



HAL
open science

IBRIDIA: A hybrid solution for processing big logistics data

Mohammed Alshaer, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid,
Mohamed Dbouk

► **To cite this version:**

Mohammed Alshaer, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid, Mohamed Dbouk. IBRIDIA: A hybrid solution for processing big logistics data. *Future Generation Computer Systems*, 2019, 97, pp.792-804. 10.1016/j.future.2019.02.044 . hal-02352954

HAL Id: hal-02352954

<https://hal.science/hal-02352954v1>

Submitted on 22 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

IBRIDIA: A Hybrid Solution for Processing Big Logistics Data

Mohammed AlShaer^{b,c}, Yehia Taher^a, Rafiqul Haque^d, Mohand-Saïd Hacid^b, Mohamed Dbouk^c

^a*Université de Versailles Saint-Quentin-en-Yvelines, France*

^b*Université Claude Bernard Lyon 1, France*

^c*Lebanese University, Beirut, Lebanon*

^d*Cognitus, Paris, France*

Abstract

Internet of Things (IoT) is leading to a paradigm shift within the logistics industry. Logistics services providers use sensor technologies such as GPS or telemetry to track and manage their shipment processes. Additionally, they use external data that contain critical information about events such as traffic, accidents, and natural disasters. Correlating data from different sensors and social media and performing analysis in real-time provide opportunities to predict events and prevent unexpected delivery delay at run-time. However, collecting and processing data from heterogeneous sources foster problems due to variety and velocity of data. In addition, processing data in real-time is heavily challenging that it cannot be dealt with using conventional logistics information systems. In this paper, we present a hybrid framework for processing massive volume of data in batch style and real-time. Our framework is built upon Johnson's hierarchical clustering (HCL) algorithm which produces a dendrogram that represents different clusters of data objects.

Keywords: Realtime Processing, Clustering, Big Data, Internet of Things, Logistics, Hierarchical Clustering Algorithm

1. Introduction

2 Lately, with the advent of the Internet of Things (IoT), the operational
3 landscape of the logistics industry is changing. Today, logistics companies

Preprint submitted to Journal Name

December 4, 2018

4 (such as DHL¹ and FedEx²) use various sensors for tracking delivery, main-
5 taining sensitive products, and many other purposes. Sensors assist in tag-
6 ging and connecting factories, ships, and machines. They also allow han-
7 dling real-time events. Additionally, connectivity of “things” enables instant
8 communication between devices via the Internet [1]. This hyper-connected
9 ecosystem promises far-reaching payoffs for logistics operators, their business
10 customers, and end-customers [2]. One of the major advantages of IoT-based
11 ecosystem is that it enables connecting the logistics sensors with external sen-
12 sors such as weather sensors and traffic (GPS) sensors, etc. Furthermore, IoT
13 enables connecting with social media such as Twitter which very often pro-
14 vides important traffic information tweeted by the users. The sensors and
15 social media produce information about events such as accident, weather,
16 natural hazards, heavy road constructions, etc. which are critical to logistics
17 companies. These information can be used to carry out some critical analysis
18 such as predictive analysis to forecast shipment delay or prescriptive analysis
19 to optimize routes to guarantee in-time delivery which increases customer
20 satisfaction and hence guarantees customer retention.

21 Although many solutions were proposed in the last two decades within
22 logistics domain to tackle various problems, delivery delay remained an open
23 issue. Timely delivery is a huge challenge for logistics companies because
24 sometimes delays are caused by factors outside of anybody’s control. De-
25 lay has various impacts such as, customer churn or cancellation of orders
26 which eventually leads to huge losses. Therefore, *timely delivery* is critically
27 important to logistics companies.

28 In recent years, logistics companies have started to investigate how to
29 exploit data in predicting delay. The data driven prediction of delay is gaining
30 popularity. Especially, with the advent of Big Data technologies, the logistics
31 providers are focusing heavily on using streams of events such as accident,
32 high-traffic stemming from external sources such as social media to perform
33 analysis and predict delay in real-time. The real-time prediction of delay
34 enables companies to pro-act such as optimizing route on the fly in (nearly)
35 real-time. We have investigated the requirements of a real-time system which
36 can perform analysis and predict delay. The core requirements are: ability
37 to collect logistics data in real-time from multiple heterogeneous sensors,

¹<http://www.dhl.com/en.html>

²<http://www.fedex.com/us/>

38 social media, and business processes; ability to process data efficiently in
39 real-time or batch-style; a model for analyzing data for predicting the delay;
40 and a model which produces an optimal routing plan to prevent the predicted
41 delay. However, since data is the key element of analysis, efficient processing
42 of data to produce quality dataset is a *sine qua non*. In this paper, we focus
43 on developing a *hybrid* solution which enables efficient processing of data in
44 realtime and batch style. It is worth noting that this paper is an extension
45 of our previous work [3] where we developed batch style data processing. In
46 this paper, we added functionalities that enable real-time processing of data.
47 Realtime processing is strongly required to enable logistics service providers
48 to perform analysis in realtime.

49 Existing data processing approaches (*e.g.*, techniques or algorithms) are
50 not adequately efficient to process data in real-time. The existing solutions
51 are built on classical data processing techniques. Therefore, conventional
52 logistics information systems are not able to process sensor or social media
53 data in real-time because these data flow with high velocity [4]. Additionally,
54 the traditional data processing approaches are not able to deal schemaless
55 data such as text. In the following, we explain the data variety and velocity
56 challenges:

57 • **Variety** denotes different types of data models such as structured (*e.g.*,
58 data stored in relational tables) and semistructured (*e.g.*, JSON and
59 XML). Also, data may not have any structure such as text. Process-
60 ing such a wide variety of data is heavily challenging and conventional
61 information systems are not ready to tackle the variety challenge. Addi-
62 tionally, modern technologies have limited capabilities to tackle variety
63 challenge. For instance, we conducted an experimental study with a so-
64 lution called “Massive Online Analysis” (MOA³) which is an advanced
65 version of one of the most widely used machine learning solution called
66 WEKA⁴. We found that it cannot load tweets (which are texts). The
67 shortcoming of existing solutions leave one very important question
68 to logistics companies: *How to handle different data representations?*
69 Moreover, variety hinders data integration [5, 6, 7]. Data integration
70 is of critical importance within Big Data because it enables correlating
71 events stemming from heterogeneous sources and enables predicting

³<http://moa.cms.waikato.ac.nz>

⁴<http://www.cs.waikato.ac.nz/ml/weka/>

72 delay with higher accuracy.

73 • **Velocity** refers to the speed at which data flow within and across the
74 Web. Unlike the earlier days, data now-a-days is in motion. Millions
75 of records may result in a millisecond. Sensor and social media data
76 move with high speed. There are solutions which can be used to collect
77 high-speed data yet processing them in real-time is heavily challenging.

78 In our solution called *ProLoD* [3], we focused on variety challenge. In
79 this paper, we extended ProLoD to tackle the velocity challenge by adding
80 functionalities for collecting and processing data streams in real-time. With
81 this extension, our solution named *IBRIDIA* has turned into a hybrid solution
82 that is able to process data in both real-time and batch style. *IBRIDIA* relies
83 on extended hierarchal clustering algorithm proposed in [8] for processing
84 data streams flowing at high-speed. The core contribution in this paper
85 include the following:

- 86 • Developing a data streamer which fetch logistics data streams from
87 sources such as social media and sensors.
- 88 • Building a real-time data processing engine that relies on extended
89 agglomerative hierarchical algorithm.

90 The remainder of the paper is organized as follows. In Section 2, we will
91 present a motivating scenario. In Section 3, we will describe the problem we
92 are trying to solve more clearly. In Section 4, we present the work related
93 to our research. Our previous work *ProLoD* and the extension of *ProLoD*
94 which is the realtime data processor *RePLoD* is discussed in Section 5 by
95 presenting the overall solution *IBRIDIA*. In Section 6, we present briefly the
96 implementation of *IBRIDIA*. We showed the results of several experiments
97 between the two data processing components *ProLoD* and *RePLoD* in Section
98 7. We conclude the work in Section 8.

99 2. Motivating Scenario

100 There are different modes of shipment used by logistics service providers
101 including air cargo, ships, and ground cargo (e.g., Lorries and trucks). A
102 single mode of transportation may not be adequate to deliver goods. Espe-
103 cially, a cross-border long-running shipment may include several modes of

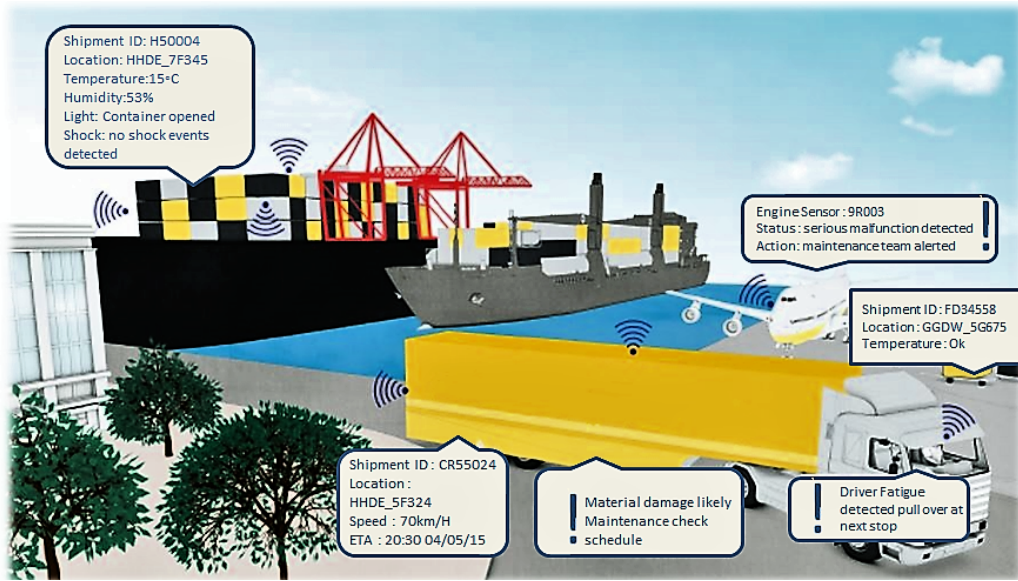


Figure 1: The Multi-modal Logistics System (Source: [9])

104 transportation. Consider a case where a product manufactured in China will
 105 be shipped to different customers located in different cities in the United
 106 States; the shipment process has to be multi-modal which means that the
 107 process will include lorries, trucks, train, ship or air *etc.* (Figure 1).

108 The integrated multi-modal logistics processes are prone to encounter
 109 various challenges namely delivery delay. For instance, *the shipment could be*
 110 *delayed if clearance at the port is delayed*, even if all other modes of trans-
 111 portation meet pre-defined schedule. Uncertain events such as natural disaster,
 112 war, strike, protest may affect one or more of the delivery modes at one
 113 or more steps of the integrated logistics processes. Uncertainty is the major
 114 challenge concerning such events. Therefore, pro-activeness to the best of
 115 our knowledge is a suitable approach which needs continuous streaming of
 116 data that contains information of events that may lead to delivery delay. In
 117 other words, realtime analysis of data to extract information of events which
 118 may lead to delivery delay.

119 3. Problem Description

120 There are different challenges involved in an integrated mission-critical
 121 logistics process. The predominant challenges reported by experts include the

122 followings: in-time delivery, cost optimization, efficient management of inter-
123 modal transportation, transferring information, Security, and Infrastructure
124 [10, 11, 12].

125 However, in-time delivery is one of the key performance indicators (KPIs)
126 of logistics services. Delay of a scheduled (expected) delivery increases cus-
127 tomer dissatisfaction. In order to prevent delay, logistics service providers
128 heavily rely on automated solutions. Business intelligence is a widely used
129 solution that enables performing different types of cycle time analytics [13]
130 that analyze delay for different combinations of goods, routes, modes and
131 weather condition.

132 However, this is a reactive approach which performs analysis on historical
133 data. In other words, traditional business intelligence especially, BI&A 1.0
134 and BI&A 2.0 use only internal data which stem from different information
135 systems and legacy systems [14].

136 Also, the process mining tool P_{RoM} [15] – a recent tool for mining busi-
137 ness processes – lacks the ability to exploit external data. Consequently, the
138 analytics misses important external data such as sensor data (for example,
139 global positioning systems (GPS) data) and social media data (such as Twit-
140 ter data). The advent of Big Data technologies created wide opportunities
141 to exploit such external data which enhances the predictability of analytics.
142 More specifically, these data are effective to forecast potential delivery delay
143 as they contain important information such as high traffic, weather report,
144 political events such as protest, and other events such as unexpected natural
145 disasters (e.g., Earthquake).

146 However, collecting, cleaning, filtering, integrating, and storing data from
147 heterogeneous sources is a non-trivial task. Particularly, a seamless integra-
148 tion of unstructured text sourcing from Twitter with structured business
149 process data is not possible by existing logistics solution frameworks. Dong
150 et al. [16], outlined several Big Data integration challenges. Furthermore,
151 there are several techniques and approaches for processing data, however,
152 our investigation suggests that there is a scope to improve these techniques
153 specifically the clustering algorithms.

154 In this paper, we aim to address the two problems discussed in the above:
155 data preparation and data processing. The goal is to provide efficiently
156 processed data streams which are employed to perform an effective analysis.
157 To that end, we developed IBRIDIA that enables to fetch data from various
158 sources, pre-process and process tasks in real-time.

159 4. Related Work

160 Clustering data in real time has drawn a huge research interests in the
161 recent past, especially with an extensive demand of using analytics on stream-
162 ing data. We choose clustering because we needed to work with unlabeled
163 data which means no model training data set is available and we know noth-
164 ing about the data previously, thus we needed an unsupervised learning
165 model. A few initiatives have been taken in the field where BIRCH (bal-
166 anced iterative reducing and clustering using hierarchies) was one of the
167 basic methods for data stream clustering [17]. Essentially, BIRCH intro-
168 duced micro and macro clustering as two new concepts, and was fabricated
169 to work with traditional data mining techniques but not with voluminous
170 amounts of data sets like data streams. Later, the STREAM algorithm pro-
171 posed by Guha et al. [18, 19] was an extension of classical K-median and the
172 first algorithm known with the ability to perform clustering on entire data
173 streams.

174 In [20], Babcock et al. suggested the sliding window model as an exten-
175 sion to STREAM and thus they changed the concept from one single pass
176 over the data, to the concept of receiving data points as a stream and taking
177 into consideration the points that fall within a specific range representing
178 the sliding window. The CluStream framework was suggested by Aggarwal
179 et al. [21] and it was considered effective in handling data streams; it divides
180 the clustering process into two components: online and offline. The former
181 periodically uses micro clusters to store detailed summary statistics, and the
182 latter uses the summary statistics to produce clusters. Later, Aggarwal et
183 al. [22] suggested HPStream that works on data streams high dimension-
184 ality reduction by means of data projection prior to clustering. Denstream
185 algorithm was proposed in [23] as an extension for DBSCAN where they
186 combined micro clustering concept to the density based connectivity search.
187 Another density-based extension is the D-Stream proposed in [24]. The pro-
188 posed solution maps each new data point to a specific grid upon its arrival;
189 the density information is stored and then clustering is applied to the density
190 data grids. Khalilian et al. [25] suggested an improvement for well-known
191 K-Means algorithm. They applied the widely-known divide and conquer
192 method that is capable of clustering objects with high quality and efficiency.
193 Specifically, the solution is suitable for analyzing high dimensional data, but
194 not for realtime data streams. EStream [26] is a data stream clustering tech-
195 nique, which supports five types of evolution in streaming data. They are as

196 follows: appearance of new cluster, disappearance of an old cluster, split of a
197 large cluster, merging of two similar clusters and changes in the behavior of
198 cluster itself. It uses a decaying cluster structure with a histogram to approx-
199 imate the streaming data. Although the algorithm has the disadvantage of
200 needing an expert intervention to specify many parameters before it works,
201 its performance is better than HPStream algorithm [27].

202 In [28], a multi-level unordered sampling technique was suggested to boost
203 the time performance of fuzzy C-means. The technique is double phased. In
204 the first phase, the random sampling is applied to estimate centroids and then
205 fuzzy C-means “FCM” is performed on the full data with the previously ini-
206 tialized centroids. Fuzzy C-means together with probabilistic clustering were
207 then extended to work on huge data sets by the sampling based proposal of
208 Richards and James [29]. In [30], an algorithm called AFCM was suggested
209 to speed up FCM. This is done using lookup table. In [31], the authors pro-
210 posed several efficient and scalable parallel algorithms for a special purpose
211 architecture description of a modified FCM algorithm known as 2rFCM. A
212 fast FCM algorithm was proposed in [32]. They employed the concept of
213 decreasing the number of distance calculations by checking the membership
214 value for each point.

215 Furthermore, many machine learning libraries are used to implement the
216 algorithms discussed above. We adopt building a connectivity model (hier-
217 archical clustering algorithm) in our framework. The three main reasons for
218 our decision are the following:

- 219 1. No prior knowledge of the nature of the coming data (format, structure,
220 features, etc.)
- 221 2. No prior knowledge of how many categories can the data be classified
222 into (number of clusters is unforeseen)
- 223 3. The clusters probably will evolve with time (keep changing dynamically,
224 i.e., creating, removing, splitting and merging clusters).

225 Since we are interested mainly in hierarchical clustering, we studied the
226 machine learning libraries within the scope of this algorithm. Datumbox
227 [33] is a robust framework that provides different functions like Sentiment
228 Analysis, Twitter Sentiment Analysis, Subjectivity Analysis, Topic Classifi-
229 cation, Language Detection, Keyword Extraction, Text Extraction and Doc-
230 ument Similarity. At low-level, the basic machine learning algorithms such

231 as K-means, hierarchical clustering, and classification algorithms perform
232 the above functionalities. Although this library is very powerful in handling
233 different data types (categorical, numerical, etc.), it was not implemented
234 to work in the environment of streaming data. Therefore, it can read bulk
235 data. Apache Spark [34] is a fast-general-purpose cluster computing system.
236 It supports a rich set of higher-level tools including Spark SQL for SQL and
237 structured data processing, MLlib for machine learning, GraphX for graph
238 processing, and Spark Streaming, For clustering, Spark offers limited fea-
239 tures in particular; it supports few algorithms such as K-Means. The library
240 is missing hierarchical clustering algorithm, which we found suitable for our
241 research project.

242 Furthermore, the library offers the streaming K-means; it is applicable
243 on numerical data only which is a limitation for Big Data where data variety
244 is major challenge. SPMF [35] offers implementations of 120 data mining
245 algorithms for association rule mining, item set mining, sequential pattern
246 mining, and of course clustering and classification. It works only with nu-
247 merical data and this was mentioned explicitly in the documentation. The
248 input is a set of vectors containing double values only, a parameter “max-
249 distance” and a distance function. This implies the same limitation that
250 Spark has. To the best of our understanding, this shortcoming is obvious
251 because, the clustering is usually done according to Euclidean or Manhattan
252 distance functions that need numerical data to be applied.

253 WEKA [36] is a widely-known library. It is an integrated system which
254 consists of a collection of machine learning algorithms for data mining tasks.
255 The algorithms can either be employed directly for a dataset or called from
256 within a Java code. WEKA contains tools for data pre-processing, classi-
257 fication, regression, clustering, association rules, and visualization. It is a
258 well-suited solution for developing new machine learning schemes. It is very
259 efficient and can be used with big data analytics but needs to work only on
260 data at rest and with a specific file format called ARFF. Recently, an initia-
261 tive has been taken to extend WEKA to be used for mining data Streams.
262 MOA (Massive Online Analysis) [37] is an open source library for data stream
263 mining. It includes a collection of machine learning algorithms (classification,
264 regression, clustering, outlier detection, and concept drift detection and rec-
265 ommender systems) and tools for evaluation. MOA provides approximately
266 all clustering algorithms for streaming data but it is confined to a specific
267 format just like WEKA (ARFF file only). It enables to generate synthetic
268 data streams and allows users to visualize data clustering in real-time.

269 Waller and Fawcett [38] underline the importance of data and analytics
270 for supply chain management (SCM). They introduce the term “SCM data
271 science”, referring to Big Data Analytics (BDA), as the “application of quan-
272 titative and qualitative methods from a variety of disciplines in combination
273 with SCM theory to solve relevant SCM problems and predict outcomes, tak-
274 ing into account data quality and availability issues”. Bi and Cochran [39]
275 argue that BDA has been identified as a critical technology to support data
276 acquisition, storage, and analytics in data management systems in modern
277 manufacturing. They attempt to connect IoT and BD to advanced manu-
278 facturing information systems to help to streamline the existing bottlenecks
279 through improving forecasting systems. Similarly, Gong et al. [40] argue
280 that a production control system (PCS) can be considered an information-
281 processing organization (IPO).

282 *Discussion*

283 The solutions and tools mentioned in the state of the art provide a variety of
284 machine learning algorithms that can be used for predictive analytics tasks,
285 such as feature selection, parameter optimization and result validation. Many
286 of these systems offer basic visualizations including residual plots, scatter
287 plots and line charts. However, the visualization feature of these systems
288 is limited to presenting the final results; they do not offer any interactive
289 means for manipulation, feature selection or model refinement; instead, these
290 systems often opt to show baseline models or simple statistical measures for
291 result validation, working as more of a black-box system.

292 SPMF and Spark worked only on numerical data. Additionally, we found
293 that Spark’s MLib library does not have an implementation of hierarchical
294 clustering. Datum box had a specific structure for storing and processing
295 data and it lacks the ability to read data by lines. It can handle data as a
296 batch that can be read in one go and thus it was not suitable for real time
297 environments.

298 WEKA developers extended it into MOA, the version that fits real time
299 processing but it is still very recent and thus had some code bugs that did not
300 allow us to benefit from it⁵. Besides that, MOA is locked into a specific file
301 format which is ARFF and hence it is unable to read other data format. An

⁵We contacted Dr. Albert Bifet the author of the library for assistance because it was only running for a specific number of data points then starts throwing errors but the problem was not solved.

302 ARFF format needs to have attributes, types, and data explicitly mentioned
303 within the file. Nevertheless, in our case the data streamed and fed into the
304 algorithm do not necessarily have an attribute. Rather, data could be a set
305 of records each of which is made up of different text words. Therefore, we
306 decided to write our own implementation.

307 In order to fulfill the needs of the logistics sector, we can use social media
308 to do the data enrichment and combine the data from multiple sources to
309 validate and analyze the current situations. But using social media alone,
310 might not be of major interest, there is a need to build a new model that is
311 enriched from social media and the different sensors to predict the delay for
312 the delivery process and suggest improvement according to it. Even, after
313 predicting the delay, more work should be done using advanced analytics
314 for the optimization of the route planing algorithms. The state of art was
315 missing the data integration to forming up the convenient model from the
316 different data sources which have different data presentations (such as text,
317 JSON, XML, audio, video, etc.)

318 In summary, considering the evaluation of data sources, most of the ex-
319 isting solutions are confined to one data source for analytics and prediction.
320 Additionally, for realtime systems with continuous improvement, the major-
321 ity of researches used large static historical datasets for their testing while
322 our approach does not depend on historical data alone.

323 5. IBRIDIA – Solution Overview

324 In this section, we describe the two main modules of IBRIDIA. In logis-
325 tics system, we have data generated from internal system for stock, orders,
326 shipments, *etc.* Also, there is a need to collect and analyze data from ex-
327 ternal sources in realtime especially to monitor the different statuses of the
328 delivery. To address both needs, within IBRIDIA, we developed a batch style
329 data processing engine that we called *ProLod* and a realtime data processing
330 engine that we called *RePLoD*. We explain these two modules in the follow-
331 ing subsections. Then, we describe the data processing model that IBRIDIA
332 relies on for both realtime and batch style data processing. Figure 2 depicts
333 a high-level architecture of IBRIDIA.

334 In IBRIDIA there are four components: *data streamer*, *the storage*, *batch*
335 *style data processing engine*, and *real-time data processing engine*. The data
336 streamer fetch data from internal and external data sources and ingest them
337 into realtime processing engine and storage. It is worth noting that we used

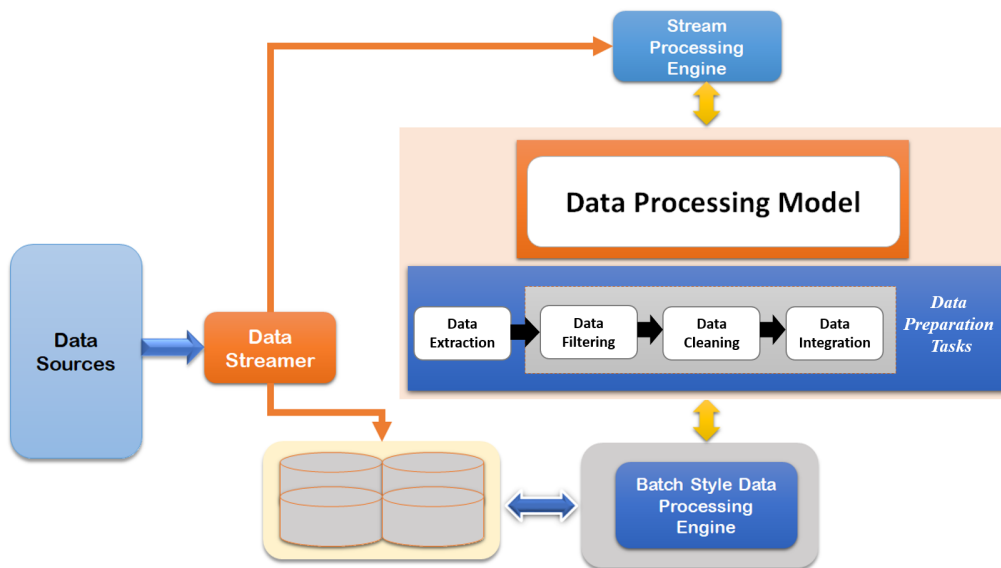


Figure 2: The high level architecture of IBRIDIA

338 native storage in our previous work, however, we developed Hadoop based
 339 scalable storage in IBRIDIA so that a massive scale data can be stored. The
 340 batch style data processing engine (ProLoD) *reads/extracts* data from stor-
 341 age and perform data preparation and processing tasks including cleansing,
 342 filtering, etc. In case of realtime processing, the streamer sends data directly
 343 into the processing engine (RePloD) which carries out processing tasks in
 344 realtime. In the following, we provide more detail about the data processing
 345 engines.

346 5.1. ProLod – Batch Style Data Processing Engine

347 ProLoD represents the batch-style processor for processing of logistics
 348 data stemming from multiple heterogeneous sensors (that include vehicle
 349 sensor, weather sensor, etc.), logistics applications, microblog (e.g., Twit-
 350 ter), and social media (e.g., Facebook). ProLoD comprises two phases: data
 351 preparation phase and processing. The former consists of data extraction
 352 (collection), data cleansing, data filtering, data integration, and data stor-
 353 age. In the latter phase, well-prepared data are clustered. Figure 2 shows
 354 different functionalities of these phases. ProLoD relies on different machine
 355 learning techniques specifically the clustering techniques for data processing.
 356 ProLoD includes five components: data extractor, data cleaner, data filter,

357 data integrator, and data storage for performing data preparation functions.
358 It has a data processor which performs clustering in the second phase.

359 *5.2. RePLoD – Realtime Data Processing Engine*

360 RePLoD represents the core component in our framework for processing
361 the realtime data. As the speed of events from sensors and social media
362 increases, it creates an emerging need for fast processing known as stream
363 processing mechanism. Events may lead to catastrophic consequences if not
364 handled properly in-time. RePLoD was designed to add the missing func-
365 tionalities to the system by adding a convenient way for handling the events
366 generated in realtime. These events are first enqueued into the memory
367 through distributed messaging system. The memory was used instead of the
368 disk because its access speed is faster than the disk by 100,000 times. In this
369 way, we prevent any overwhelming of the receiver and we guarantee fault
370 tolerance in case of any failure. This added feature prevents the loss of any
371 of the data due to their fast generation. Batch processing is not always the
372 right way to do it, sometimes it is important to do the processing on the fly
373 as soon as the events arrive to the servers. These cases can be faced in real
374 world scenarios such as accidents occurring now on roads, bad weather, main-
375 tenance of buildings which need to be notified for the driver in realtime to
376 prevent the catastrophic effects due to delay in delivery. These facts carried
377 us to extend the processing behavior to be able to do the required processing
378 in realtime without doing it in batches. RePLoD performs clustering of the
379 events in realtime and gets immediate insights over the processed data.

380 *5.3. Data Preparation Tasks*

381 Both ProLoD and RePLoD perform five data preparation tasks using two
382 different approaches namely batch style and realtime respectively. These
383 tasks are explained in the following.

- 384 • *Data Extraction:* It is the systematic approach to gather and measure
385 information from a variety of sources to get a complete and accurate
386 picture of an area of interest. The data extractor works with both inter-
387 nal and external sources of data. The internal data sources are typically
388 the information systems used by the users. Consider a user that has an
389 information system consisting of a supply chain management (SCM),
390 a customer relationship management (CRM), a logistics management
391 system, and an account management system (AMS). These systems

392 produce a large amount of data that are collected by the data extrac-
393 tor. It also fetches data from external sources such as Twitter, traffic
394 sensors, weather sensors, Facebook and other social medias. In addi-
395 tion, IBRIDIA’s processing components extract archived sensor data of
396 completed logistics processes. In most of the cases, we found that data
397 extraction from internal sources is more trivial than external ones. Ad-
398 ditionally, internal data were transferred faster than the external ones.
399 IBRIDIA can collect structured and unstructured data. For instance, it
400 collects unstructured texts from Twitter, and Facebook and structured
401 business process data from logistics information system.

402 • *Data Filtering*: It refers to a wide range of strategies or solutions for
403 refining data sets. Datasets are refined into simply what a user (or set
404 of users) needs, without including other data that can be repetitive,
405 irrelevant or even sensitive. ProLoD and RePLoD aim to eliminate
406 all possibilities of data overloading which can increase computational
407 cost and effort during data processing and may jeopardy the analysis
408 regarding accuracy. They collect data that are related to logistics and
409 specifically the data chunks whose hashtags (the words prefixed by #)
410 determine direct and indirect connections with transportation, delivery,
411 logistics, shipment, etc. Consider the term “protest” which may be a
412 political protest or else but can have a great impact on delivery of goods
413 and hence can delay the delivery. However, consider a tweet “the New
414 York stock prices are extremely high today” which will be removed
415 by the data filter because it does not carry any information related to
416 logistics processes.

417 • *Data Cleaning (i.e. data scrubbing)*: It is the process of detecting
418 and correcting (or removing) corrupted or inaccurate records from a
419 record set, table, or database. ProLoD and RePLoD clean data from
420 all unwanted symbols, numbers, stopping words, hashtags, and any
421 other data items that might lead to noise and cause inaccuracy. Figure
422 3 shows an example of cleaning Twitter data using ProLoD.

423 • *Data Integration*: In IBRIDIA, data integration is performed in two
424 steps. In the first step, the data are transformed from source to target
425 serialization format. Currently, the target format is CSV. The second
426 step is merging the transformed data.

Six car #accident in #westsacramento on WB I-80 & Capitol Ave. Reports of people injured & lanes blocked #traffic <https://t.co/S29obp3ByL>
 Police now on scene with an #accident reported on Baird Rd Penfield Rd in #Penfield #traffic #ROC
 #accident reported on Salmon Creek Rd Colby St in #Sweden #traffic #ROC
 #traffic 06:47: #A4 - #accident between LATISANA S.GIORGIO towards TRIESTE
 4 car smash on m4, right lane 500m after Merrylands on ramp. Avoid right lane.
 #traffic #accident @channelten @Channel7 @9NewsSyd
 #traffic 09:56: #A4 - #queuing traffic between PORTOGRUARO S.STINO towards VENEZIA due #accident
 #traffic 09:55: #A4 - #accident 449.4 between PORTOGRUARO S.STINO towards VENEZIA
 #traffic #A4 - #accident 449.4 between PORTOGRUARO S.STINO towards VENEZIA
<https://t.co/8zF918j0ft> <https://t.co/XeFNpGoMA8>

Figure 3: An example of cleaning data with ProLoD.

- 427
428
429
 • *Data Storage*: This step aims to deal with the storage of the integrated datasets. After preparing the integrated datasets, ProLoD and RePLoD store data into the storage.

430 5.4. Data Processing Model

431
432
433
434
435
436
437
438
439
440
441
442
443
 As mentioned earlier, IBRIDIA relies on the data processing model which we developed in our previous work [3]. We explain the data processing model in this section. Choosing techniques or methods for developing the model is not a trivial job. There is an exhaustive list of techniques available from machine learning, data mining, and statistics. In our case, we considered the nature of data and operation styles to choose the right technique for building data processing model. Our data processing model relies on *unsupervised learning techniques* [41]. Unsupervised learning is a machine learning approach in which a system only receives input (x_1, x_2, \dots, x_n) without any corresponding (supervised) output (which is also called *labeled output*). *Clustering* and *dimensionality reduction* are the two most well-known unsupervised learning techniques. We choose *clustering* for our model because the objective function is expected to produce a clustered dataset which facil-

444 itates efficient analysis in prediction of *delivery delay*. Clustering is a process
445 of grouping or segmenting data items that are *similar* between them in a
446 cluster and *dissimilar* to the data items that belong to another cluster [41].

447 There are different types of cluster models which are grouped into *Con-*
448 *nectivity models*, *Centroid model* and *Distribution models*, *Density models*,
449 *Subspace model*, *Group model*, and *Graph-based models* [42]. We are inter-
450 ested in techniques used for building connectivity model which fits to our
451 objective more than the others. *Hierarchical clustering* is a widely used
452 approach for building connectivity model based on distance connectivity be-
453 tween the data items. It is a process of producing a sequence of nested
454 cluster ranging from *singleton clusters* of individual points to an all-inclusive
455 cluster [43]. The hierarchy of the clusters are graphically represented by a
456 dendrogram [44]. There are two approaches to develop a hierarchical cluster
457 model:

- 458 • *Agglomeration* refers to an approach that start with the points as indi-
459 vidual clusters and, at each step, merge the closest pair of clusters. It
460 is also known as *Bottom-Up approach*.
- 461 • *Divisive* refers to an approach that starts with one, all-inclusive clus-
462 ter and, at each step, splits a cluster until only singleton clusters of
463 individual points remain. It is also known as *Top-Down approach*.

464 We found agglomerative hierarchical clustering approach for our solution
465 because the bottom up approach is more flexible than the others in terms
466 of choosing the number of clusters. The algorithm groups data one by one
467 based on the nearest distance measure of all the pairwise distance between the
468 data points. The distance between the data points is recalculated iteratively.
469 However, the choice of distance to consider for grouping data points is a
470 critical matter. Several methods are available to address this question. These
471 methods – found in [45] – are summarized in the following:

472 **Definition 1.** *Single-linkage:* $d(C_i, C_j) = \min_{x \in C_i, x' \in C_j} d(x, x')$. It is equiv-
473 alent to the minimum spanning tree algorithm [46]. One can set a threshold
474 and stop clustering once the distance between clusters is above the threshold.
475 *Single-linkage tends to produce long and skinny clusters.*

476 **Definition 2.** *Complete-linkage:* $d(C_i, C_j) = \max_{x \in C_i, x' \in C_j} d(x, x')$. Clusters
477 tend to be compact and roughly equal in diameter.

478 **Definition 3.** Average distance: $d(C_i, C_j) = \frac{\sum_{x \in C_i, x' \in C_j} d(x, x')}{|C_i| \cdot |C_j|}$.

479 **Definition 4.** Wards method $d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2$ is the sum
480 of squared Euclidean distance is minimized.

481 The iteration is continued by grouping data items until a cluster is formed.
482 As mentioned earlier that the clusters are presented graphically by a *dendro-*
483 *gram* which allows to calculate the number of clusters that should be pro-
484 duced, at the end. There are several variants of the agglomerative hierarchi-
485 cal clustering algorithm. Below we present the steps involved in performing
486 an agglomerative hierarchical clustering. Consider a set of data points $\mathcal{S} =$
487 $(x_1, x_2, x_3, \dots, x_n)$ as input. The agglomerative hierarchical clustering algo-
488 rithm performs the following steps:

489

- *Step 1:* Disjoint cluster (\mathcal{C}) of level $\mathcal{L}(0) = 0$ and sequence number $\mathcal{M} = 0$
- *Step 2:* Calculate the least distance (\mathcal{D}) pair of clusters in the current \mathcal{C} , say pair $\mathcal{P}(r, s)$, according to $\mathcal{D}(r, s) = \text{Min}(\mathcal{D}(i, j))$ where the minimum is over all pairs of clusters in the current clustering
- *Step 3:* Increment the sequence number, $\mathcal{M} = \mathcal{M} + 1$
- *Step 4:* Merge $\mathcal{C}(r)$ and $\mathcal{C}(s) \rightarrow \mathcal{C}(z)$ which is a new cluster. Set the level of this clustering to $\mathcal{L}(z) = \mathcal{D}(r, s)$
- *Step 5:* Update the distance matrix Ψ , (delete the rows and columns corresponding to clusters $\mathcal{C}(r)$ and $\mathcal{C}(s)$ and add a row and column corresponding to $\mathcal{C}(z)$). The distance between the new cluster, denoted (r, s) and the old cluster (k) is defined as follows: $\mathcal{D}((k), (r, s)) = \text{Min}(\mathcal{D}((k), (r)), \mathcal{D}((k), (s)))$
- *Step 6:* Repeat until ONLY one cluster remains.

490

491 In [3], we reported several disadvantages of the basic agglomerative cluster-
492 ing algorithm. In particular, undoing is not allowed and the time com-
493 plexity is $\mathcal{O}(n^2 \log n)$ where n denotes the number of data points. For a large

494 dataset, the performance with respect to processing time may not be satis-
495 factory. Based on the type of distance matrix chosen for merging, different
496 algorithms may have one or more of the following drawbacks: (i) sensitiv-
497 ity to noise and outliers, (ii) partitioning a large cluster, (iii) difficulty in
498 handling different sizes of clusters and handling convex shapes. In this al-
499 gorithm, no objective function is directly minimized. Furthermore, in some
500 cases identifying the correct number of clusters by the dendrogram can be very
501 difficult. Therefore, the basic algorithm agglomerative clustering algorithm
502 is not suitable for clustering data. Hence, we choose extended agglomera-
503 tive hierarchical algorithm proposed in [8]. We intend to use the hamming
504 distance as a measuring criteria in our algorithm, because it can be used as
505 a convenient measuring mechanism for string values which covers most of
506 the unstructured data. Hamming distance measures the minimum number
507 of substitutions required to change one string into the other (the minimum
508 number of *errors* that could have transformed one string into the other). We
509 modified Johnson’s Hierarchical Clustering algorithm to become a stream
510 clustering algorithm that supports incremental grouping of text messages
511 according to their similar characteristics directly on the go. The theoretical
512 steps performed by the modified algorithm is based on the theoretical steps of
513 any agglomerative hierarchical clustering algorithm as shown previously. We
514 identify the practical algorithmic steps of the clustering used in our solution
515 IBRIDIA as follows:

- 516 • **Step 1:** Read new data streams.
- 517 • **Step 2:** Put the *unique* items in the vector format.
- 518 • **Step 3:** Fill a matrix of absence and presence of items.
- 519 • **Step 4:** Calculate hamming distance.
- 520 • **Step 5:** Update the distance matrix.
- 521 • **Step 6:** Create Cluster using minimum distance.
- 522 • **Step 7:** Repeat **until** only one cluster remains.

523 In what follows we explain the above steps using an example where we
524 illustrate how IBRIDIA data processing model works. It begins with reading
525 records. Since data is read from the first row, thus the attribute names do
526 not exist; we expressed them here just to make the data set meaningful to
527 the reader.

528 • Start with each record as a cluster on its own.

529 • Read new data streams

530

531

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01
--	-------------------	-----------------	-------------	---------

532 • A unique item is added in the vector format.

533 • Fill in the matrix of items absence and presence.

534

535

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01
Rec1	1	1	1	1

536 • Build the similarity matrix using hamming distance. Currently, there
537 is only one record.

538 – The algorithm reads new record.

539

540

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01
	NetworkManagement	4/1/2017 8:00	NarrowLanes	LYON-06

541

542 – Place the new Unique items.

543

544

NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01	4/1/2017 8:00	LYON-06
-------------------	-----------------	-------------	---------	---------------	---------

545

546 – Update the matrix.

547

548

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01	4/1/2017 8:00	LYON-06
Rec1	1	1	1	1	0	0
Rec2	1	0	1	0	1	1

549

550 – Build similarity matrix using hamming distance

551 * The Hamming distance can only be calculated between two
552 strings of equal length. String 1: 111100 String 2: 101011.

553 * Compare the bits of each string with the other.

554 * If they are the same, record a “0” for that bit.

555 * If they are different, record a “1” for that bit.

556 * Compare each bit in succession and record either “1” or “0”
557 as appropriate.

558 * Add all the ones and zeros in the record together to obtain the
 559 Hamming distance. Hamming distance = 0+1+0+1+1+1=
 560 4.

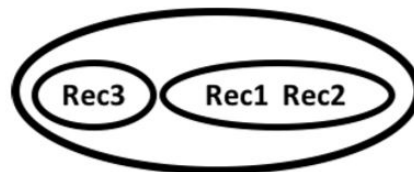
561 – Update the distance matrix.

	Rec1	Rec2
Rec1	0	4
Rec2	4	0

563 • Create a cluster with minimum distance.



564 • The systems read new records and the previous steps are repeated. At
 565 the end a new cluster is created.



566 The iteration stops at this step because the execution loop produces a
 567 single cluster and no cluster can be created any further. We discuss the
 568 implementation of IBRIDIA in the next section.

569 6. Implementation of IBRIDIA

570 We studied various technologies for implementing IBRIDIA. We investi-
 571 gated existing libraries for data extraction, filtering, and transformation. Our
 572 goal was to reuse existing ones instead of developing the new ones. Also, we
 573 studied machine learning libraries including DatumBox⁶, SPMF⁷, Massive
 574 Online Analysis (MOA), and Spark MLlib⁸ to implement our data processing
 575 model. From our study, we found that existing libraries could not be used
 576 to implement our model (discussed in the previous section). Therefore, we

⁶<http://www.datumbox.com>

⁷<http://www.philippe-fournier-viger.com/spmf/>

⁸<http://spark.apache.org/mllib/>

577 decided to implement the model on our own. For implementation, we used
578 Java language on Eclipse.

579 To sum up, IBRIDIA is a framework that integrates three APIs for ex-
580 tracting external data from different sources including Twitter API, Facebook
581 API, and Open Weather API. It uses an open source parser. Also, it includes
582 tools for cleaning and transforming incoming data. The prototype of ProLoD
583 is available in GitHub.

584 We investigated different data processing frameworks including Apache
585 Spark⁹, and Apache Storm¹⁰ to develop RePloD module of IBRIDIA. To
586 the best of our understanding Storm is more potential computation system
587 for our ProLoD. It is fast, can process over a million of tuples per second.
588 It is scalable, fault-tolerant, guarantees our data will be processed, and is
589 easy to set up and operate. Storm integrates with the queuing and database
590 technologies we already use. A Storm topology consumes streams of data and
591 processes those streams in arbitrarily complex ways. However, repartitioning
592 the streams between each stage of the running computation is needed.

593 RePloD consists of two main components: *data streamer* and *data pro-*
594 *cessor*. We implemented the *Data Streamer* using Apache Kafka¹¹ and data
595 processing engine using Apache Storm¹².

- 596 • *Apache Kafka*: It is a publish-subscribe based fault tolerant messaging
597 system. It is fast and highly scalable distributed messaging technol-
598 ogy. It is used in building durable data collection system where high
599 throughput and reliable delivery of messages are critically important.
600 Apache Kafka messaging system is merely a collection of topics split
601 into one or more partitions. A Kafka partition is a linearly ordered
602 sequence of messages, where each message is identified by their index
603 (called as *offset*). All the data in a Kafka cluster is the disjointed
604 union of partitions. Incoming messages are written at the end of a
605 partition and messages are sequentially read by consumers. Durabil-
606 ity is provided by replicating messages to different brokers. Apache
607 Kafka provides four different types of APIs. The Producer API allows
608 RePloD to publish a stream of records to one or more Kafka topics.

⁹<http://spark.apache.org>

¹⁰<http://storm.apache.org>

¹¹Apache Kafka: <https://kafka.apache.org>

¹²Apache Storm: <http://storm.apache.org>

609 The Consumer API allows RePLoD to subscribe to one or more top-
610 ics and process the stream of records produced to them. The Streams
611 API allows RePLoD to act as a stream processor, consuming an input
612 stream from one or more topics and producing an output stream to
613 one or more output topics, effectively transforming the input streams
614 to output streams. The Connector API allows building and running
615 reusable producers or consumers that connect Kafka topics to existing
616 applications or data systems. For example, a connector to a relational
617 database might capture every change to a table.

618 • *Apache Storm*: Apache Storm is a distributed realtime computation
619 system. Storm makes it easy to reliably process unbounded streams of
620 data for realtime processing. It is designed to process vast amount of
621 data in a fault-tolerant and horizontal scalable method. It is a stream-
622 ing data framework that has the capability of highest ingestion rates.
623 Though Storm is stateless, its distributed environment and cluster state
624 is managed by Apache ZooKeeper¹³. It is simple and you can execute
625 all kinds of manipulations on real-time data in parallel. Apache Storm
626 guarantees that every message will be processed through the topology
627 at least once.

628 Apache Storm consists of four main components: *tuple* is the main data
629 structure which is a list of ordered elements; *stream* is an unordered
630 sequence of tuples; *spouts* are the sources of stream; *bolts* are logical
631 units. Bolts can perform the operations of filtering, aggregation, join-
632 ing, interacting with data sources and databases. Bolt receives data
633 and emits to one or more bolts. Spouts and bolts are connected to-
634 gether and they form a topology. Real-time application logic is specified
635 inside Storm topology. In simple words, a topology is a directed graph
636 where vertices are computation and edges are stream of data.

637 The data processing topology of RePLoD comprises the *data cleaner*, the
638 *data filter*, the *data transformer*, and the *data clustering* component. Figure
639 4 shows the data processing topology of RePLoD.

640 The spouts and bolts RePLoD Topology constitute a directed acyclic
641 graph (DAG). Spout is the entry point to the topology used to read the data
642 from Apache Kafka. The Kafka-spout acts as Kafka consumer of the Kafka

¹³<https://zookeeper.apache.org>

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Erro
kafka_spout	2	2	40	40	0.000	0	0		

Showing 1 to 1 of 1 entries

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
realtime-clustering	2	2	0	0	0.015	446.000	20	0.000	0	0
stanford_nip	2	2	20	20	0.000	13.000	20	0.000	0	0
twitter_analytics	2	2	20	0	0.000	5.000	40	0.000	0	0
twitter_cleaner	2	2	40	40	0.000	4.000	40	1.500	40	0
twitter_filter	2	2	40	40	0.000	2.000	20	6.000	20	0
twitter_transformer	2	2	20	20	0.000	1.000	20	0.000	0	0

Showing 1 to 6 of 6 entries

Figure 4: The Data Processing Topology of RePLoD

643 topic. The Kafka-spout reads all the messages ingested into the Kafka topic
644 such as “tweets”. This spout acts as the only connector between Kafka and
645 Storm but what is interesting is the ability to execute every component of the
646 spout and bolt within multiple executors. Data are inserted into the storm
647 topology which consists of three bolts: `twitterfilter`, `twittercleaner`,
648 and `twitteranalytics`. The `twitterfilter` is used for filtering the at-
649 tributes of the tweets records. The tweets are consisting of several attributes;
650 however, not all of them are important for the analysis. Therefore, we
651 need to filter the relevant ones to our analysis such as “id, user, descrip-
652 tion, text, created time, location, etc”. Then the simplified records that are
653 emitted from the bolt `twitterfilter` are ingested as input to the next bolt
654 `twittercleaner` which is used for cleaning all the characters that may affect
655 the analysis. The analysis is carried out by the bolt `real-time-clustering`
656 that is built on the real-time clustering algorithm that results in different
657 clusters: merged, splitted or newly created. Once the data is cleaned, they
658 are transformed the texts (e.g, tweets) to csv-like structure using “twitter-
659 transformer” bolt. After the transformation is completed, we extract differ-
660 ent named entities to understand the text content and make the analysis able
661 to depend upon the features mentioned in the content. Finally, the cluster-
662 ing is carried out in real-time using `twitteranalytics` bolt which is built
663 on hierarchical clustering algorithm that produces clusters in real-time. The
664 clusters are saved in the disk.

665 **7. Experiments**

666 In this section, we discuss the results produced through experiments with
667 ProLoD and RePLoD. We evaluate the performance of ProLoD and RePLoD
668 over the metric execution time. Given below is the specification of the ma-
669 chine we used for our experiments:

- 670 • Processor: 2.40 GHz
- 671 • Memory: 4GB
- 672 • HDD: 500 GB
- 673 • Operating System: Windows 10(64 bit)

674 We compare the performance of our model with the one implemented by
675 WEKA. Although we tested the performance of SPMF and Spark, unfor-
676 tunately, we could not compare them to our work since they can only be
677 applied to numerical data. Concerning MOA, we found bugs in it and thus
678 we could not run our model. It allows only ARFF file formats as mentioned
679 before and even though we converted our file to the needed format, it throws
680 multiple exceptions when we tried to read data from an external file.

681 We implemented two different versions of our model: real-time and Batch
682 style. We tested both versions with a test dataset. RePLoD reads data
683 by records and clusters each incoming record. The clusters are mutable; a
684 cluster may change when a new record is added in the cluster. However,
685 since the algorithm is greedy, the execution time has a positive correlation
686 with number of records, i.e., the execution time increases as the number of
687 records increases. This can be solved using the scalability of the system by
688 inserting more nodes to the cluster for faster processing resources. Table 1
689 shows the result of our experiment with the realtime version “RePLoD”.

Table 1: The result of an experiment with realtime version “RePLoD”

Realtime	1st exec	2nd exec	3rd exec	4th exec	5th exec	average	seconds
8 records	466	426	451	432	420	439	0.439
16 records	3665	2301	2362	2069	2089	2497.2	2.4972
24 records	4363	4311	4491	4386	4135	4337.2	4.3372
32 records	7915	7926	7789	7784	7710	7824.8	7.8248
40 records	12861	12780	12849	12969	12958	12883.4	12.8834

690 The batch-style ProLoD performs bulk reading and clusters batch data.

691 The reading and processing occur only once per batch. Table 2 shows the
 692 results of batch style version “ProLoD”.

Table 2: The result of an experiment with the batch style version of “ProLoD”

Not Realtime	1st exec	2nd exec	3rd exec	4th exec	5th exec	average	seconds
8 records	132	129	136	121	147	133	0.133
16 records	262	286	257	251	286	268.4	0.2684
24 records	465	393	427	393	429	421.4	0.4214
32 records	555	549	617	560	535	563.2	0.5632
40 records	728	719	779	855	726	761.4	0.7614

693 We compared the performance of both versions. Figure 5 shows the com-
 694 parison. According to our experiments, we observed that the batch-style
 695 version performs better than the real-time version of IBRIDIA. Our study
 696 reveals that performance of the real-time version was not satisfactory due to
 697 the centralized environment of the experiment consisting of only one node.
 698 We believe that performance will improved significantly in a distributed and
 699 scalable computation framework with multiple nodes. We believe that tech-
 700 nologies which we used for implementation of our solution called Apache
 701 Storm has high computational power; hence, it can process a huge number
 702 of records within a unit of time *e.g.*, a millisecond. In addition, we assume
 703 that the performance of the batch-style version of ProLoD may decrease if
 704 the dataset is large. Currently, the dataset is small; therefore, the size of
 705 each batch is small and faster in processing (clustering).

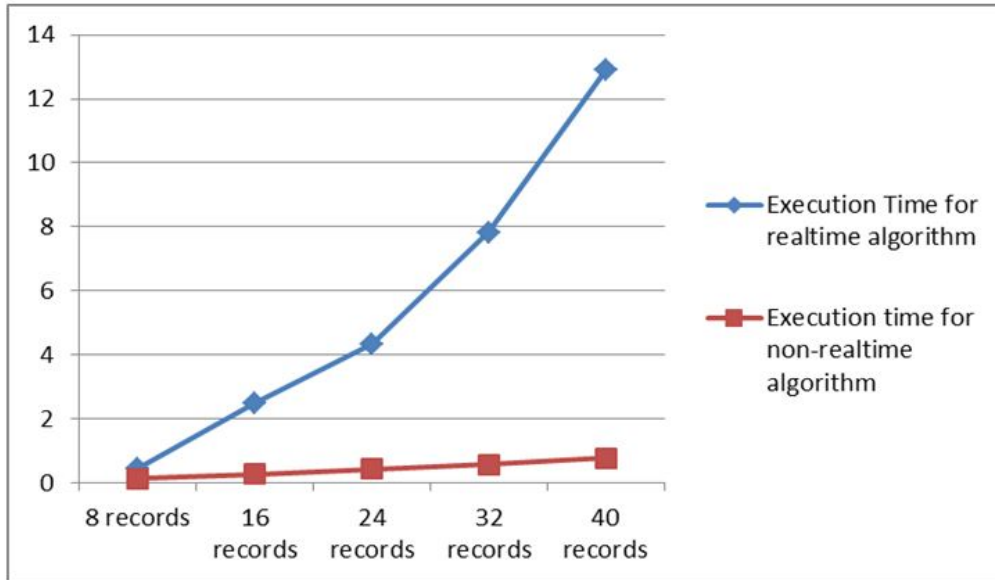


Figure 5: Execution time (in seconds) of RePLoD and ProLoD

706 We compared our model with the hierarchical clustering algorithm of
 707 WEKA. The WEKA algorithm produced 7 clusters with 93 records. Table 3
 708 shows the result which is also visualized in Figure 6 in-which the distribution
 709 of the records is also presented as percentage for each cluster separately.

Table 3: The result produced by WEKA Hierarchical Clustering algorithm

Clusters	Records
Cluster 1	2 (2%)
Cluster 2	8 (9%)
Cluster 3	37 (40%)
Cluster 4	12 (13%)
Cluster 5	3 (3%)
Cluster 6	4 (4%)
Cluster 7	27 (29%)

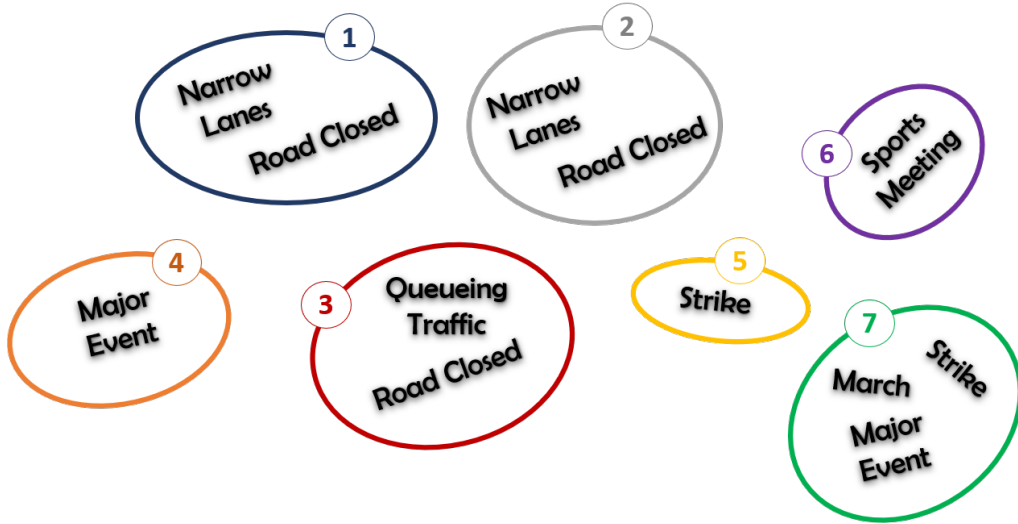


Figure 6: WEKA Text Clustering Results.

710 Furthermore, we observed that both real-time and batch style versions
 711 of ProLoD produced consistent and representable clusters that will assist
 712 in exploring data, discovering insights, and supporting predictive analytics
 713 when data distribution is observed. Table 4 shows the result which is also
 714 shown in Figure 7 in-which the distribution of the records is also presented
 715 as percentage for each cluster separately.

Table 4: The result produced by RePLoD Hierarchical Clustering algorithm

Clusters	Records and Clusters
Cluster 1	10 (11%)
Cluster 2	31 (33%)
Cluster 3	Cluster 1 and Cluster 2
Cluster 4	50 (54%)
Cluster 5	Cluster 3 and Cluster 4
Cluster 6	2 (2%)
Cluster 7	Cluster 5 and Cluster 6

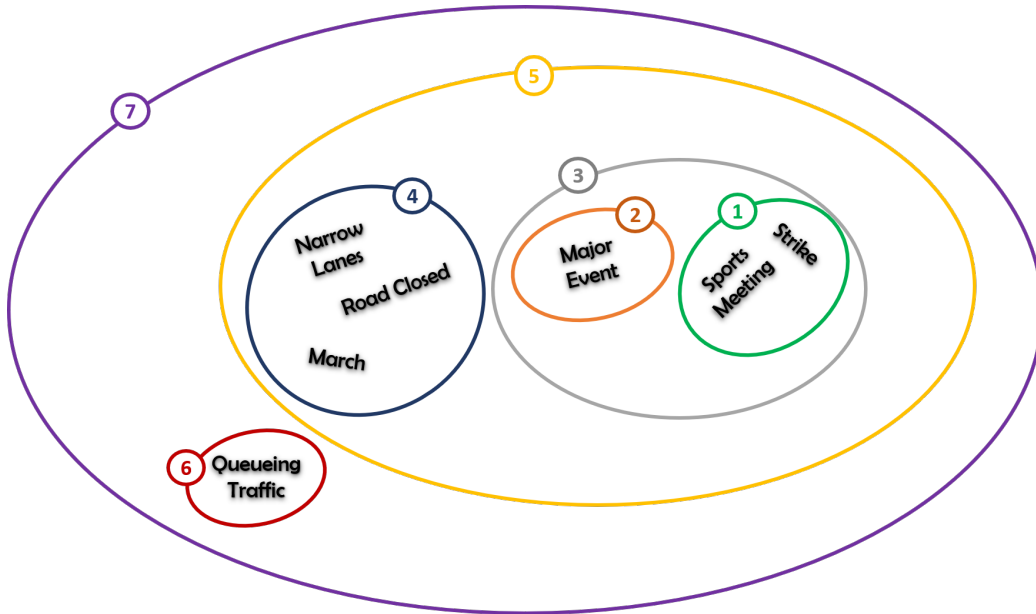


Figure 7: RePLoD Text Clustering Results.

716 According to our analysis, any labeling process using any of the attributes
 717 from the dataset will produce results that are representable and understand-
 718 able (by representable we meant a reasonable number of clusters produced
 719 by the algorithm). Many of the clusters produced by WEKA intersect with
 720 other clusters while treated independently. For example cluster 1 and cluster
 721 2 do not need to be represented as two different clusters and cluster 4
 722 and cluster 7 intersect due to the fact that they have a common value called
 723 *Major Event* of the attribute *type of event*. According to our observation,
 724 the separation of clusters (if applicable) is more effective than clusters with
 725 intersection over common values such as *Major Events*.

726 Some clusters are misclassified and not representable, according to our ob-
 727 servation. In Figure 6 some clusters such as *strike*, *sports meeting* etc. do not
 728 represent meaningful result. On the other hand, in Figure 7 clusters are more
 729 representative in terms of meaningfulness. For instance, an interpretation of
 730 data over the *type of event* attribute will produce clusters containing similar
 731 records. Similarly, an interpretation of data over the *public event type* and
 732 *network management type* attributes will produce groups separating public
 733 events (e.g. Activities) from network management events (e.g. narrow lanes,
 734 road closed). Such simple and comprehensive visualization will make anal-

735 ysis readable for the experts. In addition, automated systems can reap the
736 benefits from categorizing data before applying analytics.

737 *Discussion*

738 In this work we considered two quality attributes regarding our solution:
739 *performance* of the system and *accuracy* of our results that is the number
740 of clusters produced by our algorithms. Our observation reveals that the
741 clustering process in RePLoD (which we proposed in this paper for realtime
742 clustering) consume a significant amount of time which was essentially un-
743 expected. We found that the time consumption increases due to the need of
744 continuous listening to new data being fetched. Another reason was the need
745 for applying the clustering process all over again each time a new record is
746 fetched in order to deal with the evolution characteristic of clusters in real
747 time.

748 Furthermore, the clusters that were produced during our experiments
749 both in RePLoD and ProLoD are better than WEKA's hierarchical clustering
750 algorithm. The results obtained by WEKA were a set of seven clusters
751 placing common records (e.g. cluster 1 and cluster 2) without any meaningful
752 insight; nevertheless, the records corresponding to *Road Closed* are found in
753 three different clusters. On the other hand, our clustering results showed
754 that activities events (major event, sports meeting, and strike) were clustered
755 together, network management events (road closed and narrow lanes) created
756 a cluster, and traffic events (queueing traffic) created a cluster. Therefore,
757 predictive analytics can be applied to such self-explanatory clusters instead
758 of data points.

759 **8. Conclusion**

760 In this paper, we presented a framework called IBRIDIA for processing
761 logistics data. These data are stemming from various internal and exter-
762 nal sources within the logistics domain. The proposed framework integrates
763 several data preparation and processing functions. The output of such func-
764 tions is a clustered dataset. Afterward, the analytical engine exploits this
765 dataset to perform predictive analytics. In order to develop our data pro-
766 cessing model, we studied different machine learning algorithms; eventually,
767 we choose Johnson's hierarchical clustering algorithm. However, we mod-
768 ified the algorithm to become a stream clustering algorithm that supports

769 incremental grouping of text messages according to their similar characteris-
770 tics directly on the go. We presented the solution architecture of IBRIDIA
771 which consists of components for performing data preparation tasks such as
772 collection, filtering, cleaning, integration, and data storage. It also includes
773 components for processing data and applying some clustering techniques to
774 get self-explanatory groups of data supports predictive analytics for forecast-
775 ing delivery delay.

776 Furthermore, we studied various tools to implement our algorithm. We
777 implemented IBRIDIA by adopting both real-time and batch style computa-
778 tional models (ProLoD and RePLoD). Finally, we evaluated our solution with
779 the most widely used machine learning library called WEKA. We reported
780 and discussed the evaluation results. According to our observation, IB-
781 RIDIA's hierarchical clustering produced better representative results. How-
782 ever, the real-time version's performance of ProLoD was worse than the batch
783 style - which was due to the centralized environment of experiment consisting
784 of only one node. Thus, we presented both ProLoD as well as our frame-
785 work RePLoD for real-time collection and processing of logistics data that
786 are stemming from sources such as social media. We tested the framework
787 over a simple use case. Several works are planned for future work.

788 We plan to introduce time window to the algorithm so that we do not
789 have to store the data more than a specific time limit in order to be processed;
790 this effervescent data store is more suitable for realtime architectures. Using
791 weighted distances other than binary hamming distance with the aim of rep-
792 resenting important data point more significantly. Also, we plan to introduce
793 the ability to deal with numerical data as categories because in this article we
794 focused on text data whereas in the big picture numerical data is essential.
795 RePLoD needs to be tested over a real-world use case which will be done
796 in near future. RePLoD will be extended in two aspects: developing a pre-
797 dictive analytics and a prescriptive analytics model to suggest an optimized
798 route plan in order to prevent delay.

799 9. References

- 800 [1] G. Braun, IoT in the Supply Chain - Inbound Logistics, <http://www.inboundlogistics.com/cms/article/iot-in-the-supply-chain/>,
801 2016. [Online; accessed 04-January-2018].
802
- 803 [2] J. Macaulay, M. Kückelhaus, Internet of Things in Logis-

- 804 tics, [https://delivering-tomorrow.de/wp-content/uploads/2015/](https://delivering-tomorrow.de/wp-content/uploads/2015/08/DHLTrendReport_Internet_of_things.pdf)
805 [08/DHLTrendReport_Internet_of_things.pdf](https://delivering-tomorrow.de/wp-content/uploads/2015/08/DHLTrendReport_Internet_of_things.pdf), 2015.
- 806 [3] M. AlShaer, Y. Taher, R. Haque, M.-S. Hacid, M. Dbouk, Prolog: An
807 efficient framework for processing logistics data, in: OTM Confederated
808 International Conferences” On the Move to Meaningful Internet
809 Systems”, Springer, pp. 698–715.
- 810 [4] I. Taxidou, P. Fischer, Realtime analysis of information diffusion in
811 social media, Proc. VLDB Endow. 6 (2013) 1416–1421.
- 812 [5] G. Nieva, Integrating heterogeneous data, 2016.
- 813 [6] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud
814 computing and internet of things: a survey, Future Generation Com-
815 puter Systems 56 (2016) 684–700.
- 816 [7] E. Mezghani, E. Exposito, K. Drira, M. Da Silveira, C. Pruski, A
817 semantic big data platform for integrating heterogeneous wearable data
818 in healthcare, Journal of medical systems 39 (2015) 185.
- 819 [8] R. Power, B. Robinson, D. Ratcliffe, Finding fires with twitter, in:
820 Australasian language technology association workshop, volume 80.
- 821 [9] M. Heutger, M. Kückelhaus, K. Zeiler, D. Niezgoda, G. Chung, Self-
822 driving vehicles in logistics, 2014.
- 823 [10] LogisticsPlus, Top Logistics Challenges Facing
824 Shippers Today., [http://www.logisticsplus.net/](http://www.logisticsplus.net/top-logistics-challenges-facingshippers-today/)
825 [top-logistics-challenges-facingshippers-today/](http://www.logisticsplus.net/top-logistics-challenges-facingshippers-today/), 2015. [Online;
826 accessed 30-March-2016].
- 827 [11] D. Morales, Logistics & Transportation Executives Facing To-
828 days Challenges, Seek Solutions Well into the Future., [http://](http://www.stantonchase.com/logistics-transportion-executives-facing-todays-challenges-see-solutions-well-into-the-future/)
829 [www.stantonchase.com/logistics-transportion-executives-](http://www.stantonchase.com/logistics-transportion-executives-facing-todays-challenges-see-solutions-well-into-the-future/)
830 [facing-todays-challenges-see-solutions-well-into-the-future/](http://www.stantonchase.com/logistics-transportion-executives-facing-todays-challenges-see-solutions-well-into-the-future/),
831 2015. [Online; accessed 30-March-2017].
- 832 [12] F. Nĕmec, Distinguished problems of logistics, in: First International
833 Symposium on Business Administration, Challenges For Business Ad-
834 ministrators in the new Milleninium, pp. 1–3.

- 835 [13] J. Nwaubani, Business intelligence and logistics, in: Proceedings of
836 the 1st Olympus International Conference on Supply Chain, Katerini,
837 Greece.
- 838 [14] H. Chen, R. H. Chiang, V. C. Storey, Business intelligence and analytics:
839 From big data to big impact., MIS quarterly 36 (2012).
- 840 [15] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, W. M.
841 Van Der Aalst, The prom framework: A new era in process mining tool
842 support., in: ICATPN, volume 3536, Springer, pp. 444–454.
- 843 [16] X. L. Dong, D. Srivastava, Big data integration, in: Data Engineering
844 (ICDE), 2013 IEEE 29th International Conference on, IEEE, pp. 1245–
845 1248.
- 846 [17] Z. T. Birch, an efficient data clustering method for very large databases,
847 in: Proceedings of the 1996 ACM SIGMOD international conference on
848 Management of data (SIGMOD'96).-New York: ACM, pp. 103–114.
- 849 [18] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Cluster-
850 ing data streams: Theory and practice, IEEE transactions on knowledge
851 and data engineering 15 (2003) 515–528.
- 852 [19] S. Guha, N. Mishra, Clustering data streams, in: Data Stream Man-
853 agement, Springer, 2016, pp. 169–187.
- 854 [20] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan, Maintaining vari-
855 ance and k-medians over data stream windows, in: Proceedings of the
856 twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Prin-
857 ciples of database systems, ACM, pp. 234–243.
- 858 [21] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for cluster-
859 ing evolving data streams, in: Proceedings of the 29th international
860 conference on Very large data bases-Volume 29, VLDB Endowment, pp.
861 81–92.
- 862 [22] C. C. Aggarwal, J. Han, J. Wang, S. Y. Philip, On high dimensional
863 projected clustering of data streams, Data Mining and Knowledge Dis-
864 covery 10 (2005) 251–273.

- 865 [23] F. Cao, M. Estert, W. Qian, A. Zhou, Density-based clustering over
866 an evolving data stream with noise, in: Proceedings of the 2006 SIAM
867 international conference on data mining, SIAM, pp. 328–339.
- 868 [24] Y. Chen, L. Tu, Density-based clustering for real-time stream data,
869 in: Proceedings of the 13th ACM SIGKDD international conference on
870 Knowledge discovery and data mining, ACM, pp. 133–142.
- 871 [25] M. Khalilian, F. Z. Boroujeni, N. Mustapha, M. N. Sulaiman, K-means
872 divide and conquer clustering, in: Computer and Automation Engineering,
873 2009. ICCAE'09. International Conference on, IEEE, pp. 306–309.
- 874 [26] K. Udommanetanakit, T. Rakthanmanon, K. Waiyamai, E-stream:
875 Evolution-based technique for stream clustering, *Advanced Data Mining
876 and Applications* (2007) 605–615.
- 877 [27] C. Mahobiya, M. Kumar, Performance comparison of two streaming
878 data clustering algorithms (2014).
- 879 [28] T. W. Cheng, D. B. Goldgof, L. O. Hall, Fast fuzzy clustering, *Fuzzy
880 sets and systems* 93 (1998) 49–56.
- 881 [29] R. J. Hathaway, J. C. Bezdek, Extending fuzzy and probabilistic clus-
882 tering to very large data sets, *Computational Statistics & Data Analysis*
883 51 (2006) 215–234.
- 884 [30] R. L. Cannon, J. V. Dave, J. C. Bezdek, Efficient implementation of
885 the fuzzy c-means clustering algorithms, *IEEE transactions on pattern
886 analysis and machine intelligence* (1986) 248–255.
- 887 [31] C.-H. Wu, S.-J. Horng, Y.-W. Chen, W.-Y. Lee, Designing scalable
888 and efficient parallel clustering algorithms on arrays with reconfigurable
889 optical buses, *Image and Vision Computing* 18 (2000) 1033–1043.
- 890 [32] M. B. Al-Zoubi, A. Hudaib, B. Al-Shboul, A fast fuzzy clustering al-
891 gorithm, in: Proceedings of the 6th WSEAS Int. Conf. on Artificial
892 Intelligence, Knowledge Engineering and Data Bases, volume 3, pp. 28–
893 32.
- 894 [33] V. Vryniotis, DatumBox machine learning framework, [http://www.
895 datumbox.com/](http://www.datumbox.com/), 2013-2016. [Online; accessed 04-January-2018].

- 896 [34] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu,
897 J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning
898 in apache spark, *The Journal of Machine Learning Research* 17 (2016)
899 1235–1241.
- 900 [35] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu,
901 V. S. Tseng, Spmf: a java open-source pattern mining library, *The*
902 *Journal of Machine Learning Research* 15 (2014) 3389–3393.
- 903 [36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Wit-
904 ten, The weka data mining software: an update, *ACM SIGKDD explo-*
905 *rations newsletter* 11 (2009) 10–18.
- 906 [37] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online
907 analysis, *Journal of Machine Learning Research* 11 (2010) 1601–1604.
- 908 [38] M. A. Waller, S. E. Fawcett, Data science, predictive analytics, and big
909 data: a revolution that will transform supply chain design and manage-
910 ment, *Journal of Business Logistics* 34 (2013) 77–84.
- 911 [39] Z. Bi, D. Cochran, Big data analytics with applications, *Journal of*
912 *Management Analytics* 1 (2014) 249–265.
- 913 [40] Q. Gong, Y. Yang, S. Wang, Information and decision-making delays
914 in mrp, kanban, and conwip, *International Journal of Production Eco-*
915 *nomics* 156 (2014) 208–213.
- 916 [41] R. S. Michalski, J. G. Carbonell, T. M. Mitchell, *Machine learning:*
917 *An artificial intelligence approach*, Springer Science & Business Media,
918 2013.
- 919 [42] V. Estivill-Castro, Why so many clustering algorithms: a position paper,
920 *ACM SIGKDD explorations newsletter* 4 (2002) 65–75.
- 921 [43] K. J. Cios, W. Pedrycz, R. W. Swiniarski, Data mining and knowledge
922 discovery, in: *Data mining methods for knowledge discovery*, Springer,
923 1998, pp. 1–26.
- 924 [44] T. Galili, dendextend: an r package for visualizing, adjusting and com-
925 paring trees of hierarchical clustering, *Bioinformatics* 31 (2015) 3718–
926 3720.

- 927 [45] B. Everitt, S. Landau, M. Leese, Cluster analysis.: Arnold, a member
928 of the hodder headline group, 2001.
- 929 [46] R. L. Graham, P. Hell, On the history of the minimum spanning tree
930 problem, Annals of the History of Computing 7 (1985) 43–57.