg1CongestionRequest;
                rdf:_2 :seg2CongestionRequest;
                rdf:_4 :seg3CongestionRequest;
                ces:hasWindow "5"^^xsd:integer];
   ces:hasConstraint [ a ces:NFPConstraint;
                  ces:onProperty ces:Availability;
                  ces:hasExpression
                 [ emvo:greaterThan "0.9"^^xsd:double]],
                 [ a ces:NFPConstraint;
                  ces:onProperty ces:Accuracy;
                  ces:hasExpression
                 [ emvo:greaterThan "0.9"^^xsd:double]].
```

Listing 4: Complex event service request

function correlating observed properties with their feature-of-interests. Similarly, a sensor service request is denoted

$$s_r = (t_r, q_r, P_r, FoI_r, f_r, pref, C)$$

. Compared to s_d, s_r do not specify service groundings, q_r represents the constraints over QoS metrics, $pref$ represents the QoS weight vector specifying users' preferences on QoS metrics and C is a set of functional constraints on the values of P_r. s_d is considered a match for s_r *iff* all of the following three conditions are true:

- t_r subsumes t_d,
- q_d satifies q_r, and
- $\forall p_1 \in P_r, \exists p_2 \in P_d \implies T(p_1)$ subsumes $p_2 \wedge f_r(p_1) = f_d(p_2)$, where $T(p)$ gives the most specific type of p in a property taxonomy.

Listing 2 shows a snippet of a traffic sensor description in Turtle syntax. The traffic sensor monitors the estimated travel time, vehicle count and average vehicle speed on a road segment (annotated as observed properties). As a sensor service, it presents a service profile (`:sampleProfile`) that describes the service category (`ces:TrafficReportService`) for discovery and supports a service grounding (`:sampleGrounding`) for invocation. Listing 3 shows a snippet of a sensor service request matched by the traffic sensor service. Compared to Listing 2, the request does not contain a grounding. The property `ces:EstimatedTime` is requested on the feature-of-interest `:FoI_3`. Also, the requested service type (`ces:TrafficReportService`) is an exact match for the service type in Listing 2, making the service described in Listing 2 a matching service candidate for the request. When the discovery component finds all candidate services suitable for the request, a Simple-Additive-Weighting algorithm [55] is used to rank the service candidates based on q_d, q_r and *pref*.

Alternatively to the above discovery approach implemented by a middleware that parses the requests and service descriptions, a user can also perform some basic discovery function using Simple Protocol and RDF Query Language (SPARQL)[9] . Listing 5 shows a sample SPARQL query that identifies sensor services by querying the properties they measure. The query results shall contain the service in Listing 2, if the `ces:AverageSpeed` is annotated as a sub-class of `ces:Speed`.

```
prefix ces: <http://www.insight-centre.org/ces#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn>

SELECT ?sensorService
WHERE { ?sensorService ssn:observes ces:Speed. }
```

Listing 5: Simple discovery via SPARQL

3.4.2 Sensors Streams Integration

Sensor stream discovery deals only with primitive event service discovery. To discover and integrate (composite) sensor streams for complex event service requests, the event patterns specified in the complex event service requests/descriptions need to be considered.

We insert the rule in Listing 6 into the Complex Event Pattern Ontology, to allow reasoners to entail sub-pattern relationships. Notice that `rdfs:member` is the super-property for the container membership property (i.e., `rdf:_1`, `rdf:_2` ...) in RDF Schema version 1.1.

In the context of integrated sensor stream discovery and composition, the definition of sensor stream description is extended to denote composite sensor stream descriptions

9. http://www.w3.org/TR/rdf-sparql-protocol

```
[Rule1: (?x rdfs:member ?y) -> (?x ces:hasSubPattern ?y)]
```

Listing 6: Entail sub-patterns via RDF containers.

$S_d = (ep_d, Q_d, G)$, where ep_d consists of a set of (primitive or composite) sensor stream descriptions, and a set of event operators including *Sequence*, *Repetition*, *And*, *Or*, *Selection*, *Filter* and *Window*, q_d is the aggregated QoS metrics for S_d and G is the grounding for the composite sensor stream. Similarly, a complex event service request is denoted as $S_r = (ep_r, Q_r, pref)$, where ep_r is a *canonical* event pattern consisting of a set of primitive sensor service requests and a set of event operators, Q_r describes the QoS constraints for the requested complex event service and *pref* specifies the weights on QoS metrics.

An S_d is a match for S_r *iff* ep_d is *semantically equivalent* to ep_r and Q_d satisfies Q_r. When no matches are found during the discovery process for S_r, it is necessary to compose S_r with a set of primitive or composite sensor streams which are *reusable* to S_r. Informally, these (composite) sensor streams describe part of the semantics of ep_r and can be reused to create a composition plan, which contains an event pattern with concrete service bindings. The composition plan can be used as a part of the event service description for the composed event service. The discovery or composition results can be ranked w.r.t the QoS metrics and preferences in the same way as sensor stream discovery. We refer readers to [53], [55] for detailed definitions of concepts related to event patterns as well as algorithms to perform an efficient pattern-based and QoS-aware event service discovery and composition. Listing 4 shows a snippet of a sample complex event service request with an event pattern and some NFP constraints. The requested pattern is a conjunctive (`ces:And`) correlation between four requested primitive events (e.g., `:locationRequest`, `:seg1CongestionRequest`, etc.). The QoS constraints in the request ask for the `ces:Availability` and `ces:Accuracy` to be above 90%.

Leveraging the sub-pattern property in CES ontology, one can query the provenance relations (within the same `EventProfile`) specified in composition plans (as well as other event patterns) using the query specified in Listing 7, with OWL reasoners (augmented with the rule in Listing 6). In order to track provenance relations among different event profiles, additional rules in Listing 8 must be used.

```
prefix ces: <http://www.insight-centre.org/ces#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn>

SELECT ?subpattern
WHERE { :SampleService owls:presents ?sampleProfile.
      ?sampleProfile ces:hasPattern ?pattern.
      ?pattern ces:hasSubPattern ?subPattern. }
```

Listing 7: Tracking pattern provenance via SPARQL

```
[Rule2: (?ep1 ces:hasSubPattern ?s)
       (?s owls:presents ?p)
       (?p ces:hasPattern ?ep2)
       -> (?ep1 ces:hasSubPattern ?ep2)]
```

Listing 8: Entail sub-patterns among different event services.

4 EVALUATIONS

To evaluate the system, we examine each component using the dataset that has been published by the city of Aarhus. It contains various sensor data including 449 pairs of traffic sensors, and one weather sensor. We use these real sensors as the basis of our experiment dataset. We also simulate 449 air pollution sensors, each one is hypothetically deployed next to a traffic sensor to report the Air Pollution Index at that location. In addition to the ODAA datasets, we also use the Here Traffic[10] dataset provided by Nokia[11] for analysis of QoI. The evaluations are organised in three parts as follows:

- *Data Aggregation*: we evaluate the performance of the SAX algorithm on time series traffic data obtained from the city of Aarhus. We examine the performance difference between the virtualised/annotated raw data observations and the results obtained with SAX using various segmentation sizes. We evaluate our system based on two main criteria: (i) data reduction and (ii) data reconstruction error rate.
- *Quality Analysis*: the evaluations involve two parts: (i) single data stream and (ii) multiple dependent data stream. In the first part, we provided a comprehensive technical analysis of the integrity of a single data stream, and further evaluated our system with a multiple dependent data approach comparing ODAA data streams against the Here Traffic dataset to analyse the incidents that occurred in the city of Aarhus.
- *Semantic discovery and integration of data streams*: we create different user requests about traffic monitoring on different routes in Aarhus city and use the discovery and composition algorithms to find the optimal combination of sensors to fulfil the requests. We test the performance of the sensor discovery and quality-aware composition algorithms in terms of execution time and the quality of composition results. After transforming these composition results into complex event processing queries, we test the performance of evaluating the queries based on raw and aggregated data streams.

4.1 Smart City Traffic Management Scenario

The city of Aarhus has deployed a set of traffic sensors on the streets. These sensors are paired as start nodes and end nodes. Each pair is capable of monitoring the average vehicle speed v and vehicle count n on a street segment (from the start node to the end node). Combined with the

10. https://www.here.com/
11. http://nokia.com

distance d between the two sensors, the estimated travel time $t = d/v$ and congestion level $c = n/d$ can be easily derived. In addition to traffic sensors, there are also sensors such as weather sensors and air pollution sensors deployed in the city, and a user can plan his/her travel taking different physical measurements into consideration. For example, some may be travelling on a tight schedule so they need the route with lowest estimated travel time, while some others maybe cycling and they need routes with less cars and better air quality. Despite the functional requirements, different users may have different non-functional requirements as well.

4.2 Data Aggregation

Prior to presenting the evaluation of the data discovery and integration, we present evaluation of our approach for data aggregation. To examine the performance of the aggregation method, we applied four different segmentation sizes, namely 30 mins ($S30m$), 1 hour ($S1h$), 2 hours ($S2h$), and 4 hours ($S4h$) to divide dataset into frames with alphabet size of 5. We measured the average reconstruction rate using Euclidean distance measured between original and reconstructed data, and examined whether or not there was a significant difference by using Analysis of Variance (ANOVA) test. Afterwards, we calculated the data size and analysed the effects on the RDF representations of the raw and aggregated sensor observations. The evaluations were performed on a Personal Computer (PC) running Ubtuntu 14.04 operating system with an Intel Core i5-3470 3.2GHz processor and 8GB RAM memory. The performance of the system based on different segmentation sizes is reported in Figure 5.

Results show that the lowest data reconstruction error was obtained for the $S30m$, where average speed was in range of $11.2 \leq \Delta\ S30m \leq 12.9$, and vehicle count was in range of $2.4 \leq \Delta\ S30m \leq 3.4$. For the $S1h$, there was an increase in reconstruction error rate ($17.7 \leq \Delta\ S30m \leq 19.5$ for average speed, and $4.0 \leq \Delta\ S30m \leq 5.0$ for vehicle count). Compared to $S30m$, it is worth pointing out that there was a (highly significant) difference between $S30m$ and $S1h$ ($p \leq .05$). Subsequently, highly significant differences were found between $S1h$ and $S2h$ ($p \leq .001$), and $S2h$ and $S4h$ ($p \leq .001$) in terms of error rate. Moreover, the error rate continued to increase in parallel to the number of segment sizes. For, $2h$, the error rate was between $27.2 \leq \Delta\ S2h \leq 29.0$ for average speed, and $6.6 \leq \Delta\ S2h \leq 7.6$ for vehicle count, and was highest for $S4h$, $39.7 \leq \Delta\ S4h \leq 41.3$ for average speed, and $9.9 \leq \Delta\ S4h \leq 10.9$ for vehicle count.

On the contrary, the increase in segmentation size had a different effect on the data size. Although the quality of the aggregated data reduced with the increasing size of segmentation, there was (highly significant) data reduction for the annotated data streams. While the annotated raw data was initially between $60.6MB \leq \Delta\ RawData \leq 60.9MB$, with the segmentation of $S30m$ data size (significantly) dropped to $3.3MB \leq \Delta\ S30m \leq 3.5MB$ ($p \leq .001$). There were also (highly significant) differences for all other parameters ($p \leq .001$). As a result, the data size was reduced to $1.8MB \leq \Delta\ S1h \leq 2.1MB$ for the $S1h$, $.9MB \leq \Delta\ S2hm \leq 1.2MB$ for the $S2h$, $.5MB \leq \Delta\ S4hm \leq .7MB$ for the $S4h$. Overall, data size was reduced in range of 94.24% to 99.8% depending on segmentation size.

Consequently, we found that segmentation size has a (highly significant) impact on the data size and the quality of the data reconstruction for time series data. However, since the change in segmentation size causes opposite effect on the data size and reconstruction error rate results, the selection of the right parameters plays an important role as a trade off. However, it is worth to point out that our system has to operate in real-time, and our aim is not only to obtain good performance in terms of minimum error in reconstruction rate and data size, but also to execute complex event processing on top of the obtained outputs once the data streams are aggregated. Therefore, it was interesting to see that although there was significant difference between segmentation size, there was no significant difference of the data aggregation results on the complex event processing as we will detail in Figure 13(b) in section 4.4.

4.3 Quality of Information - Analysis

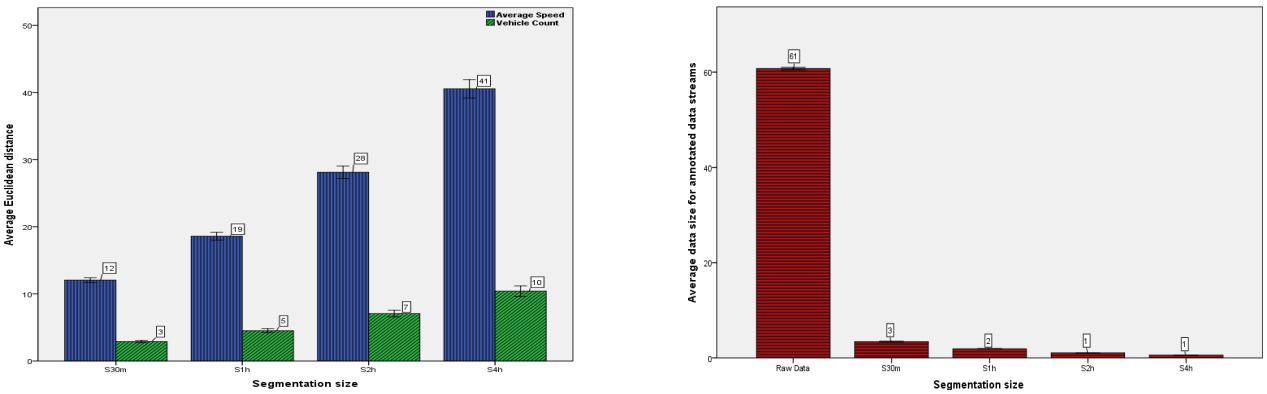
This section presents experimental results regarding the quality of sensor observations in the city of Aarhus (Denmark). The evaluation includes both single data and multiple data stream sources describing the city's traffic flow. In the evaluation we took into account subcategories of the two main categories (Accuracy and Timeliness) in the QoI description. Rated values mentioned in this section refer to the output of the reward and punishment algorithm introduced in section 2. The following evaluations were performed on a PC with a Core2 Duo CPU 2.80GHz and 4GB memory running an Ubuntu 14.04 operating system.

4.3.1 Quality Evaluation of Single Data Streams

In the experiments, we used the following subset of QoI metrics, which are introduced in Table 2 and calculated as follows in the context of Aarhus traffic data:

Completeness (`Accuracy`): Each observation within the data stream is composed of a set of features. An observation is considered complete if all features are present and contain valid values. The validity of a feature has to be defined depending on the context. For example, *is an empty string a valid attribute or the absence of it?* The Aarhus traffic sensor network in our experiment used a web service as a bridge to provide access to the sensor network. Direct access to the sensor nodes was not possible. Since the web service always delivered the last observations, completeness in our experiment was always 100%, unless the web service itself was not reachable. The omitted observations only affected the Frequency metric.

Frequency (`Timeliness`): We calculated just over 4 million frequency QoI annotations for 1 month of traffic data streams. Figure 6 shows the distribution of QoI values annotated during the experiment. As shown in the Figure, about 75% have been rated with 0.95 or higher and almost 95% have been rated 0.8 or higher. Figure 7 shows that about 50 of the 449 sensor nodes have an average frequency QoI value of 0.9 or less, but still over 0.85. About 290 sensor nodes more than half of the sensor nodes observed during the experiment, could reach an average frequency QoI of 0.95 or higher.



(a) Data reconstruction error rate

(b) Data size

Fig. 5: Summary of the evaluation results for the average reconstruction rate of SAX over different segment sizes depicted in 5(a), and the data size of the aggregated data with various segmentation sizes after RDF serialisation in 5(b). The bars refer to the following evaluation metrics: euclidean distance between original and reconstructed data and average data size in MB based on segmentation with every 30 minutes ($S30m$), segmentation with every hour ($S1h$), segmentation with every two hours ($S2h$), segmentation with every 4 hours data ($S4h$).

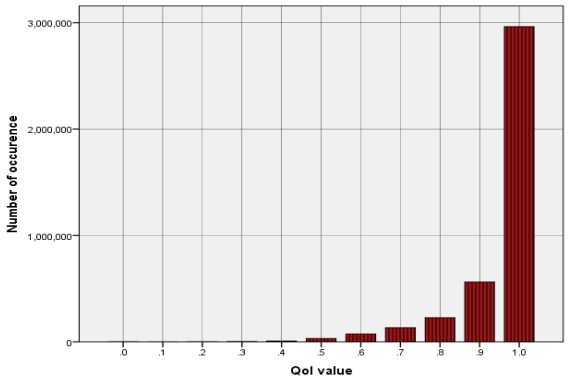


Fig. 6: Distribution of QoI values for the QoI Frequency metric .

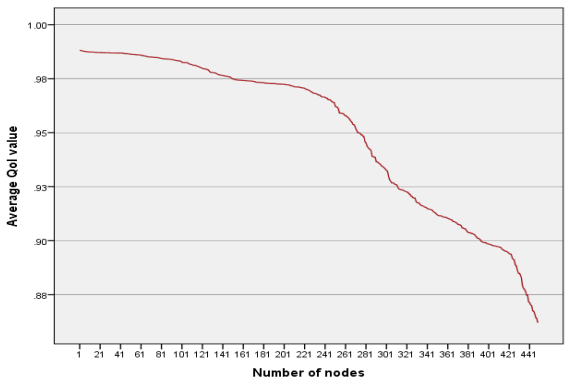


Fig. 7: Average for the QoI metric and how many sensor nodes could reach at least this average.

Correctness (`Accuracy`): It is a well-known fact that in the meteorology community, it is very challenging to measure real world physical values. Many error sources, such as noise, drift and aging of the sensing equipment, influence the measurements in a random and unpredictable way. Depending on the available information, different strategies to compensate for those influences can be applied. Investigating a single sensor observation limits the validation to check for possible and probable value ranges. A series of observations of a single sensor allows the system to check if the sensor got stuck at a certain value or to detect outliers (such as sudden spikes). In case of multiple available sensors, a common strategy is to use redundant sensors, measuring the same or a correlated value and to perform validations between both sensors. In our experiment the three nearest neighbouring sensors are used to evaluate the correctness of each observation. Figure 8 depicts the distribution of annotated QoI values for the Correctness metric.

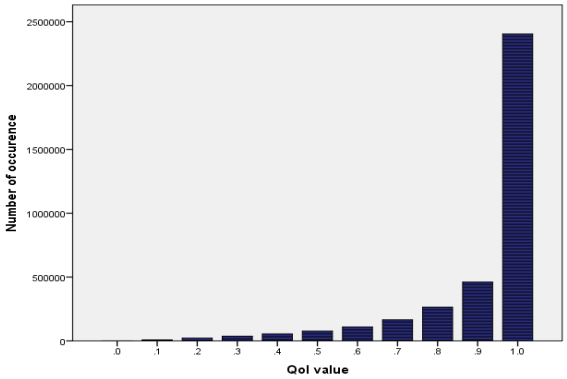


Fig. 8: Distribution of QoI values for the QoI metric Correctness.

4.3.2 Quality Evaluation of Multiple Dependent Data Streams

The evaluation of single data streams enables a comprehensive technical analysis of the integrity of the stream, whereby the analysis of the individual data values cannot be assured if there are no directly comparable data streams available. The utilisation of corresponding high level information (events) published by another issuer through a

distinct data stream enables discovery of consistent and contradictory information. To evaluate the *average speed* and *vehicle count* data streams, which are published via ODAA platform, the traffic incident information[12] for congestion and closed roads has been investigated to get a confirmation of reported traffic events. Therefore, the timespan of the incident is considered as a period, where the traffic flow should change. The incident data set describes the start time t_{Istart} and the end time t_{Iend} of the congestion incident, which defines the incident period $p_I = [t_{Istart}, t_{Iend}]$ with the duration d_I of the incident. To evaluate the dependency between the streams, time series of the average speed is investigated for $p_{Before} = [t_{Istart} - d_{Incident}, t_{Istart})$ before and for $p_{After} = (t_{Iend}, t_{Iend} + d_{Incident})$ after the congestion period $p_{Incident}$. To remove daily repeating traffic patterns and analyse the trend of the time series independently of the seasonal components, the seasonal components were removed. The quartiles $Q1$, $Q2$ and $Q3$ (the 25th, 50th (median) and 75th percentiles) of p_I are now compared to p_{Before} and p_{After} (see Figure 9). If the majority of quartiles leads to an increase of the average speed from p_{Before} to p_I and a decrease from p_I to p_{After} we detect a correlation in the detected events of our data streams which increases their reputation and consolidates their Correctness level. The evaluation of the comparison of 25 distinct traffic incidents against the ODAA data shows only a 76% confirmation of the traffic incident reports using the raw data stream. By taking into account the results of the Frequency QoI analysis (see section 4.3.1) missing data can be identified as a cause of this low confirmation rate. By ignoring ODAA data streams that had noticeable faults for single stream quality analysis, the confirmation of reported events increases to 83%.

4.4 Complex Event Service Composition and Query Processing

In this section we evaluate the performance of the event service composition algorithms using service descriptions annotated with CES ontology and quality ontology used in the semantic annotation. We also test the complex event processing based on primitive sensor observations as well as the summarised observations produced by the data aggregation component described in Section 3.2. In the following we first present the use case scenario designed for the experiments, followed by a decision of the datasets used and finally the results and discussion on the results.

4.4.1 Performances of QoS-aware Complex Event Service Composition

To test the performance of the system we create three user requests on the average vehicle speed and vehicle counts of the routes in Aarhus, as shown in Figure 10. $Q1$ is a short path of 1.5 kilometres, $Q2$ is a medium path of 6.5 kilometres and $Q3$ is a long path of 12.1 kilometres. In the following we test the performance of the sensor stream discovery and integration algorithms with regard to these queries. All experiments are carried out on a MacBook Pro with a 2.53 GHz duo core CPU and 4 GB 1067 MHz memory, prototypes are developed using JDK 1.7.

12. Nokia Here traffic

The discovery and composition algorithm finds 2, 5 and 10 traffic sensors relevant for $Q1, Q2$ and $Q3$ respectively. To test the algorithm on a larger scale, we further increase the size of the sensor repositories by creating N functional equivalent dummy sensors with random NFPs for each sensor in R_0 (i.e., the original sensor repository with 899 sensors), resulting in 9 different sensor repositories, as listed in Table 3.

The results in Figure 11(a) indicate that the composition time of a Brute-Force (BF) enumeration grows exponentially with regard to the complexity of the query (number of sensors involved) and to the number of sensors in the repository. When we apply the Genetic Algorithms[13] (GA) over $Q3$ on R_2, we save approximately 80% of the execution time. However, for $Q1$ and $Q2$, the solution space is too small for the GA evolution. The results in Figure 11(b) shows that the composition time in GA approach for $Q2$ and $Q3$ over R_3 to R_9. The results indicate that the GA composition time grows almost linearly with regard to the size of the repository. To further analyse the performance of the GA algorithm we calculate the aggregated QoS utility for the composition results derived by GA. The detailed calculation of the utility is described in [55], and higher utility means better a composition plan according to user-defined QoS constraints and preferences. Figure 12(a) shows the QoS utility derived for $Q2$ over R_3 to R_9. The QoS utility derived by GA is compared to the maximum and minimum utility derived from brute-force enumeration. To better understand the effectiveness of GA we calculate a *q-score*:

$$\textit{q-score} = \frac{u - u_{min}}{u_{max} - u_{min}} \tag{5}$$

where u is the QoS utility for the GA results, u_{max} and u_{min} are the maximum and minimum QoS utility of the repository, respectively.

Figure 12(b) shows the *q-score* of $Q2$ over R_3 to R_9. The results indicate that GA (with the aforementioned parameters) can produce 66% to 99% optimal QoS utility for $Q2$. The results also suggest that GA gives worse results over larger repositories, because the larger the repository, the less likely the best results will be generated. However, this does not mean GA is more suitable for small repositories. To explore whether using GA is cost-effective, we observe the execution time of GA and BF for $Q2$ over R_3 to R_9 and calculate a cost-effective score *c-score*:

$$\textit{c-score} = \frac{\textit{q-score} \times t_{bf}}{t_{ga}} \tag{6}$$

where t_{ga} and t_{bf} are the execution time of GA and BF algorithms, respectively. In this equation we take the q-score as the gain and the time ratio of t_{ga} and t_{bf} as the cost and hence calculate the cost-effectiveness.

Results in 12(b) show that the *c-score* increases when GA is applied to larger repositories, despite the fact that the *q-score* decreases. This is because, the time save by GA (i.e., the cost reduced) is greatly increased even though it is slightly more difficult to get good composition results in larger repositories using GA. For R_3 and R_4 the *c-score* is below 1, suggesting GA less cost-effective than BF over

13. GA parameters: crossover rate = 95%, mutation rate = 5%, initial population size = 200, selection policy: Roulette Wheel + Elitism

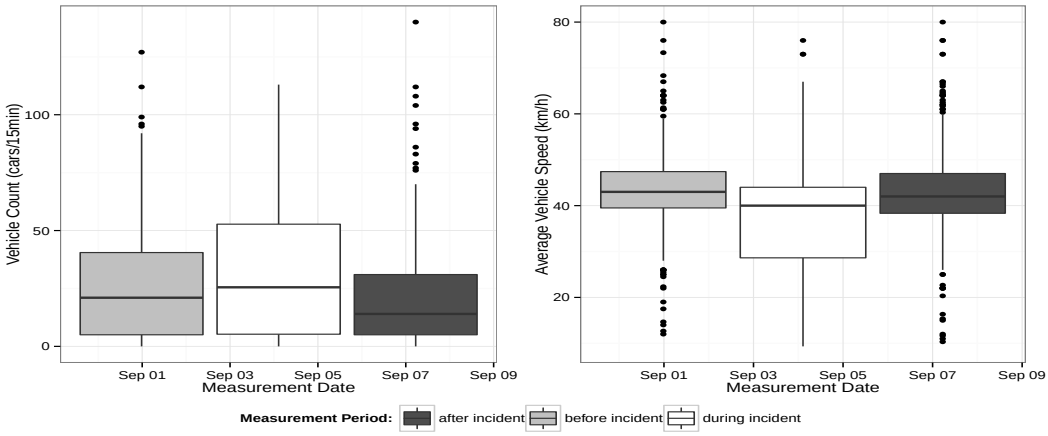
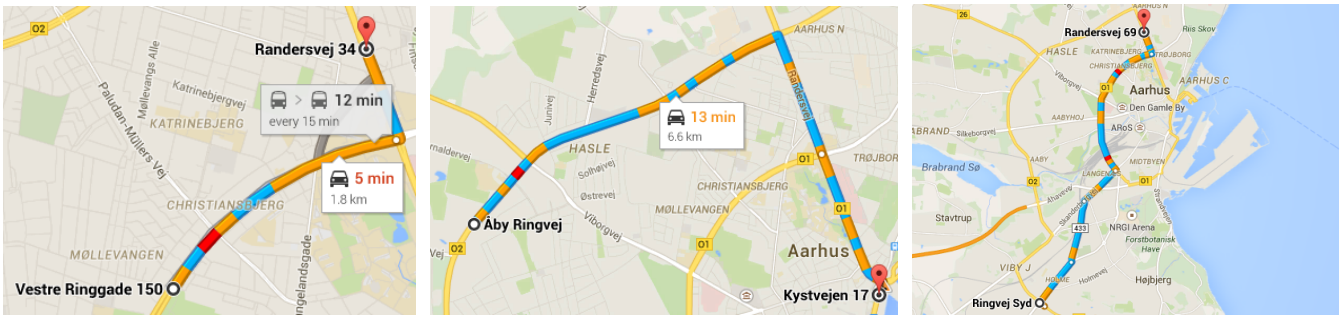


Fig. 9: Comparing Average Speed Before, During, and after a traffic incident

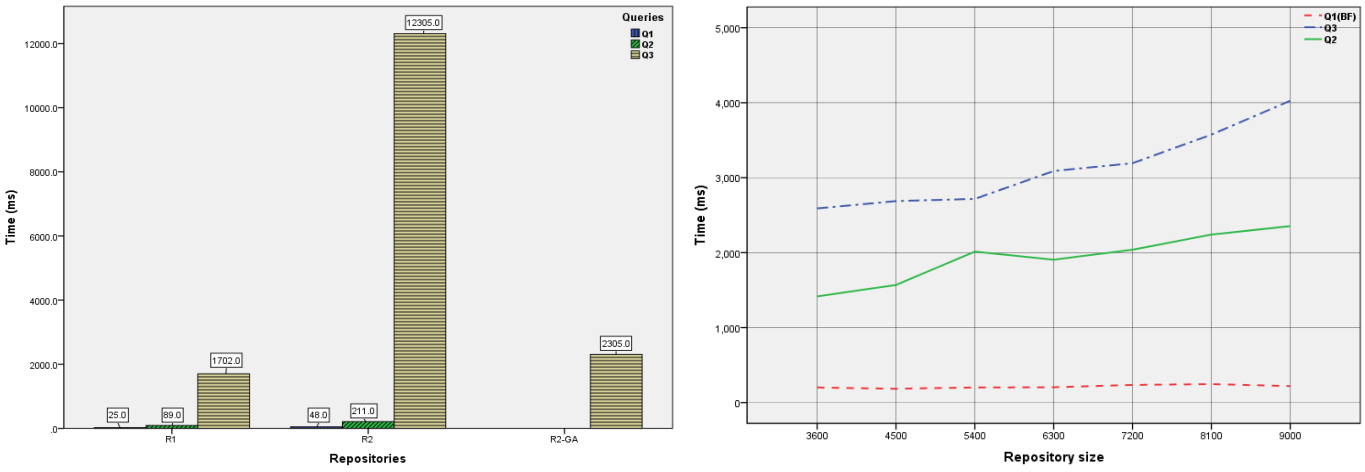


(a) Q1 (b) Q2 (c) Q3

Fig. 10: Different route queries in Aarhus, captured from Google Maps

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
N	1	2	3	4	5	6	7	8	9
total size	1798	2697	3596	4495	5394	6293	7192	8091	8990

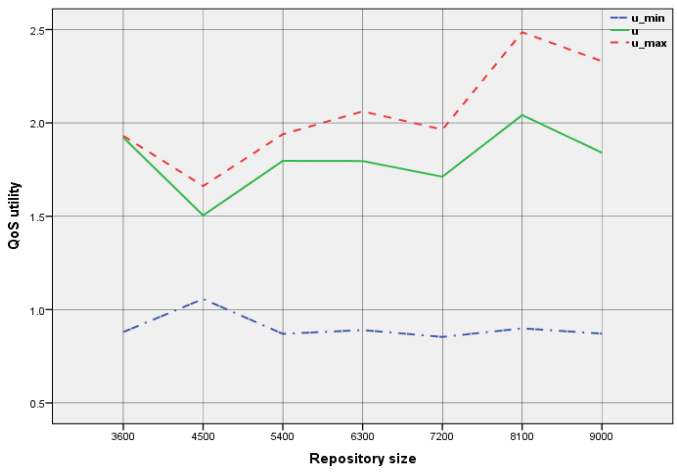
TABLE 3: Simulated sensor repositories



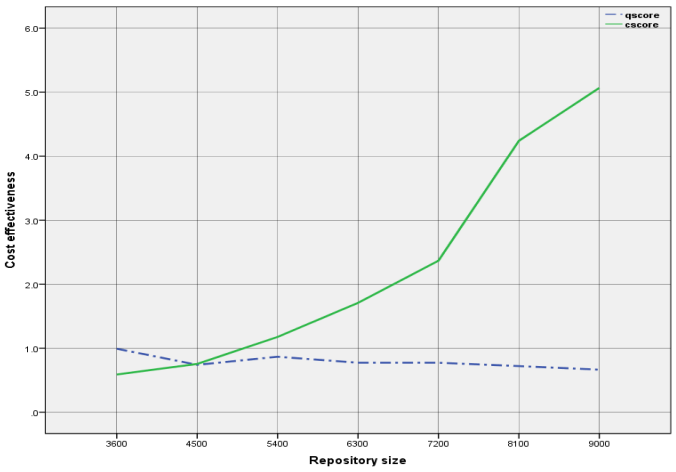
(a) Brute-force vs. GA (b) GA execution time over different repositories

Fig. 11: Performance of Genetic Algorithms: Execution Time

these two repositories. However, as the repository size gets bigger, the *c-score* increased to above 5, indicating that on R_9 GA is 5 times as cost-effective as BF.

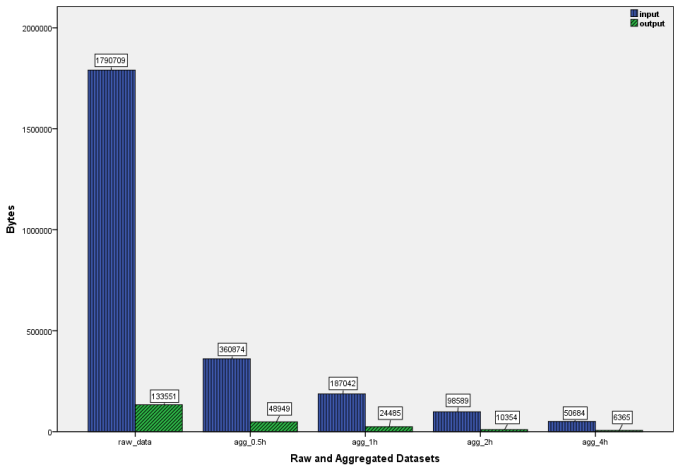


(a) QoS utility of GA results

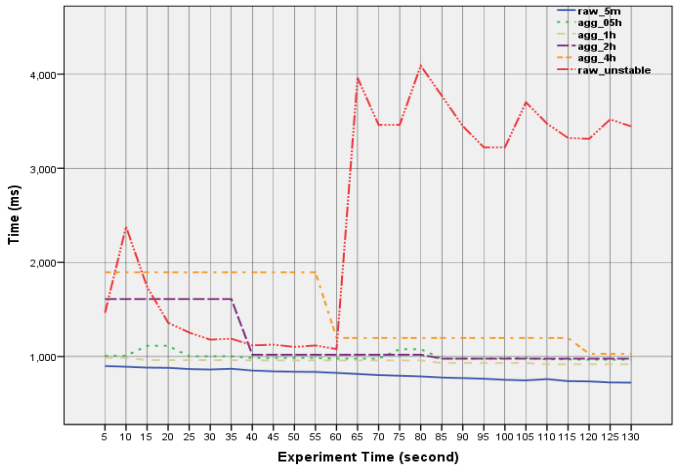


(b) GA cost effectiveness

Fig. 12: Performance of Genetic Algorithms: Quality of Results



(a) Input and Output sizes



(b) CEP processing delays

Fig. 13: CEP performance over primitive and aggregated data

4.4.2 Performances of Complex Event Processing over Primitive and Aggregated Observations

Once we have the event service composition plans, we transform them into stream query and deploy the queries on stream engines to detect complex events using algorithms described in [56]. In this section we test the performance of complex event processing when using raw and aggregated data streams. The stream engine used is C-SPARQL [57]. Listing 9 shows a snippet of the C-SPARQL query transformed from $Q3$.

```
REGISTER QUERY test AS
PREFIX ...
SELECT DISTINCT ?obId1 ?avgSpd1 ?obId2 ?vehicleCnt1 ...
FROM STREAM <...#sensor_540> [RANGE 2s step 1s] ...
WHERE {
{?obId1 rdf:type ?ob. ?obId1 ssn:observedBy sensorRepo:sensor_540.
?obId1 ssn:observedProperty <.../citytraffic#AvgSpeed>.
?obId1 sao:hasValue ?avgSpd1. ...}
...}
```

Listing 9: C-SPARQL query for Q3

We collected the traffic data (i.e., raw and aggregated) for the 10 sensors involved in $Q3$ over one day and replay them. The replayed streams are fed to the C-SPARQL engine. The original traffic data streams from ODAA create 1 report containing 1 observation for average speed and 1 observation for vehicle count every 5 minutes, an aggregated traffic data is abstracted out of 6, 12, 24 and 48 ODAA traffic reports, i.e., updated every 30, 60, 120 and 240 minutes (i.e., same segmentation sizes as in Section 3.2). In our experiments, we test the amount of messages consumed and produced by the C-SPARQL engine as well as the processing delay. To see how the C-SPARQL engine performs with higher throughput, we accelerate the update frequency 150 times. Under this accelerated rate the raw and aggregated data reports every 2 to 96 seconds, respectively. As a result, the input rate of the C-SPARQL engine varies from 40 to 0.83

triples per second in our experiments[14]. Figure 13(a) shows the amount of inputs and outputs and Figure 13(b) shows the average processing delay over time.

From the results in Figure 13(a) we can see that using aggregated data over 30 minutes reduces 80% of the data consumed, and 63% of the results produced compared to using raw data, and the I/O traffic decreases linearly to the segmentation size. The results in Figure 13(b) show the processing delay decreases over the experiment time (C-SPARQL engine warming-up), and for raw data and aggregated datasets, the delay decreased to less than 1000 ms within 125 seconds of experiment time. The results also show that using aggregated data with larger segmentation size has slower decrease rate. If we extend the experiment until all processing delays are stabilised, the difference of delays are small, i.e., they all converged to about 600 ms (+/- 50ms). However when we further accelerate the raw data stream frequency to 1 report per second (i.e., 80 triples per second, relatively high for the large query with many joins in Listing 9), the C-SPARQL engine becomes unstable and the processing delay increases to up to 4000 ms after a while and then the engine stopped giving query results.

5 Discussion and Conclusion

In this paper we examined large-scale stream processing including data aggregation, quality analysis, as well as semantic data discovery, integration, and complex event processing issues in the domain of smart city applications. We showed how techniques such as SAX can be used to reduce the size of data and presented different techniques for accessing and processing semantically annotated data streams.

5.1 Data aggregation/reduction without loosing essential information

Many time-series analysis approaches have been introduced within the last decades. The key issues are to keep good quality synopsis of data and to provide lower bounding support for the reduced data. SAX algorithm provides a considerable data reduction with accurate correlation between the original time series and the symbolic representation, while supporting lower bounding principle. Using SAX algorithm in our experiments, we obtained significantly reduced data for each sensor observation, which possess an average data size of 60.75MB, reducing the content of data in range of 94.24% and 99.18% depending on the segmentation size. On the other hand, it was also interesting to see how the change in segmentation size is significantly affected by the quality of time series data in an unfavourable way, which caused an overall error rate in range of 11.2km/h and 41.3km/h for average speed, and 4.0 and 10.9 number of vehicles. Overall, the results lead to a conclusion that $S30m$, which provided the minimum error and no large difference on the complex event processing is an ideal segmentation size for our framework.

14. Every observation is annotated with 4 triples for streaming, less than those annotated in listing 1 because some triples there are only useful for historical analysis.

5.2 Quality data leads to intelligent decisions

The QoI provided by the data streams is determined in an objective and application independent kind of way. A normalised QoI value enables comparability and eases the stream selection. Distinct measurable QoI metrics like the frequency are evaluated for every observation and compared against the initial annotated metric. This allows a continuous observation of the performance of the sensors over their lifetime and ensures reliable execution of the application using them.

The usage of an event ontology, which is describing the effects of incidents, will enable the modeling of the mutual impact of various data streams. This information can further be used to evaluate the correctness of sensor observations and events.

5.3 On-demand data discovery, federation and complex event processing

In the smart city environment, smart city applications facilitate its users by allowing to submit real time queries which can only be answered by complex event processing over various data streams. Considering the huge number of data streams availability within smart city infrastructure, automated discovery and composition of relevant data streams for complex event processing is a taunting task. CityPulse framework enables CEP engines to perform complex event processing while catering for non-functional properties (e.g. completeness, timeliness and accuracy etc.) of the data streams contributing within the composition plan.

Using the brute force enumeration over available data streams for discovery and composition results in the exponential growth of composition time, which is not scalable for real-time smart city applications. However, application of genetic algorithm over a large repository of data streams resulted in better performance for cost effectiveness without much compromise over the optimality. Results shown in this paper are conducted under a specific setting for the tuning GA parameters. However, the performance of GA can vary on different settings over a number of parameters, such as initial population size, selection policy and mutation rate. There is always a trade-off between the processing time and quality of the results produced by GA and finding the best settings to fine tune GA parameters is heavily dependent over application domain. In the context of the smart city applications, real-time applications can decrease the initial population size to reduce the processing time with a little compromise over the quality of the results. Moreover, effectiveness of the GA is better realised over a large dataset of data stream repository (as depicted in Figure 12(b)), which meets the requirement of large-scale stream federation and processing for smart city applications.

In summary, the data abstraction/aggregation was found to have two effects on real-time stream processing: 1) data aggregation can significantly decrease the I/O traffic of the stream engine and 2) data aggregation can reduce the stream rate so that the stream engine can process more complicated queries without becoming unstable, and although slower streaming rate results in slower warming-up period, when the warming-up phase ends the stabilised delay is

similar for the same query, regardlessly of the streaming rate.

5.4 Future work

In future work we'll further investigate data aggregation process to obtain an adaptive segmentation size, where the system will proceed aggregation solely when there is an event. In addition, due to the fact that it is possible to have time series data, which is not obeying to Gaussian distribution and this can worsen the efficiency of the reduction, we will investigate ways to compensate it with different data distributions. We will also work on event discovery and analysis in large-scale distributed smart city data streams. Reusing the developed techniques and models in different use-case scenarios and applications will be also explored in our future work.

ACKNOWLEDGMENT

This work was supported by the European Commission's Seventh Framework Programme for the CityPulse Project under Grant 609035 and partially by the European Commission's Horizon 2020 IoTCrawler under Grant 779852.

REFERENCES

[1] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges," *Computer*, vol. 44, no. 6, pp. 32–39, June 2011.

[2] Şefki Kolozali, D. Puschmann, M. Bermudez-Edo, and P. Barnaghi, "On the effect of adaptive and non-adaptive analysis of time-series sensory data," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1084–1098, 2016.

[3] M. Hashmi, M. Wright, K. Sultana, B. Barratt, E. M. Lia Chatzidiakou, Ş. Kolozali, R. L. Jones, S. Beevers, L. Smeeth, F. J. Kelly, and J. K. Quint, "Preliminary results from the cope study using primary-care electronic health records and environmental modelling to examine copd exacerbations," *British Journal of General Practice*, vol. 68, 2018.

[4] L. Yang, W. Li, M. Ghandehari, and G. Fortino, "People-centric cognitive internet of things for the quantitative analysis of environmental exposure," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2353–2366, 2018.

[5] P. Barnaghi, A. Sheth, and C. Henson, "From data to actionable knowledge: Big data challenges in the web of things," *Intelligent Systems, IEEE*, vol. 28, no. 6, pp. 6–11, 2013.

[6] P. Bellini, M. Benigni, R. Billero, P. Nesi, and N. Rauch, "Km4city ontology building vs data harvesting and cleaning for smart-city services," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 827–839, 2014.

[7] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smartsantander: Iot experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[8] D. Pfisterer, K. Römer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth *et al.*, "Spitfire: toward a semantic web of things," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 40–48, 2011.

[9] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, and I. P. Žarko, "Openiot: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*. Springer, 2015, pp. 13–25.

[10] S. De, F. Carrez, E. Reetz, R. Tönjes, and W. Wang, *Test-Enabled Architecture for IoT Service Creation and Provisioning*. Springer, 2013.

[11] E. Kalampokis, A. Nikolov, P. Haase, R. Cyganiak, A. Stasiewicz, A. Karamanou, M. Zotou, D. Zeginis, E. Tambouris, and K. Tarabanis, "Exploiting linked data cubes with opencube toolkit," in *Proceedings of the 13th International Semantic Web Conference (ISWC)*, 2014.

[12] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," *Communications Magazine, IEEE*, vol. 51, no. 6, pp. 102–111, June 2013.

[13] F. Lecue, S. Tallevi-Diotallevi, J. Hayes, R. Tucker, V. Bicer, M. L. Sbodio, and P. Tommasi, "Star-city: semantic traffic analytics and reasoning for city," in *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM, 2014, pp. 179–188.

[14] A. Bar-Noy, G. Cirincione, R. Govindan, S. Krishnamurthy, T. LaPorta, P. Mohapatra, M. Neely, and A. Yener, "Quality-of-information aware networking for tactical military networks," in *Proceedings of the Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2–7.

[15] C. Bisdikian, L. M. Kaplan, and M. B. Srivastava, "On the quality and value of information in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 4, p. 48, 2013.

[16] J. Benesty, J. Chen, and Y. A. Huang, "On the importance of the pearson correlation coefficient in noise reduction," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 4, pp. 757–765, 2008.

[17] K. Aberer, M. Hauswirth, and A. Salehi, "The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks," Tech. Rep., 2006.

[18] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Kranenburg, S. Lange, and S. Meissner, *Enabling things to talk: designing IoT solutions with the IoT architectural reference model*. Springer, 2013.

[19] F. den Hartog, L. Daniele, and J. Roes, "Study on semantic assets for smart appliances interoperability: D-s1: First interim report," 2014.

[20] F. Cicirelli, G. Fortino, A. Guerrieri, G. Spezzano, and A. Vinci, "Metamodeling of smart environments: from design to implementation," *Advanced Engineering Informatics*, vol. 33, pp. 274–284, 2017.

[21] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio, "Enabling iot interoperability through opportunistic smartphone-based mobile gateways," *Journal of Network and Computer Applications*, vol. 81, pp. 74–84, 2017.

[22] A. Altomare, E. Cesario, C. Comito, F. Marozzo, and D. Talia, "Trajectory pattern mining for urban computing in the cloud," *IEEE Transanctions on Parallel and Distributed Systems*, vol. 28, no. 2, 2017.

[23] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, "Edge computing and social internet of things for large-scale smart environments development," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2557–2571, 2018.

[24] C. Faloutsos, R. Agrawal, and A. Swami, "Efficient similarity search in sequence databases," *Proceedings of the 4th Conference on Foundations of Data Orgamisation and Algorithms*, pp. 69–84, 1993.

[25] A. Haar, ""zur theorie der orthogonalen funktio- nensysteme". german," in *Mathematische Annalen*, vol. 69, no. 331–371, 1910, pp. 38–53.

[26] Z. R. Struzik and A. Siebes, "The haar wavelet transform in the time series similarity paradigm," *Principles of Data Mining and Knowledge Discovery*, pp. 12–22, 1999.

[27] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Transactions on Database Systems*, vol. 27, no. 2, pp. 188–228, 2002.

[28] K. Chakrabarti, E. Keogh, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Journal of Knowledge and Information Systems*, vol. 3, pp. 263–286, 2000.

[29] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[30] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti, *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. InTech, 2012, ch. 3, pp. 71–96.

[31] H. Andre-Jonsson and D. Z. Badal, "Using signature files for querying time-series data," in *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, 1997, pp. 211–220.

[32] Y. wu Huang and P. S. Yu, "Adaptive query processing for time-series data," in *Proceedings of the 5th International Conference on*

Knowledge Discovery and Data Mining. ACM Press, 1999, pp. 282–286.

[33] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. ACM Press, 2003, pp. 2–11.

[34] 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.

[35] B. Stvilia, L. Gasser, M. B. Twidale, and L. C. Smith, "A framework for information quality assessment," in *Journal of the American Society for Information Science and Technology*, vol. 58, no. 12. Wiley Online Library, 2007, pp. 1720–1733.

[36] C. Bisdikian, L. M. Kaplan, M. B. Srivastava, D. J. Thornley, D. Verma, and R. I. Young, "Building principles for a quality of information specification for sensor information," in *Proceedings of the Information Fusion, 2009. FUSION'09. 12th International Conference on*. IEEE, 2009, pp. 1370–1377.

[37] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*. ACM, 1999, pp. 53–61.

[38] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 2006, pp. 407–418.

[39] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, p. 15, 2012.

[40] T. Moser, H. Roth, S. Rozsnyai, R. Mordinyi, and S. Biffl, "Semantic event correlation using ontologies," in *On the Move to Meaningful Internet Systems: OTM 2009*. Springer, 2009, pp. 1087–1094.

[41] K. Taylor and L. Leidinger, "Ontology-driven complex event processing in heterogeneous sensor networks," in *The Semanic Web: Research and Applications*. Springer, 2011, pp. 285–299.

[42] K. Teymourian and A. Paschke, "Semantic rule-based complex event processing," in *Rule Interchange and Applications*. Springer, 2009, pp. 82–92.

[43] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th international conference on World wide web*, ser. WWW '09. ACM, 2009, pp. 881–890.

[44] M. Jaeger, G. Rojec-Goldmann, and G. Muhl, "Qos aggregation for web service composition using workflow patterns," in *Proceedings of the Eighth IEEE InternationalEnterprise Distributed Object Computing Conference. EDOC 04.*, 2004, pp. 149–159.

[45] Q. Wu, Q. Zhu, and X. Jian, "Qos-aware multi-granularity service composition based on generalized component services," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds. Springer Berlin Heidelberg, 2013, vol. 8274, pp. 446–455.

[46] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, 2004.

[47] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in *Proceedings of the IEEE International Conference on Web Services*, ser. ICWS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 72–82.

[48] L.-J. Zhang and B. Li, "Requirements driven dynamic services composition for web services and grid solutions," *Journal of Grid Computing*, vol. 2, no. 2, pp. 121–140, 2004.

[49] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "A lightweight approach for qos-aware service composition," in *Proceedings of 2nd international conference on service oriented computing (ICSOC 04)*, 2004.

[50] C. Zhang, S. Su, and J. Chen, "A novel genetic algorithm for qos-aware web services selection," in *Proceedings of the Second international conference on Data Engineering Issues in E-Commerce and Services*, ser. DEECS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 224–235.

[51] C. Gao, M. Cai, and H. Chen, "Qos-aware service composition based on tree-coded genetic algorithm," in *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 01*, ser. COMPSAC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 361–367.

[52] F. Karatas and D. Kesdogan, "An approach for compliance-aware service selection with genetic algorithms," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds. Springer Berlin Heidelberg, 2013, vol. 8274, pp. 465–473.

[53] F. Gao, E. Curry, and S. Bhiri, "Complex event service provision and composition based on event pattern matchmaking," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*. Mumbai, India: ACM, 2014.

[54] M. A. Hossain, P. K. Atrey, and A. E. Saddik, "Modeling and assessing quality of information in multisensor multimedia monitoring systems," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 7, no. 1, p. 3, 2011.

[55] F. Gao, E. Curry, M. A. Intizar, S. Bhiri, and A. Mileo, "Qos-aware complex event service composition and optimization using genetic algorithms," in *Proceedings of the 12th International Conference on Service Oriented Computing*. Paris, France: Springer, 2014.

[56] F. Gao, M. A. Intizar, and A. Mileo, "Semantic discovery and integration of urban data streams," in *Proceedings of the 5th Workshop on Semantics for Smarter Cities*, ser. 5, 10 2014.

[57] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus, "C-sparql: Sparql for continuous querying," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 1061–1062.