2012 European Intelligence and Security Informatics Conference

Detecting Anomalous Maritime Container Itineraries for Anti-fraud and Supply Chain Security

Elena Camossi Tatyana Dimitrova Aris Tsois
European Commission, Joint Research Centre (JRC)
via Enrico Fermi 2749, Ispra, Varese, Italy – I-21027
Email: {elena.camossi, tatyana.dimitrova, aris.tsois}@jrc.ec.europa.eu

***Abstract*—An important contribution to anti-fraud and supply-chain security comes from the development of Risk Analysis tools targeted to the discovery of suspicious containerized transportations. In this work we present the Anomalous Container Itinerary Detection (ACID) framework that analyses Container Status Messages to discover irregular container shipments. The system has been developed at JRC as part of an in-house global route-based risk analysis facility for containers monitoring. It adopts a flexible and modular design and its preliminary prototype applies an SVM one-class classifier to detect anomalies. The experimental evaluation demonstrates that the analysis module may be set to detect efficiently the expected number of suspicious itineraries, which can be further investigated by Customs authorities thanks to the web-based visualization tool provided with the system.**

I. Introduction

It has been estimated that about 90% of the world's trade is shipped in cargo containers, and 18 millions of containers travel by sea every year. In the effort to enforce supply chain security, recent legislation obliges carrier companies to provide authorities with detailed trip information [1], [2] and attempts to increase inspections [3]; however, currently less than 2% of containers are physically checked and, from many parts [4], [5], it is argued that the full implementation of 100% scanning would cause huge economic losses for the world trade market. In such a complex scenario, the usage of Risk Analysis tools becomes essential to improve the capacity of Customs authorities to target anomalous shipments, hence improving the effectiveness of physical inspections [4].

Key objectives of the European Commission Joint Research Centre (JRC) are the support of EU policies for internal security, customs and maritime affairs; the promotion of Community Customs Cooperation and to assist the work of the European Anti-Fraud Office. Amid the research activities undertaken in the area of maritime surveillance, an in-house global route-based risk analysis facility for maritime containers monitoring, namely *ConTraffic*, has been developed. This system analyses information on cargo vessels and containers and develops risk indicators to fight commercial frauds and to improve supply chain security. In particular, it embeds a subsystem for data collection of Container Status Messages (CSMs). CSMs describe the activities undertaken by carrier companies on containers, including their position, the operations that are carried out on them (referred to as *container event*), their loading status and the vessels used for their transportation.

In this paper, we propose a framework for the analysis of CSMs data named Anomalous Container Itinerary Detection (ACID). The system encompasses a pre-processes module to aggregate CSMs into *container itineraries*, which describe trips to accomplish the shipments; afterwards, it analyses them to detect *anomalies*. Indeed, CSMs alone are not expressive enough to evaluate the behaviour of containers. By contrast, container itineraries enable to check the container activity against typical values, referred to as *risk indicators*.

The ACID system focuses in particular on spatio-temporal risk indicators such as the origin, the destination, and the duration of a shipment; the duration of the trip accomplished by sea; the ports a container passed through to complete the itinerary and the number of times it has been transshipped during the trip (i.e., unloaded from a vessel to be loaded on a different one). The analysis module of ACID processes these risk indicators to discover irregular container shipments. For this purpose, *anomaly detection* [6], an unsupervised mining task that detects outliers deviating from the typical distribution of a dataset[1], is applied.

In this work, a classifier has been used to detect outliers, then the anomaly-detection task is referred to as *one-class classification* [7], because the classifier model is build directly on the test data instead of using a training dataset prepared on purpose, making this approach *unsupervised*. Other approaches to anomaly detection exist, including: kernel Principal Component Analysis (PCA) [8], multivariate analysis [9]; Gaussian Mixture Models (GMM) [10]; bayesian networks [11]; clustering [12]; fuzzy set theory [13]. Recently, the scientific community has been investigating anomaly detection for spatio-temporal data [14], trajectories [15] and moving object data [16], [17], [18]. Some of these approaches have been used to discover outliers for sea traffic data [19], [20], [21], [22], relying also on the structural properties of graphs [23].

Unsupervised approaches usually provide a less precise scoring than supervised ones, but they are useful whenever counter-examples are hard to collect, as it happens in risk analysis, where counter-examples are rarely available and usually unprofitable, because new risk patterns continue to emerge. Therefore, in this domain unsupervised anomaly detection may

[1] Anomaly detection is also known as outlier detection, novelty detection, and density estimation.

978-0-7695-4782-4/12 $26.00 © 2012 IEEE
DOI 10.1109/EISIC.2012.39

turn out to be a valuable resource, because it does not rely on legacy case studies but helps discover novel anomalous patterns.

In this work, we demonstrate the viability of unsupervised anomaly detection to analyse container itineraries for the discovery of suspicious, i.e., potentially illegal, transportations. The architecture of the ACID system includes modules for data preparation, analysis and visualization, and implements the workflow for Knowledge Discovery from Data (KDD) [24] for container risk analysis. Moreover, the system design is extendible to alternative visualization tools and to other analysis algorithms.

The analysis module we present herein adopts Support Vector Machines (SVM) [25] to implement one-class classification. SVM are a collection of robust machine learning methods for linear classification and regression analysis. With respect to other classification algorithms, SVM have several advantages, including the possibility to analyse high-dimensional data and the capability to treat non linearly separable data through the definition of appropriate kernel functions [26] that transform the data in a high-dimensional feature space in which the data become linearly separable. Thanks to their flexibility SVM are used in a variety of application domains, including sequence analysis and knowledge discovery from moving object trajectories.

The experimental evaluation of this preliminary version of the system has been encouraging and demonstrated the feasibility of the ACID approach. Indeed, the system prototype has been tested on a dataset of real world CSMs taken from the ConTraffic facility that includes more than three hundred thousand itineraries for fifty thousand containers collected in the last three years. The tests demonstrated that ACID may discover efficiently the expected number of anomalous itineraries; moreover, the system parameters can be tuned to augment the classification accuracy to the desired level.

The paper is organized as follows. The architecture and the process flow of the ACID system is introduced in Section II. In Section III we describe the input dataset, describing how CSMs are aggregated into itineraries and prepared to be analysed. In Section IV we describe the analysis API, while in Section V we show how the analysis results are visualized to be evaluated by Customs authorities. In the last two sections, we describe the experiments and outline future research directions and developments.

II. THE ACID DESIGN, ARCHITECTURE AND DATA FLOW

The architecture of the ACID system is depicted in Fig. 1. It encompasses four modules for data representation, preparation, analysis and visualization. Each module is independent from the others to allow the development of alternative and complementary implementations, while continuing to fulfill the ACID workflow.

The *Data Module* (DM) is made by the database (DB) sources used by the system. We leverage on DBMS properties to achieve data persistence and integrity whilst preserving access performance.

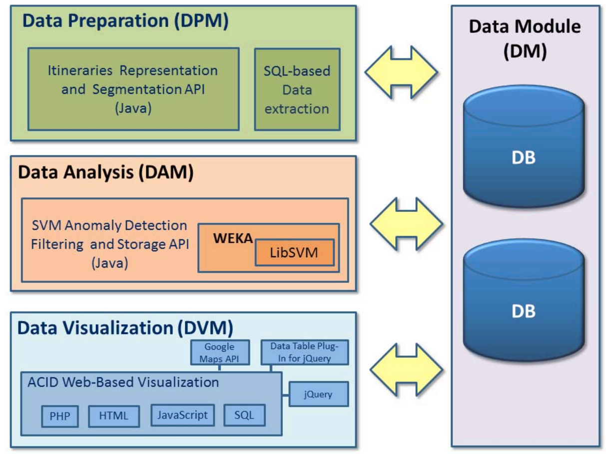


Fig. 1. ACID architecture

The ACID input dataset is made by CSMs sequences. Each of them describes the shipment history of a single container, which is identified through its international identification code as assigned by the *Bureau International des Containers et du Transport Intermodal* [27] according to ISO 6346:1995 standard [28].

The *Data Preparation Module* (DPM) includes the procedures for the segmentation of CSMs sequences into *container itineraries*, each describing a shipment from its origin to the location of the final consignment[2]. The segmentation algorithm relies on the semantics of container events, i.e., the deed occurring to a container, such as loaded, discharged, transshipped, etc. Thanks to container events we identify the different phases of a shipment and distinguish one container trip from another.

The itineraries are stored in the DM and analysed by the *Data Analysis Module* (DAM), which applies SVM one-class classification to find a preliminary set of anomalies. Then, the DAM applies a filtering step to reduce them to a tractable set of suspicious itineraries, which can be investigated by Customs authorities. In our experience spatio-temporal risk indicators such as the time required for travelling and accomplishing a shipment, information on the transshipments of a containers, etc., can drive effectively the discovery of interesting suspicious cases. In order to focus a general purpose SVM to these risk indicators, we put a great effort in preparing the input dataset for the DAM, aiming at including most of the relevant information on itineraries, while at the same time designing a data structure that is processable by the classifier.

[2]We use the term *itinerary* instead of *trajectory* that is more common in the literature, because container itineraries include fewer spatial positions than typical GPS trajectories. Indeed, only the main samples of the container trajectory are included in the container itinerary, i.e., the positions of the container at the ports or locations it has been handled to accomplish the shipping procedures. Moreover, in addition to container positions, itineraries include also information on the container events, which are not part of a pure spatial trajectory.

This trade-off has shortcomings, because some information has to be excluded (n particular, information that can not be represented through numeric data); nevertheless, as discussed in the following, this preliminary version of the system enabled us to discover interesting anomalous cases, validating the approach that can be further extended in the future with other analysis algorithms.

The *Data Visualization Module* (DVM) facilitates the evaluation of anomalous container itineraries though a web-based geographical application based on Google Maps API [29]. In alternative, the anomalies can be visualized querying the corresponding geo-spatial layer which is included in the result dataset (for example defining a Web Map Service (WMS) [30] and visualizing the corresponding spatial layer on top of any worldwide map (see for example [31]).

The modules included in the architecture of ACID concur to accomplish the analysis workflow as depicted in Fig. 2. The details of each module are described in the following sections.

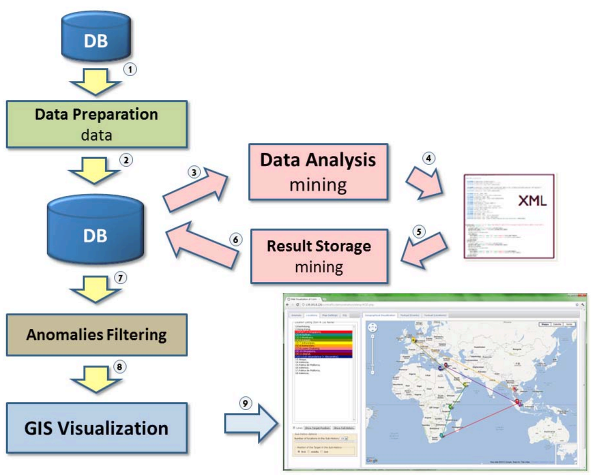


Fig. 2. Anomaly detection workflow of ACID

III. TRANSFORMING CSMs INTO CONTAINER ITINERARIES

The input data of the system are extracted from the CSM data collected by JRC as part of the container monitoring activity. Each CSM has the following format:

1) an ISO 6346 container identifier made by: a code for the owner of the container; a code for the category of container; a unique serial number; and a check digit [3].
2) the text description of the event occurring to the container, such as loaded, discharged, etc.;
3) the date when the event occurred;
4) the place, usually a port, where the event took place;
5) the loading status of the container (empty or full);
6) depending on the event type, a vessel identifier.

[3]Given for instance container identifier ABCD1234567, ABC identifies the carrier company, D is the container category; 123456 is a serial number and 7 is a check digit.

Extract-Transform-Load (ETL) procedures are applied to CSMs in order to prune errors and to harmonize them with respect to standard taxonomies for for events, loading status, locations and vessels. Afterwards, SQL scripts in the DPM extract the cleaned container CSMs sequences that will be analysed. Each sequence represents the time ordered history of shipments of a single container (see Fig. 3). Then, a Java [32]

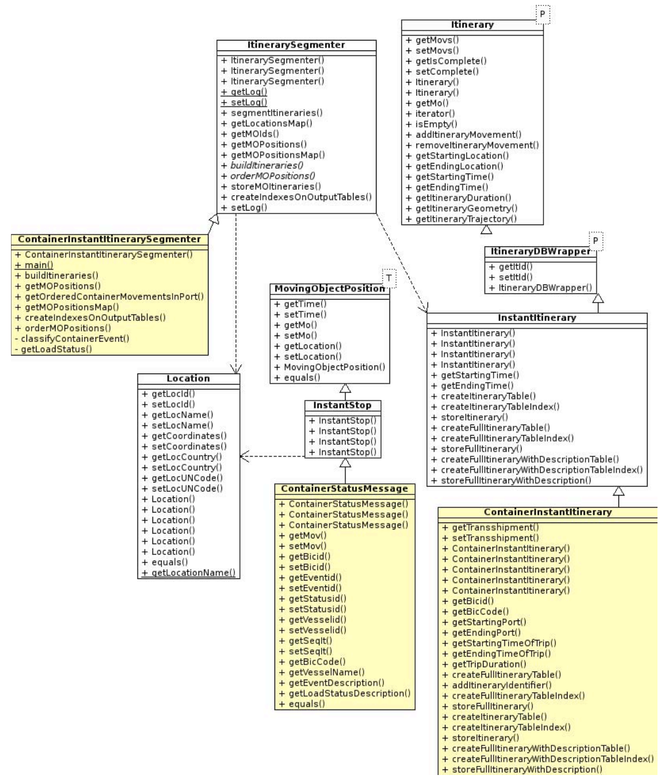


Fig. 4. Excerpt of the ACID class diagram for data preparation

API included in the DPM segments CSM sequences into container itineraries (see the class diagram of Fig. 4). Given for instance the sequence in Fig. 3, it includes two itineraries for container ABCD1234567: the first starting at Shangai in Cina the 27 May and ending at Antwerpen in Belgium the 16 July; and the second, which is partial, starting at Antwerpen the 20 August.

The method *SegmentItineraries()* of classes *ContainerInstantItinerarySegmenter* and *ItinerarySegmenter* in Fig. 4 accomplishes the segmentation of CSM sequences. Its algorithm is summarized in Fig. 5. It processes iteratively the dataset of CSM sequences, one container sequence per time.

Steps (7-8) are the core of the algorithm. The sequence of CSMs of a given container is first ordered according to the time of CSM events taking into account the contextual information of the events and their semantics. Within the project an effort towards standardization has resulted in nineteen container events, that at a coarser level can be group into five classes representing the main phases of a shipment: 1)

CSM identifier	Container identifier	Time	Event	Location	Loading status	Vessel
12345	ABCD1234567	27 May 2010	Received at Origin	Shangai (CN)	Empty	–
12346	ABCD1234567	27 May 2010	Gate In	Shangai (CN)	Full	–
12350	ABCD1234567	30 May 2010	Loaded/Ramped	Shangai (CN)	Full	Aurora
12365	ABCD1234567	15 Jun 2010	Discharged/Deramped	Port Kelang (MY)	Full	–
12366	ABCD1234567	17 Jun 2010	Loaded/Ramped	Port Kelang (MY)	Full	Dawn
12381	ABCD1234567	03 Jul 2010	Discharged/Deramped	Antwerpen (BE)	Full	–
12399	ABCD1234567	09 Jul 2010	Gate Out	Antwerpen (BE)	Full	–
12455	ABCD1234567	16 Jul 2010	Final Destination	Antwerpen (BE)	Full	–
12484	ABCD1234567	20 Aug 2010	Received at Origin	Antwerpen (BE)	Empty	–
12545	ABCD1234567	23 Aug 2010	Gate In	Antwerpen (BE)	Full	–
12555	ABCD1234567	24 Aug 2010	Loaded/Ramped	Antwerpen (BE)	Full	Sun

Fig. 3. Example of container history for container ABCU1234567

CSM Itinerary Segmentation

(1) create the output tables;
(2) reindex the CSM dataset;
(3) locMap $\leftarrow$ getLocationsMap();
(4) contList $\leftarrow$ getMOIds();
(5) **For each** c $\in$ contList
(6) $\quad$ $\mathrm{CSM}_c \leftarrow$ getMOPositions(c);
(7) $\quad$ $\mathrm{CSM}'_c \leftarrow$ orderMOPositions(CSM_c);
(8) $\quad$ $\mathrm{it}_c \leftarrow$ buildItineraries(CSM'_c);
(9) $\quad$ storeMOItinerararies(it_c);
(10) createIndexesOnOutputTables();

Fig. 5. CSM Segmentation algorithm

Itinerary start; 2) *Container export*; 3) an optional sequence of *Container transshipment*s from one vessel to another in some intermediate port; 4) *Container import*; 5) *End of Itinerary*. We refer to such high level sequential pattern to order CSMs sequences, in particular for events that describe operations handled in the same port at the same date and to split sequence of ordered CSMs into container itineraries (step 8 in Fig. 5).

The segmentation algorithm takes into account also events that do not describe a container movement but are deeds occurring to prepare the container for the shipping at the source port or to complete it at the port of destination (e.g., released to shipper for cargo stuffing, empty returned). These events, complemented with the loading status of the container, help localize a container in a specific port at the begin and at the end of a shipment, therefore define more precisely the temporal period a container was in a port.

The database structure for container itineraries includes summarized values useful for the analysis, such as aggregated value on the time to perform a shipment and the number of transshipments, which, together with the port of origin and destination, are the risk indicators the classifier will analyse to compare each itinerary against typical values for the same route.

In Fig. 6 some of the columns of the itinerary table are reported; in particular: the origin (**Origin**) and destination (**Destination**) of the itinerary; the starting (**Start**) and ending (**End**) dates; and the duration of the shipment (**ItineraryT**), in days; the port of origin (**PoOr**) and of destination (**PoDest**) of the maritime part of the itinerary, which can differ from origin and destination; the corresponding starting (**MStart**) and ending (**MEnd**) dates (referring to the loading and discharging events); the duration of the maritime trip (**MTripT**), in days; and the number of transshipments (**Transs**). In addition, the itinerary record includes also the geographic representation of the itinerary trajectory as line, specified as ORACLE SDO_GEOMETRY data type [33], which is currently discarded by the analysis module but can be useful to visualise the analysis results.

Other information, such as the list of events occurring within an itinerary and the list of ports the container passed through, have not been included in this version of the prototype because it was not possible to define an aggregated representation for them whose data structure was processable by the general purpose SVM. For this type of information, specialized tools such as kernel functions for strings or structured data are required, but a specific implementation is not currently available in the data mining toolkit we have adopted.

As shown in the class diagram in Fig. 4, the data preprocessing API includes a set of classes that have been extended to represent containers and itineraries. For example, *ContainerInstantItinerary* and *ContainerStatusMessage* implement container itineraries and CSMs, respectively, extending *Itinerary* and *MovingObjectPosition*. These classes may be extended to represent other types of moving entities and their itineraries, such as vessels, vehicles, persons, cattle. Some of these classes are *abstract*, i.e., non-instanceable, because they include non-implemented methods, because their final implementation depends on the semantics of the particular moving object. This is the case, for example, of method *buildItineraries*, which is called by the container segmentation routine but whose original definition is in class *ItinerarySegmenter*. In the case of containerized transportation, its final implementation leverages the semantics of container events as described above. Other classes, such as *Location*, *Time*, *Coordinates*, *TemporalCoordinates*, are used to represent generic concepts such as time and locations.

IV. Analysing Container Itineraries

The ACID analysis API extends the open source Weka Data Mining toolkit [34] and the SVM library LibSVM [35]. Weka is a Java API that includes a wide collection of machine learning algorithms for data mining. It provides also a graphical user interface for data mining testing (namely, Explorer and Experimenter). LibSVM [35] is an award winner library for SVM that supports both supervised and unsupervised classification, i.e., one-class classification for outlier detection,

Itinerary identifier	Start	Origin	End	Destination	ItineraryT	MStart	PoOrig	MEnd	PoDest	MTripT	Transs
ABCU1234567001	27 May 10	Shangai	16 Jul 10	Antwerpen	50 days	30 May 10	Shangai	03 Jul 10	Antwerpen	34 days	1
ABCU1234567002	20 Aug 10	Antwerpen	10 Set 10	Valencia	21 days	24 Aug 10	Antwerpen	07 Set 10	Valencia	14 days	0

Fig. 6. Example of container itineraries for the history in Fig. 3.

and that includes various kernel functions to parametrise the classification model. The library is available in different languages, including Java, and can be imported in Weka. No other SVM library in the literature compares to LibSVM for number of classification problems and languages supported, therefore it has been widely validated in different application domains. However, has pointed out above, the API enables to analyse only a limited set of data types. For other datatypes, such as strings, targeted kernel implementation are required.

ACID extends Weka and LibSVM to apply one-class classification to container itineraries.

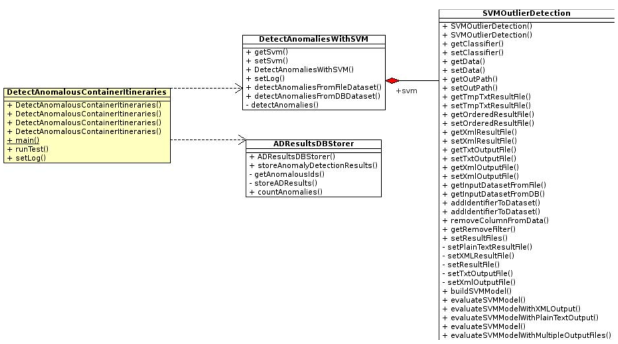


Fig. 7. Excerpt of the ACID class diagram for the data analysis module

In the class diagram in Fig. 7, class *SVMOutlierDetection* wraps Weka classification functionalities defining the methods to set up, build and execute the SVM anomaly detection model, and to store the anomaly detection results in the local file system. Class *DetectAnomaliesWithSVM*, using such methods, implements the corresponding anomaly detection task applying a typical classification flow, as described in the algorithm in Fig. 8. Steps (1-2) create the model, (1) setting the output path and (2) loading the input data of the classification.

At this stage the classification model can be tuned by setting the input data fields that will be evaluated by the classifier and the kernel parameters at step (3). Such parameters specify the kernel function according to the format required by LibSVM.

Moreover, other parameters (3) serve to filter out identifiers from the classification while preserving record's identify. The output parameters specify which columns, taken from the input dataset, have to be included in the output file and the header of the output file. Then, the SVM model is built (4) and executed (5). Cross validation is applied to run the classification. The classification results are stored in the local file system at step (6), in text and XML formats. Class *AnomalyDetectionResultsStorer* integrates the API with the functionalities to copy the classification results in the database.

Relying on these basic functionalities, we can run the anomaly detection task on every dataset of moving object

Anomaly detection

(1) create the local output folder;
(2) load the input dataset;
(3) set the classifier parameters:
  (3a) set the kernel parameters;
  (3b) set the filters on the input dataset;
  (3c) set the output parameters;
(4) build the SVM model;
(5) evaluate the SVM model against the input dataset;
(6) write the results on the local file system .

Fig. 8. Algorithm of the anomaly detection task

ACID analysis algorithm

(1) create the output table to store the anomaly detection results;
(1) prepare the input parameter of the SVM task;
(2) create the handler for output data storage;
(3) create an SVM anomaly detection task;
(4) run classification (**Anomaly Detection** algorithm);
(5) store the classification results in the DB.

Fig. 9. ACID algorithm

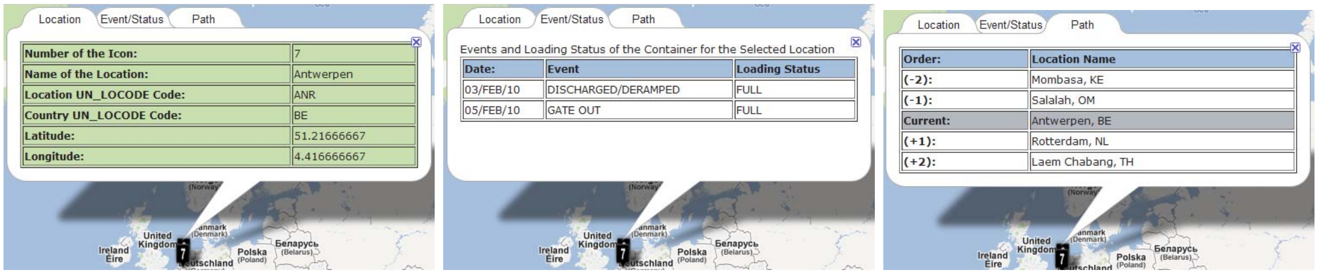
itineraries. For the application case of ACID (cf. the algorithm in Fig. 9), class *DetectAnomalousContainerItineraries* takes in input the dataset of container itineraries generated by the DPM, then calls the methods defined in class *DetectAnomaliesWithSVM* to run the analysis. As discussed in the previous section, the input dataset includes the origin and destination of the shipments, the bounding ports of the maritime itinerary, the duration of the shipment and of the maritime trip, and the number of transshipments. These data defines the itinerary dimensions the classifier takes into account to estimate the dataset distribution. In particular, the classifier evaluates which itineraries fit into the estimated distribution and which are instead the outliers, i.e., the anomalous itineraries.

V. Visualising Anomalous Container Itineraries

The Visualization Module (DVM) is as a web-based application. The application supports two itinerary visualizations: geographical and tabular. The geographical visualization draws the linear interpolation of the itinerary onto a map, and is implemented using Google Maps API [29]. The tabular visualization shows the itinerary details and is realized in JavaScript, using the library jQuery [36].

When the application is launched, the PHP-SQL module connects to the ORACLE database and loads the list of the anomalous container itineraries. Selecting one itinerary from the list, the application visualizes the itinerary information (Fig. 10). The numbered icons in the figure represent the ports the container passed through, and the numbers represent the position of the ports within the itinerary sequence.

Clicking on any icon, an *info-window* appears (Fig. 11) providing detailed information on the corresponding location,



(a) Location (b) Events and load status (c) Path

Fig. 11. Info-windows for locations.

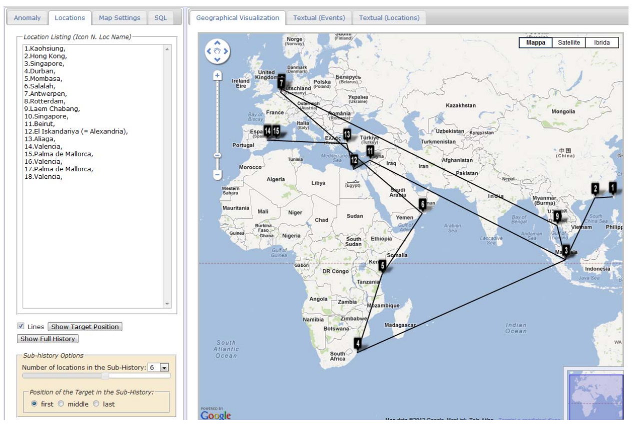
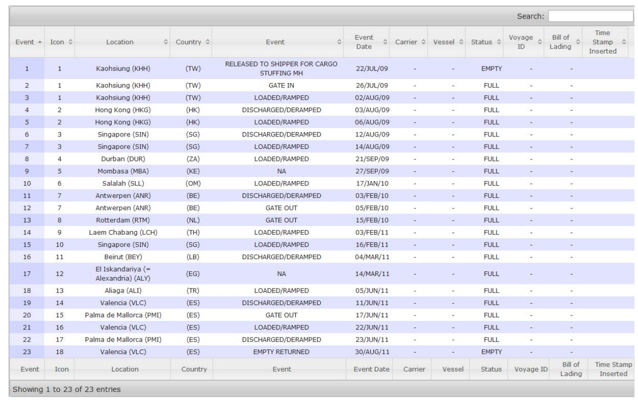


Fig. 10. Geographical visualization of an anomalous container itinerary



Search:

Event	Icon	Location	Country	Event	Event Date	Carrier	Vessel	Status	Voyage ID	Bill of Lading	Time Stamp Inserted
1	1	Kaohsiung (KHH)	(TW)	RELEASED TO SHIPPER FOR CARGO STUFFING MH	22/JUL/09	-	-	EMPTY	-	-	
2	1	Kaohsiung (KHH)	(TW)	GATE IN	26/JUL/09	-	-	FULL	-	-	
3	1	Kaohsiung (KHH)	(TW)	LOADED/RAMPED	02/AUG/09	-	-	FULL	-	-	
4	2	Hong Kong (HKG)	(HK)	DISCHARGED/DERAMPED	03/AUG/09	-	-	FULL	-	-	
5	2	Hong Kong (HKG)	(HK)	LOADED/RAMPED	06/AUG/09	-	-	FULL	-	-	
6	3	Singapore (SIN)	(SG)	DISCHARGED/DERAMPED	12/AUG/09	-	-	FULL	-	-	
7	3	Singapore (SIN)	(SG)	LOADED/RAMPED	14/AUG/09	-	-	FULL	-	-	
8	4	Durban (DUR)	(ZA)	LOADED/RAMPED	21/SEP/09	-	-	FULL	-	-	
9	5	Mombasa (MBA)	(KE)	NA	27/SEP/09	-	-	FULL	-	-	
10	6	Salalah (SLL)	(OM)	LOADED/RAMPED	17/JAN/10	-	-	FULL	-	-	
11	7	Antwerpen (ANR)	(BE)	DISCHARGED/DERAMPED	03/FEB/10	-	-	FULL	-	-	
12	7	Antwerpen (ANR)	(BE)	GATE OUT	05/FEB/10	-	-	FULL	-	-	
13	8	Rotterdam (RTM)	(NL)	GATE OUT	15/FEB/10	-	-	FULL	-	-	
14	9	Laem Chabang (LCH)	(TH)	LOADED/RAMPED	03/FEB/11	-	-	FULL	-	-	
15	10	Singapore (SIN)	(SG)	LOADED/RAMPED	16/FEB/11	-	-	FULL	-	-	
16	11	Beirut (BEY)	(LB)	DISCHARGED/DERAMPED	04/MAR/11	-	-	FULL	-	-	
17	12	El Iskandariya (= Alexandria) (ALY)	(EG)	NA	14/MAR/11	-	-	FULL	-	-	
18	13	Aliaga (ALI)	(TR)	LOADED/RAMPED	05/JUN/11	-	-	FULL	-	-	
19	14	Valencia (VLC)	(ES)	DISCHARGED/DERAMPED	11/JUN/11	-	-	FULL	-	-	
20	15	Palma de Mallorca (PMI)	(ES)	GATE OUT	17/JUN/11	-	-	FULL	-	-	
21	16	Valencia (VLC)	(ES)	LOADED/RAMPED	22/JUN/11	-	-	FULL	-	-	
22	17	Palma de Mallorca (PMI)	(ES)	DISCHARGED/DERAMPED	23/JUN/11	-	-	FULL	-	-	
23	18	Valencia (VLC)	(ES)	EMPTY RETURNED	30/AUG/11	-	-	EMPTY	-	-	
Event	Icon	Location	Country	Event	Event Date	Carrier	Vessel	Status	Voyage ID	Bill of Lading	Time Stamp Inserted

Showing 1 to 23 of 23 entries

Fig. 12. Tabular representation of container itinerary

including: name, UN/LOCODE [37], country, and geographical coordinates (Fig. 11a); the list of container events occurring in that location, with dates and the loading status of the container (Fig. 11b); and the previous and next locations in the container itinerary (Fig. 11c).

Alternatively, the user can visualize the details of the itinerary in a tabular format (Fig. 12) [4].

VI. Experimental Evaluation

We have evaluated the ACID prototype against a real world dataset containing more than 18 millions CSMs, which refer to 50 thousand containers travelling worldwide along a period of 3 years (from January 2009 to January 2012). The dataset has been segmented into more than 300 thousand container itineraries with the data preparation API described in Section III. The segmentation and the analysis tests have been executed on an AMD Athlon® 64 bit X2 Dual Core 3800+ 2GHz with 4GB of RAM running Debian GNU / Linux 6.0. The itinerary segmentation took 45 minutes, while between 15 and 17 minutes were necessary to run the analysis.

In order to estimate the distribution of the dataset, the classifier had been set to use a gaussian kernel function, that can be parametrised with respect to the expected rate of anomalies (i.e., the nu parameter of the gaussian density distribution). As user requirement, we had to keep the number of anomalies under a certain threshold, to be treatable by the final user; on the other hand, the number of anomalies in this application domain is usually very limited. Therefore, we set the classifier to estimate a density distribution tolerating a percentage of outliers around the 0,001%. This setting turns into a request to achieve a very high classification accuracy, i.e., the 99,999% of the itineraries should fit into the density distribution as *typical*, classifiable, data. This threshold was used to find a suitable estimate for cross validation. We increased gradually the number of folds, and we evaluated that a 25-fold cross validation enables an acceptable classification accuracy on our dataset (and, in turn, a limited number of anomalies).

However, the size of the input dataset exceeded the capability of the available memory, therefore we decided to divide the dataset in blocks and to run the analysis iteratively, processing one block at each iteration. As a preliminary subdivision we created blocks of data according to the origin and destination of itineraries, but this created input datasets not homogeneous in size, that resulted in a greater number of anomalies discovered because of the small groups. Similar results have been obtained with blocks of itineraries of containers travelling

[4]Sensitive information such as the name of the carrier, the name of the vessel, etc. have been removed from the figure.

among the same countries.

We obtained a better classification accuracy using blocks of data with the same size. However, before creating the blocks, we ordered the itineraries with respect to the countries of origin and destination. This, combined with a suitable block size, forced the classifier to analyse similar itineraries together. As shown in Table I, enlarging the size of the block of data analyzed by the classifier, the overall classification accuracy (**accuracy (%)**) improves and the overall rate of anomalies (**anomaly rate (%)**) gets closer to the target.

We analysed the results obtained by the classifier comparing the average values of the risk indicators discussed above, i.e., the time of the shipping and of the maritime trip, and the number of transshipments, obtaining a considerable difference, in average, between the itineraries following the distribution and the outliers. In the right side of Table I we reported the rates of itinerary times ($\mathbf{it}_a/\mathbf{it}_n$) and number of transshipments ($\mathbf{tt}_a/\mathbf{tt}_n$) between anomalous and *normal* itineraries, respectively. We can note that anomalous itineraries are usually longer than normal ones (from twice to five times), with a bigger number of transshipments (from 1.19 to more than twice transshipments). However, exceptionally longer itineraries can be created because of some failure in the data collection phase. Indeed, when the CSMs gatherer does not collect all the events of a container, in particular events signalling the begin of a new shipment or the accomplishment of a consignment, the segmentation phase may create a single itinerary merging events that actually belong to different itineraries. For the same reason, incomplete itineraries can be included in the dataset.

Such longer (and shorter) itineraries are marked as anomalous by the classifier, but these should be filtered out, because they are of no interest from an anti-fraud and security point of view. To handle this issue, we included in the system a post-processing phase that discards exceptionally longer and shorter itineraries. The filtering condition we set out takes into consideration also the average number of transshipments. The final results of our experiments are reported in the last column of the table (**filtered anomalies**).

We can note that, increasing of few points the accuracy, longer itineraries have a higher probability to become outliers (cf. the trend of ($\mathbf{it}_a/\mathbf{it}_n$) with respect to **accuracy**). However, these uninteresting cases can be filtered out very easily (146 cases are filtered out in the second test), leaving more interesting anomalous case for evaluation (22 instead of 8, as in the last test). Therefore, weakening the accuracy threshold may be reasonable if a post-processing filtering is applied, because this avoids that interesting cases are discarded by the classifier, while remaining in the boundaries of a tractable number of anomalies.

The final set of anomalies can be furthered analysed by the Customs Authorities using the ACID visualization tool described in the previous section. For example, in Fig. 13 we can see a suspicious case detected by the ACID system. This is a shipment starting from the port of Savannah, US, with destination Callao, Perù. The anomaly in this itinerary lays in the long time taken to accomplish the shipment (32 days), caused by the handling time of the container of the the port of origin, (15 days for good stuffing and container loading). Such a delay may have been caused by exceptional circumstances, but it is enough to raises an alert and to suggest to make further investigations.

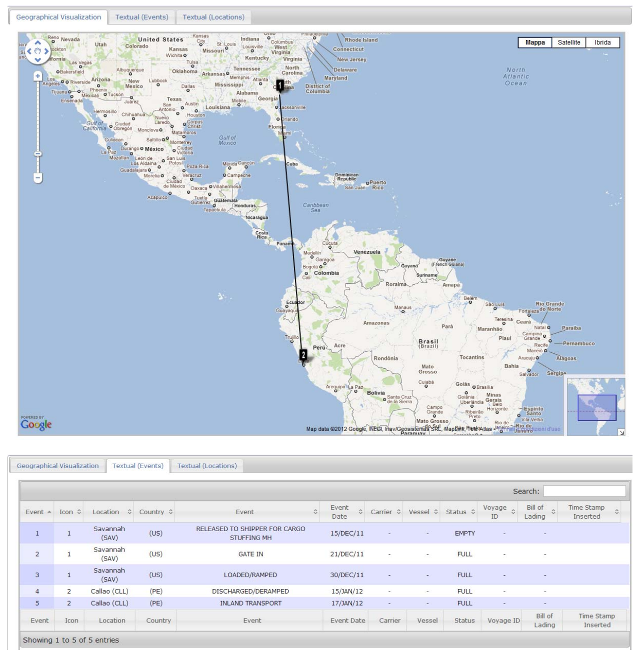


Fig. 13. Example of anomalous itinerary: the shipment is longer than the typical time for the considered route.

VII. Conclusion and Future Work

In this paper we have presented ACID, a system to detect anomalous container itineraries. ACID has a modular design that enables to integrate alternative analysis algorithms and visualization tools. Unfortunately, at the time of this submission we can not yet present a completed experimental evaluation of the system, because the final validation of the anomalies that ACID targets entails a very long and expensive post-clearance process that has to be run by Customs authorities. However, the preliminary tests we have run show the viability of the ACID approach towards the development of risk analysis tools for anti-fraud and supply chain security.

As a further development, the integration in the system of other anomaly detection algorithms (such as clustering and other kernel methods) will enable to improve the targeting capability of the system. In particular, the development of spatio-temporal data mining algorithms and kernel functions for spatio-temporal data to analyse container and vessel trajectories is under investigation. Another important issue to consider is the scalability of the approach. The number of features that can be analysed by a Risk Analysis system, as well as the size of data sets, can be enormous. Therefore, investigating incremental on-line analysis methods for high-

TABLE I
RESULTS OF EXPERIMENTAL EVALUATION USING A GAUSSIAN KERNEL AND 25-FOLD CROSS VALIDATION

nu	block size	exec. time	accuracy (%)	anomalies	anomaly rate (%)	it$_a$/it$_n$ (avg)	tt$_a$/tt$_n$ (avg)	filtered anomalies
0,001%	10000	11’	99.769	720	0.231	2.18	1.19	349
0,001%	20000	12’	99.946	168	0.054	3.15	1,39	22
0,001%	30000	13’	99.972	86	0.0276	4.11	2.09	19
0,001%	40000	15’	99.967	102	0.032	2.81	1.37	11
0,001%	50000	15’	99.986	54	0.017	4.94	2.13	9
0,001%	60000	17’	99.986	42	0.013	5.02	2.51	8

dimensional data is considered a fundamental development for the system.

REFERENCES

[1] Department of Homeland Security, U.S. Customs and Border Protection, “Importer Security Filing and Additional Carrier Requirements - Final Rule,” 2008.

[2] E. Commission, “Commission regulation (ec) no 1875/2006 amending regulation (eec) no 2454/93 laying down provisions for the implementation of council regulation (eec) no 2913/92 establishing the community customs code,” December 2006. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:360:0064:0125:EN:PDF

[3] “Implementing recommendations of the 9/11 commission act of 2007, public law 110–53—aug. 3,” 2007. [Online]. Available: http://intelligence.senate.gov/laws/pl11053.pdf

[4] E. Commission, “Secure trade and 100% scanning of containers,” 2010. [Online]. Available: http://ec.europa.eu/taxation_customs/resources/documents/common/whats_new/sec_2010_131_en.pdf

[5] R. Bergami, “US Customs: The New 10 + 2 Rules,” *International Review of Business Research Papers*, vol. 5, no. 2, pp. 273–282, March 2009.

[6] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Survey*, vol. 41, no. 3, pp. 1–58, 2009.

[7] J. H. M. Janssens, I. Flesh, and E. O. Postma, “Outlier Detection with One-Class Classifiers from ML and KDD,” in *International Conference on Machine Learning and Applications*. IEEE Computer Society, Washington, DC, USA, 2009, pp. 147–153.

[8] H. Hoffmann, “Kernel PCA for novelty detection,” *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, Mar. 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0031320306003414

[9] P. Filzmoser, “A multivariate outlier detection method,” in *Proceedings of the Seventh International Conference on Computer Data Analysis and Modeling*, vol. 1, no. 1989. Minsk: Belarusian State University, 2004, pp. 18–22. [Online]. Available: http://computerwranglers.com/Com531/Handouts/Mahalanobis.pdf

[10] T. Veracini, S. Matteoli, M. Diani, G. Corsini, and U. Pisa, “Fully Unsupervised Learning of Gaussian Mixtures for Anomaly Detection in Hyperspectral Imagery,” in *Intelligent Systems Design and Applications, Ninth International Conference on (ISDA ’09)*, 2009, pp. 596–601.

[11] A. Cansado and Á. Soto, “Unsupervised Anomaly Detection in Large Databases Using Bayesian Networks,” *Applied Artificial Intelligence: An International Journal*, vol. 22, no. 4, pp. 309–330, 2008. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/08839510801972801

[12] C. Piciarelli and G. L. Foresti, “On-line trajectory clustering for anomalous events detection,” *Pattern Recognit. Lett*, pp. 1835–1842, 2006.

[13] M. Last and A. Kandel, “Automated detection of outliers in real-world data,” in *Proceedings of the second international conference on intelligent technologies*. Citeseer, 2001, pp. 292–301. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.15.8466&rep=rep1&type=pdf

[14] K. Smets, B. Verdonk, and E. M. Jordaan, “Discovering Novelty in Spatio / Temporal Data Using One-Class Support Vector Machines,” in *International Joint Conference on Neural Networks (IJCNN)*, June 2009, pp. 2956–2963.

[15] C. Piciarelli, C. Micheloni, and G. L. Foresti, “Trajectory-Based Anomalous Event Detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4633642&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4633642

[16] X. Li, J. Han, S. Kim, and H. Gonzalez, “ROAM: Rule- And Motif-Based Anomaly Detection in Massive Moving Object Data Sets,” in *SIAM International Conference on Data Mining*. SIAM, 2007.

[17] A. Basharat, A. Gritai, and M. Shah, “Learning object motion patterns for anomaly detection and improved object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*. IEEE Computer Society, Washington, DC, USA, June 2008, pp. 1–8. [Online]. Available: http://vision.eecs.ucf.edu/papers/cvpr2008/3.pdf

[18] X. Li, J. Han, S. Kim, and H. Gonzalez, “Anomaly Detection in Moving Object,” in *Intelligence and Security Informatics, SCI 135*. Springer-Verlag Berlin Heidelberg, 2008, ch. 19, pp. 357–381.

[19] R. Laxhammar, G. Falkman, and E. Sviestins, “Anomaly detection in sea traffic - a comparison of the Gaussian Mixture Model and the Kernel Density Estimator,” in *12th International Conference on Information Fusion (ISIF)*. IEEE Computer Society, Washington, DC, USA, Jul. 2009, pp. 756–763.

[20] M. Riveiro and G. Falkman, “Empirical evaluation of visualizations of normal behavioral models for supporting maritime anomaly detection (abstract),” in *Geoviz 2011*, 2011. [Online]. Available: http://www.geomatik-hamburg.de/geoviz/abstracts/42_Riveiro.pdf

[21] L. Etienne, T. Devogele, and A. Bouju, “Spatio-temporal trajectory analysis of mobile objects following the same itinerary,” in *Proceedings of the International*, 2010. [Online]. Available: http://www.isprs.org/proceedings/XXXVIII/part2/Papers/50_Paper.pdf

[22] J. Janssens, E. Postma, and J. Hellemons, Eds., *Maritime Anomaly Detection, Proceedings of the International Workshop*, Tilburg, The Netherlands, June 17 2011. [Online]. Available: http://mad.uvt.nl/mad/mad2011-proceedings.pdf

[23] W. Eberle and L. Holder, “Anomaly detection in data represented as graphs,” in *Intelligent Data Analysis*. IOS Press, 2007, ch. 11, pp. 663–689.

[24] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” 1996. [Online]. Available: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

[25] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[26] J. Shawe-Taylor and N. Cristianini, *Kernel methods for Pattern Analysis*. Cambridge University Press, 2004.

[27] International Container Bureau. [Online]. Available: www.bic-code.org/

[28] I. O. for Standardization. (1995) Freight Containers - Coding, Identification and Marking. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=20453

[29] “Google maps api,” (Accessed in April 2012). [Online]. Available: https://developers.google.com/maps/

[30] Open Geospatial Consortium (OGC), “OpenGIS® Web Map Service Interface Standard (WMS),” 2011, http://www.opengeospatial.org/standards/wms.

[31] “Openstreetmap,” (Accessed in April 2012). [Online]. Available: http://www.openstreetmap.org/

[32] “The Java programming language,” 2011, http://java.com/.

[33] Oracle, “Oracle Database Online Documentation 11g Release 1 (11.1),” 2011.

[34] Weka 3 - Data Mining with Open Source Machine Learning Software. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[35] LIBSVM - A Library for Support Vector Machines. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[36] “Jquery,” (Accessed in April 2012). [Online]. Available: http://jquery.com/

[37] United nations code for trade and transport locations. [Online]. Available: http://www.unece.org/cefact/locode/welcome.html