Vela et al. EURASIP Journal on Wireless Communications and Networking (2018) 2018:82
https://doi.org/10.1186/s13638-018-1098-1

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

# Estimating the effect of network element events in a wireless network

Mario Vela[1*], Jeff Kraus[2], Steve Friedman[2], Michael Irizarry[3] and Prakash Suman[4]

## Abstract

Network events, like outages, are costly events for communication service providers (CSPs) not only because they represent lost revenue but also because of adverse effects suffered by the CSP's customers. Quantifying the effect of negative events on certain key performance indicators allows the CSP to measure the network resources impacted, to provide data for a more robust revenue assurance process, and to assign appropriate severity to the events. These additional insights may help optimize the resource allocation, ticketing, and troubleshooting response times. This paper presents a novel heuristic algorithm that takes advantage of the daily patterns observed in most key performance indicators of a wireless network and the stability observed in the differences between the original time series and the lagged version. The proposed algorithm uses those differences and the previous actual values to make accurate predictions of time-series traffic volume data that represent the estimated effect of a wireless network event. The performance of the algorithm is compared with that of the state-of-the-art autoregressive, integrated, moving average (ARIMA) model and the results are reported. The proposed algorithm has reduced standard deviation in error percentage by 4.8 percentage points, has no negative bias, and executes 97% faster than the ARIMA model. The algorithm provides an accurate methodology for online or batch network event impact estimation that could potentially be implemented in traditional relational database management systems (SQL) or Big Data environments.

**Keywords:** Network event, Forecasting, Anomaly detection, ARIMA, Time series, Self-Organizing Networks (SON)

## 1 Introduction

Understanding the effects of certain wireless network events (i.e., network service disruptions, network outages) allows communication service providers (CSPs) to drive initiatives that minimize the adverse effects of service impacting events on the overall experience of CSP customers. For instance, the estimated effect of service degrading events can be used to prioritize the repair efforts of operational teams when simultaneous events happen. Similarly, the same impact information can be used to determine the loss of wireless network usage during events, which can be categorized into severity levels that drive the urgency of repair efforts for the individual events.

The challenge of estimating the effect of wireless network events involves calculating the difference between (1) the expected system behavior under normal conditions and (2) the real behavior observed during the network event. A method is needed to estimate the typical values for the service during the time of the event for the quantification of the effect.

Our contribution consists of solving the problem of calculating the effect of a network event using an efficient and novel anomaly detection algorithm; we are calling this algorithm the "delta algorithm." The delta algorithm can be used not only to detect an anomalous event but also to accurately estimate the normal conditions for a particular set of service metrics. The network event effect is then calculated by comparing the algorithm-estimated expected values with the observed values during the event. The algorithm's predictive performance is analyzed and benchmarked against state-of-the-art autoregressive, integrated, moving average (ARIMA) time-series models in the R environment [1], and the algorithm is implemented as described in Section 2.4 with the results and implementation suggestions provided. The purpose of the paper is to demonstrate that the delta algorithm is a more accurate, unbiased, and a faster alternative than the state-of-the-art ARIMA

* Correspondence: Mario.Vela@uscellular.com
[1]US Cellular Corp., 800 Cornerstone Drive, Knoxville, TN 37932, USA
Full list of author information is available at the end of the article

Springer Open

© The Author(s). 2018 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

model when estimating the impact of a network event for a CSP.

### 1.1 Literature review

The 3rd Generation Partnership Project [2] supports the development of standards used for technologies and solutions deployed in CSP wireless networks. Additionally, this organization promotes a general framework for Self-Organizing Networks (SON) [3] which enables self-optimizing [4] and self-healing [5] functionalities. These functionalities can be initiated through trigger conditions [4, 5] based on alarms or detection of faults. However, 3GPP does not standardize the underlying methods or algorithms themselves. The delta algorithm may be one such solution for creating trigger conditions of anomalous events observed in network metrics that CSPs can leverage for SON solutions in their network.

The effect of a wireless network event can be estimated if the problem is treated as the need to quantify the effect of an anomalous event on a particular metric or set of metrics. In that regard, a network event scenario can be studied within the topic of anomaly detection applied to wireless networks. We follow this approach in the present paper.

Anomaly detection is a broad area of research that has applications in medicine [6], finance [7], computer networks [8, 9], and most recently the Internet of things [10–12] as well as several other business domains. According to a seminal work by Chandola et al. [13], "anomalies are patterns in data that do not conform to a well-defined notion of normal behavior."

Most of the anomaly detection methodology literature has focused on two broad areas of research [9, 13, 14]:

- Statistical analysis (Gaussian-based analysis, regression, correlation, statistical process control, intervention analysis)
- Machine learning (supervised learning, unsupervised learning, semi-supervised learning)

Additionally, the topic of scalability for an anomaly detection algorithm has been further discussed, and the limitations of some methods in extending to large numbers of metrics have been pointed out [9]. The complexity of the detection method is strongly correlated to the scalability properties of the method. The more complicated the methodology, the more challenging it will be to scale the methodology to many metrics. Current research efforts are focused on developing new paradigms of anomaly detection suitable for new large-scale distributed and cloud architectures as well as new computing models associated with them [14].

In particular, the proposed algorithm was developed in an attempt to address some of the above challenges by providing important efficiency benefits in comparison with traditional robust methods, such as ARIMA models, while providing improved accuracy performance. ARIMA models were originally introduced by Box and Jenkins [15] and had subsequently been adapted and implemented in various software environments. ARIMA modeling was implemented in the R statistical software [1] by Rob J. Hyndman using the forecast package [16].

ARIMA is one of many techniques available for the analysis of dependent observations known as time series data. It is a process in which identification, estimation, and diagnostic checking are performed on stationary data to fit a model and produce a forecast. A standard non-seasonal ARIMA ($p$,$d$,$q$) model is defined by three ordered parameters: an autoregressive component ($p$), which specifies the number of lags, an integrated component ($d$), which specifies the degree of differencing, and a moving average component ($q$) that represents the error and its possible lag components. This function can be written as the following linear equation:

$$Y_t = c + \Phi_1 y_{d^{t-1}} + \cdots + \Phi_p y_{d^{t-p}} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

However, to produce a suitable model, ARIMA components must be identified, which is an involved process and tends to be done manually on an as-needed basis. It is typically accomplished through differencing the time series data to achieve a stationary series and then tested for suitability. Next, visual inspection of autocorrelation plots known as autocorrelation function (ACF) and partial autocorrelation function (PACF) is performed. Although the forecast package [16] introduced by Rob J. Hyndman provides an "auto.arima" function that enables the automatic selection of the ARIMA model structure, there is increased computational overhead in allowing this function to cycle through numerous models to identify a suitable model to fit the time series data. The power of the proposed solution is that it leverages the existing time series data with minimal processing and improved prediction accuracy, and allows self-tuning of the model to remove the requirement to rediscover model parameters as new data is introduced to the model.

Additionally, although the present paper describes an offline application of the proposed methodology, the same algorithm could be used for an online anomaly detection system that could scale to a large number of metrics at lower time resolutions. This scenario is supported by the time responses reported later in the document that provides evidence that the algorithm could be executed against a large number of time series within a very short period of time. Similar efforts have been made using other methods leveraging ensemble methods and modified Holt–Winter models to address this similar use case [17, 18] (Table 1).

**Table 1** Related work

| Paper | Business domain | Related work |
|---|---|---|
| G Ciocarlie, U Lindqvist, K Nitz, S Novaczki, H Sanneck (2014) [17] | Cellular networks | An ensemble method is used to perform cell anomaly detection using Key Performance Indicators (KPI) typically used in 3GPP SON frameworks. |
| M Szmit, A Szmit (2012) [18] | Computer networks | Holt-Winters method is used to understand the traffic behavior in a computer network with the aim to perform intrusion detection. |

## 2 Methodology

The proposed algorithm allows the automatic detection of anomalous values in a time series. The algorithm takes advantage of the daily patterns observed in network datasets and attempts to model the difference or "delta" from one period to the next. Such deltas, characterized over a large historical dataset, can be used not only to predict the ranges of expected values for a future point in time (for anomaly detection purposes) but also to provide an estimate of future values of the data (for forecasting purposes). In this particular application, the estimation of the effect of a network event will involve using the algorithm's predicted values to facilitate the calculation of the effect by taking the difference between the real measurements and the predictions (i.e., the prediction error) as the most general case. In practical applications, outage events, for instance, usually reduce the actual network traffic to a value close to zero for the affected Key Performance Indicator (KPI), and each predicted value by the algorithm represents the estimated outage effect. On the basis of this scenario, the paper focuses on the prediction accuracy of the algorithm as the important benchmarking approach without losing applicability to the most general case.

### 2.1 Algorithm

As previously mentioned, the core objective of the algorithm is to characterize deltas and make a prediction for the next period given the most recent actual data point. This process is accomplished through two tasks: (1) one that calculates the expected "deltas" or changes from 1 hour to another using historical data points (Algorithm 1) and (2) another task that uses the pre-calculated "deltas" and the most recent data point to predict the next one (Algorithm 2).

**Algorithm 1** – Calculation of expected "Deltas" (for an individual KPI)

**IN:** a time-series of univariate KPI hourly data for three weeks = $X_t = \{x_1, x_2, x_3, \ldots, x_N\}$

**OUT:** 24 "Delta" points corresponding to the expected hourly change from one hour to the next within any day.

1: **for all** $x_t \in X_t$ **do**

2: $d_{t+1} \leftarrow x_{t+1} - x_t$ ▷ creates differenced series $D_t = \{d_2, d_3, d_4, \ldots, d_N\}$

3: **end for**

4: **for** i ← 1…24 **do**

5: $S_i \leftarrow \{d_{i+1+(24*0)}, d_{i+1+(24*1)}, d_{i+1+(24*2)}, \ldots\}$ ▷ creates hourly "delta" groups $S_i$

6: $\Delta S_i \leftarrow \text{MEDIAN}(S_i)$ ▷ creates expected "delta" for group $S_i$ (hour i)

7: **end for**

In general, a time series with $N$ elements is expressed as an ordered sequence:

$$X_t = \{x_1, x_2, x_3, \ldots\ , x_N\}$$

We generate a sequence of deltas according to

$$D_t = \{x_2 - x_1, x_3 - x_2, x_4 - x_3, \ldots\ , x_N - x_{N-1}\},$$

$$D_t = \{d_2, d_3, d_4, \ldots\ , d_N\},$$

where

$d_t = x_t - x_{t-1}$ $t = 2, 3, 4, \ldots . , N.$

The process provides a unique approach in that it uses $D_t$ rather than $X_t$ as the core input data for the modeling exercise. In other words, the algorithm attempts to model the changes or deltas in the data rather than the raw data themselves. This provides a more robust methodology given the fact that the changes from one time period to another in most network KPIs are predictable and slow.

Once $D_t$ has been calculated, the next step in the algorithm is to subdivide $D_t$ into 24 groups corresponding to the group of delta values for each corresponding hour across the multiple days included in our original dataset $X_t$:

$$S_i = \{d_{i+1+(24\times0)}, d_{i+1+(24\times1)}, d_{i+1+(24\times2)}, \ldots.\}, \quad i = 1 \ldots 24$$

$$S_1 = \{d_2, d_{26}, d_{50}, \ldots.\},$$

$$S_2 = \{d_3, d_{27}, d_{51}, \ldots.\},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$S_{24} = \{d_{25}, d_{49}, d_{73}, \ldots.\}.$$

To estimate the expected delta for each hour, the median value is taken from each group. The selection of the median is driven by the fact that the distribution of deltas observed during testing showed a consistent skewed unimodal distribution for which the central tendency is better captured by the median rather than other metrics such as the mean or mode. We have

$$\Delta S_1 = \text{median}\{d_2, d_{26}, d_{48}, \ldots.\},$$

$$\Delta S_2 = \text{median}\{d_3, d_{27}, d_{49}, \ldots.\},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\Delta S_{24} = \text{median}\{d_{25}, d_{49}, d_{73}, \ldots.\}.$$

The prediction of a future data point then becomes

$$\hat{x}_{t+1} = x_t + \Delta S_{h(x_t)}, \tag{1}$$

where $h(x_t) = 1 \ldots 24$ represents the corresponding hour of the day for $x_t$.

---

**Algorithm 2** – Predicting next hour value.

---

**IN:** most recent hour data point $x_t$; set of expected deltas $\Delta S_i$

**OUT:** prediction for next hour $\hat{x}_{t+1}$.

1: $\hat{x}_{t+1} \leftarrow x_t + \Delta S_{h(x_t)}$ ▷ $h(x_t) = 1 \ldots 24$ is a mapping function that returns the hour of the day corresponding for $x_t$

---

Note that once the set of expected deltas $\{\Delta S_1, \Delta S_2, \ldots, \Delta S_{24}\}$ has been calculated and persisted, any future value is easily predicted from the most current value and its corresponding delta. This efficient approach is particularly useful for a real-time large-scale implementation of the algorithm (online anomaly detection). A recursive application of (1) allows multiple-point forecasting for any time span:

$$\begin{gathered}\hat{x}_{t+1} = x_t + \Delta S_{h(x_t)}, \\ \hat{x}_{t+2} = \hat{x}_{t+1} + \Delta S_{h(\hat{x}_{t+1})} = x_t + \Delta S_{h(x_t)} + \Delta S_{h(\hat{x}_{t+1})}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \hat{x}_{t+n} = x_t + \sum_{k=0}^{n-1} \Delta S_{h(\hat{x}_{t+k})}\end{gathered} \tag{2}$$

Formulas (1) and (2) provide a scalable set of mathematical expressions that support a large-scale implementation

through a combination of batch and real-time online processes to provide an estimate of the event effect as soon as it happens and data become available.

### 2.2 Dataset description

The present study used a dataset comprising the hourly data volume (bytes) of a Long-Term Evolution (LTE) service for 4 weeks to test the algorithm (3 weeks of training and 1 week for testing). The selection of the time window for training the algorithm is configurable, but it should be chosen carefully to balance the user expectation of how much history is needed to get the first detection and how much accuracy is required. The more history used, the higher the probability of a better tuning process, but the first analysis will have to wait for more time. Our empirical results suggest that a window of 3 weeks provides sufficient data for a good balance between accuracy and prediction availability for practical applications. The data volume is a typical KPI for core network operations as defined by the 3rd Generation Partnership Project [2], which provides standards and specifications for CSPs and equipment manufacturers. Sample data were collected from 20 serving gateways (SGWs) of a major CSP (Fig. 1b). The eNodeBs attached to these 20 SGWs are distributed in 10% and 90% urban and rural morphologies, respectively (Table 2). The data were reviewed to ensure that no major events occurred during the selected time window and that they did not contain subsets of missing or zero values due to errors in data collection. The first 21 days of data were used to calculate the expected deltas $\{\Delta S_1, \Delta S_2, \ldots, \Delta S_{24}\}$, while the remaining data points (7 days) were used to validate the prediction. Figure 1a is a sample plot of the

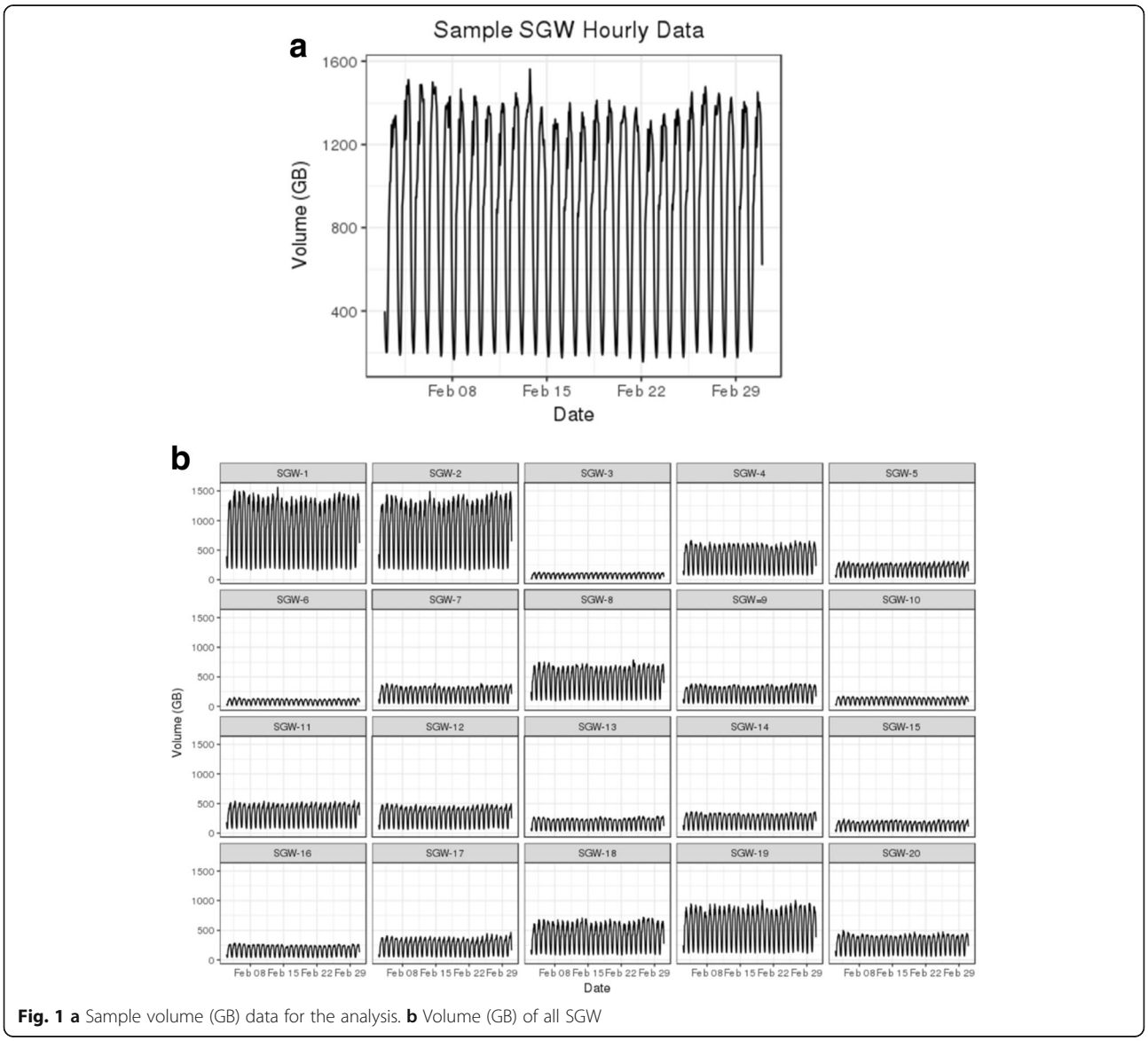


**Fig. 1** **a** Sample volume (GB) data for the analysis. **b** Volume (GB) of all SGW

**Table 2** Count of eNodeB per SGW

| SGW ID | Number of eNodeB |
|---|---|
| 1 | 419 |
| 2 | 458 |
| 3 | 296 |
| 4 | 423 |
| 5 | 433 |
| 6 | 243 |
| 7 | 204 |
| 8 | 272 |
| 9 | 791 |
| 10 | 172 |
| 11 | 534 |
| 12 | 545 |
| 13 | 395 |
| 14 | 315 |
| 15 | 219 |
| 16 | 338 |
| 17 | 447 |
| 18 | 234 |
| 19 | 340 |
| 20 | 546 |

raw data volume for one of the selected SGWs with a visible strong daily pattern.

### 2.3 Benchmarking methodology

Results from the delta algorithm outlined in Section 2 were benchmarked against the results of ARIMA models applied to the same dataset. ARIMA modeling is a sophisticated time-series modeling technique well suited to daily, weekly, and monthly patterns that exist in CSP network data. Hence, it is a common choice for time-series analysis requiring high levels of predictive accuracy. One drawback of ARIMA models is that they are computationally expensive to implement on databases, hence the need for algorithms that are more efficient.

Additionally, we compared the computing time taken by each approach to perform the modeling. If the two methods have similar accuracy, the decision on which one is more efficient to implement could be based on computational expense. This issue becomes more important as the number of KPIs increases or the time allowed for computation decreases (i.e., real-time application).

The following performance metrics were used in the benchmark process:

- Mean error (GB)
- Mean error (%)
- Median error (GB)
- Median error (%)
- Mean absolute error (GB)
- Mean absolute percentage error
- Mean runtime
- Median runtime

The error (GB) represents the residuals of the model defined as the difference between the predicted and actual volume in gigabytes. The error (%) represents the residuals of the model defined as the percent difference between the predicted and actual values observed. These two methods of measurement help visualize the performance of the model in both original units of the KPI scale as well as a relative percent scale. Additionally, since residuals tend to be a combination of positive and negative values due to prediction over and under estimations, the mean absolute error (GB) and mean absolute percentage errors are also considered. In this scenario, the absolute value of all residuals is used to produce all positive values before calculating the average error (GB) and percentage error. Lastly, the performance of runtime is examined as the average execution time needed for each forecast.

Each of the 20 time series corresponding to each of the SGW hourly data was processed using the delta algorithm and the automatic ARIMA forecasting routines available in the R “forecast” package. Special considerations were taken to remove the time consumed by the “forecast” package to search for the optimal model for a given time series. Hence, the analysis only considered the time that the selected ARIMA model used to tune its parameters. This detail was an important consideration needed to present a fair comparison of the time each algorithm took to create a forecast under the same hardware and software conditions.

Furthermore, to create the forecast data for benchmarking, 100 one-point forecasts were randomly calculated from the test set to estimate the predictive accuracy of each method for each of the SGWs. One-point forecasts are defined as a single forecast for a single hour data point from any of the hours in the test set. Of all possible 1-h time periods available in the test set, 100 random hourly periods were selected to perform forecasts. This selection was done with the intent of obtaining a representative sample with unbiased estimation and allows consistency with statistical simple random sampling techniques [19]. This process created 2000 random sample estimates (20 SGWs each with 100 one-point forecasts) available for each model. The performance metrics were applied to measure the relative accuracy and execution performance of the two methodologies. As mentioned before, prediction accuracy was in turn used as a proxy for successful event impact

estimation capability. In other words, the smaller the prediction error characteristics between the estimate and actual values, the more effective the method was for event impact estimations.

### 2.4 Hardware and software environment

A virtual environment instance was established running Redhat Linux (Kernel version 2.6.18-348.4.1.el5) with 16 cores (Intel Xeon 2.70GHz processors) and 16 GB of memory, and using the R Studio Server 0.99.447 [20], R base 3.2.3 [1], zoo 1.7.12 [21], ggplot2 1.0.1 [22], reshape 0.8.5 [23], and forecast 6.1 [16] packages to perform the analysis.

## 3 Results

Table 3 provides benchmark metrics for the delta algorithm and ARIMA models. The mean error percentage of the models were slightly different at − 0.21% and 0.53% for the ARIMA model and delta algorithm, respectively. However, the standard deviation indicates that the ARIMA model has a much larger spread of variation at 13.99% compared with the delta algorithm at 9.19%. The median values also suggest that (a) the delta algorithm is closer to zero error, indicating less bias in the results, and (b) the difference between the mean and median show the distributions are likely not normal distributions. The mean absolute percentage error provides a measurement that considers all model errors as positive with a higher error implying a less accurate model. This measurement suggests that the delta algorithm provides improved accuracy.

From a different point of view, the boxplot of errors depicted in Fig. 2 provides a non-parametric way of comparing the range of error between the two models. The plot clearly confirms the larger spread of variation for the ARIMA model evident by the larger range of outliers in comparison with the delta algorithm. Furthermore, the increased height of the box for the ARIMA model compared with that for the delta algorithm provides additional evidence of the larger variation observed for the ARIMA model. Lastly, the bold center line in the box represents the median or 50th percentile and shows that the ARIMA model falls slightly below the 0 GB level on the $y$-axis whereas the delta algorithm model appears to fall on the 0 GB level. This result provides additional evidence of the ARIMA model having a negative bias in its accuracy. In other words, the ARIMA model underestimates the real values observed for the test set.

Furthermore, the density distribution of error percentages in Fig. 3 depicts that the two models have a similar central tendency. However, the delta algorithm provides a slightly better prediction result as demonstrated by the reduced error bias and reduced error variability. This situation is evident by the kurtosis of the algorithm density plot showing more errors tightly grouped around the zero value whereas the ARIMA density plot slightly underestimates errors and has a lower peak indicating higher variability in the results that produces the fatter tails in the distribution.

Similarly, as part of the evaluation, it is important to assess if the error distributions are statistically different from each other to confirm if the observed differences in the benchmark metrics are real. Given the fact that neither distribution has normality properties, a non-parametric test is needed to determine whether the distributions are statistically different.

The Mann–Whitney independent sample test was used to make the above determination by performing a rank sum test against the errors of each model (delta algorithm versus ARIMA algorithm), resulting in a test metric $W = 2{,}270{,}619$ and $p$ value $\leq 0.0001$. This result provides strong evidence that the delta algorithm model and ARIMA model results are not from the same population (i.e., the difference between modeling results is real).

Additionally, a Wilcox test was performed to test whether the central tendency of the error percentage distributions was equal to zero. For the ARIMA model, the Wilcox test indicated a 95% confidence interval of [− 1.91%, − 1.10%] with $V = 820{,}010$ and $p$ value $\leq 0.0001$, providing strong evidence that the ARIMA model had a small negative error percentage bias and that the true error was not 0%. In contrast, the Wilcox test for the delta algorithm indicated a 95% confidence interval of [− 0.36%, 0.20%] with $V = 985{,}540$ and $p$ value = 0.5624, providing insufficient evidence to reject the null hypothesis that the true error percentage is 0%. This result implies that the delta algorithm has less error percentage bias and the true error percentage may be closer to 0%.

Applying the same analysis to the actual errors in volume (gigabytes) tells a similar story in favor of the delta algorithm. A Wilcox test indicates that the 95%

**Table 3** Error performance metrics—both algorithms

| | Error (%) | | | | Error (GB) | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Stdev. | Median | Median abs. | Mean | Stdev. | Median | Median abs. |
| ARIMA | − 0.205 | 13.987 | − 1.859 | 9.069 | − 2.445 | 39.260 | − 3.505 | 22.622 |
| Delta algorithm | 0.531 | 9.191 | − 0.123 | 5.944 | − 0.616 | 21.959 | − 0.309 | 13.226 |

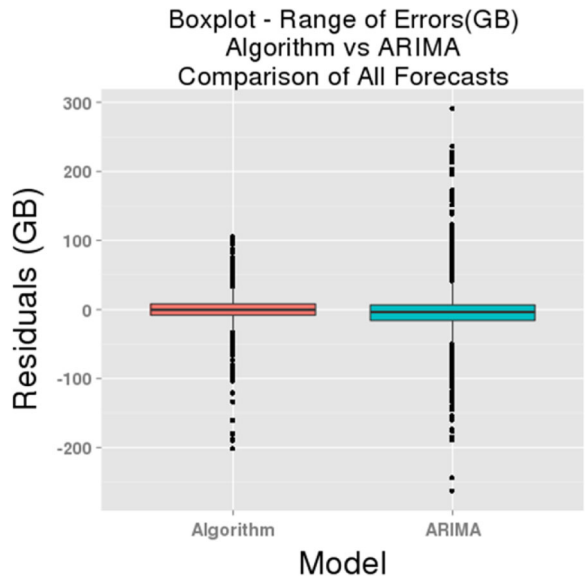


**Fig. 2** Boxplot of the range of errors (GB)

confidence interval is [− 4.82 GB, − 3.06 GB] for the ARIMA errors and [− 0.60 GB, 0.62 GB] for the delta algorithm. The ARIMA Wilcox test parameter was $V = 778{,}260$ and $p$ value ≤ 0.0001, providing substantial evidence that the true model error was not 0 GB. The Wilcox results for the delta algorithm were $V = 1{,}001{,}300$ and $p$ value = 0.9751, providing insufficient evidence to reject the null hypothesis that the true error is 0 GB. This result implies that the delta algorithm has less error bias and the true error may be closer to 0 GB.

The final benchmark used to compare the models was execution time. Table 4 depicts the execution time

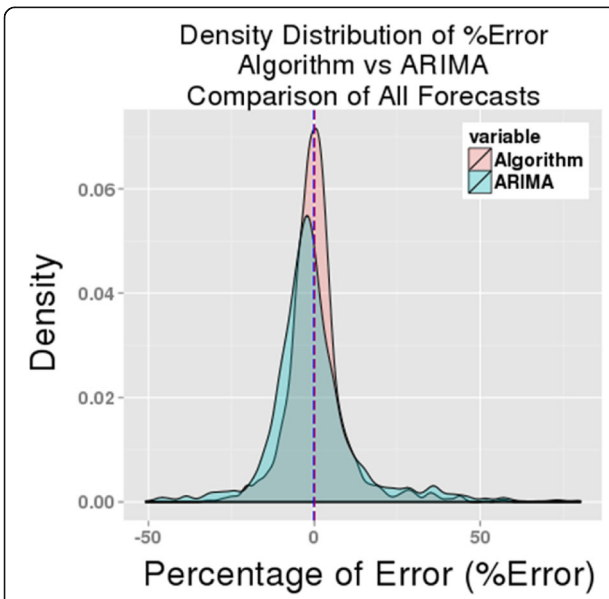


**Fig. 3** Density distribution of error percentages

**Table 4** Execution times in seconds—both algorithms

| | | | | auto.arima()[a] | | |
|---|---|---|---|---|---|---|
| | Mean | Stdev. | Median | Mean | Stdev. | Median |
| ARIMA | 0.482 | 0.330 | 0.381 | 420.542 | 97.182 | 399.357 |
| Delta algorithm | 0.017 | 0.002 | 0.016 | NA | NA | NA |

[a]Execution time for auto.arima() to complete 100-point forecasts

metrics in seconds elapsed for the forecasts to be completed by each method. Note that a second set of columns for the auto.arima() function from the "forecast" package is included to identify the performance metrics of using the automatic modeling capabilities of the R package that was used to identify the best ARIMA model for each forecast. These values represent the average (mean) time required to execute the 100-point forecasts for each of the 20 SGWs. From Table 4, on average, the delta algorithm execution time per point forecast is 0.47 s less or approximately 97% faster than the ARIMA algorithm execution time. Additionally, the delta algorithm model has a far narrower variation as evidenced by the reduced standard deviation of 0.002 s for the algorithm versus 0.33 s for the ARIMA algorithm. This reduced variation demonstrates the consistent and powerful nature of the proposed methodology. In summary, the delta algorithm provides not only superior error performance and overall prediction accuracy but also has important computational efficiencies that can be leveraged for a large number of metrics in a production system.

Finally, a proposed implementation scheme for large-scale real-time processing is described in Fig. 4. A batch process is charged with creating $D_t$ for each KPI using Algorithm 1. This particular dataset is small and could be stored in memory for fast random access. Additionally, an online or real-time process will analyze the incoming data and access the corresponding delta to create the forecast following Algorithm 2 and take the corresponding action depending on the use case.

## 4 Discussion

The delta algorithm provides superior results regarding improved accuracy, reduced variability, unbiased errors, reduced complexity, and fast computation. The solution is lightweight and can be implemented in existing SQL query language or similar code without the need for external software packages. Additionally, the solution is designed in a manner that is potentially highly scalable allowing adaptation to numerous applications of anomaly detection, such as outage estimation, outage impact, billing reconciliation, and record validation. The solution's measured performance relative to the performance of well-known ARIMA models provides evidence of the

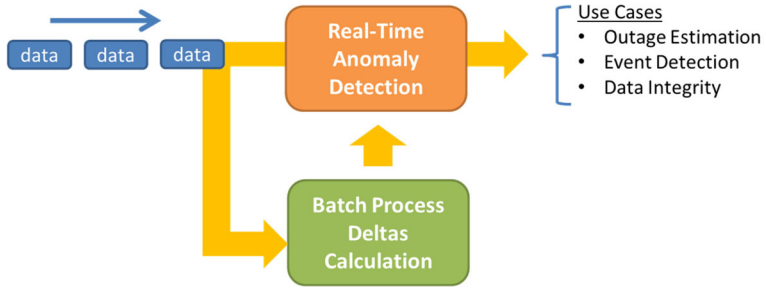


**Fig. 4** Simple implementation scheme for large-scale online application of the delta algorithm

solution's ability to provide high accuracy with minimal computing overhead.

Even though the average (mean) time required to execute the 100-point forecasts for each of the 20 SGWs was discounted for the benchmarking process, it is important to note that an implementation of the ARIMA methodology must incur such a delay when searching for the best model to execute the best forecast. An alternative methodology that prevents such delay would be to select a generic ARIMA model with a fixed structure to be applied to all datasets. However, that approach would sacrifice overall accuracy. This detail provides further evidence of the delta algorithm's ability to provide higher accuracy without external constraints and additional data processing.

The study has the following limitations:

- Only 1 month of data was used for both training (3 weeks) and testing (1 week).
- Data were collected at a highly aggregated network element (SGW) as opposed to a less aggregated network element, such as cell towers that would have more variable data.

There are additional research opportunities to adapt the current algorithm to data with higher levels of variability.

## 5 Conclusions

The proposed algorithm provides superior performance for the selected datasets than the robust and widely used ARIMA methodology in predicting time series data. It demonstrates less variation (measured by the standard deviation) than the ARIMA model by 4.8 percentage points, has no negative bias, and executes 97% faster than the ARIMA model. Additionally, its simplicity and consistency of performance make the algorithm suitable for deployments requiring fast response. The algorithm can be easily implemented on traditional databases or NoSQL databases to support a large number of time series. Additionally, the algorithm not only provides a way to calculate the effect of a network element event impact but could also be adapted to perform anomaly detection (i.e., measuring deviation from the estimated points) on the same datasets in an online fashion.

Further research might be needed to support lower time resolutions (i.e., 5-min data) and to investigate the performance effects when the algorithm is applied to datasets with reduced seasonal patterns and higher variability.

### Abbreviations

ARIMA: Autoregressive, integrated, moving average; CSP: Communication service provider; KPI: Key Performance Indicator; LTE: Long-Term Evolution; SGW: Serving gateway

### Acknowledgements

We thank US Cellular Corporation for providing the data for this study.

### Availability of data and materials

Datasets used in this paper are the property of US Cellular Corporation and not publicly available.

### Authors' contributions

MV conceived the presented idea and performed data interpretation and critical revision of the article. SF performed data collection, data analysis and interpretation, and critical revision of the article. JK performed design, data collection, data analysis and interpretation, and critical revision of the article. MV, SF, and JK drafted the article. MI and PS provided critical revision and final approval of the version to be published. All authors read and approved the final manuscript.

### Competing interests

The authors declare that they have no competing interests.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

### Author details

[1]US Cellular Corp., 800 Cornerstone Drive, Knoxville, TN 37932, USA. [2]US Cellular Corp., 50 Commerce Drive, Suite 130, Schaumburg, IL 60173, USA. [3]US Cellular Corp., 8410 W Bryn Mawr, Suite 700, Chicago, IL 60631, USA. [4]US Cellular Corp., 10 Corporate Drive, Suite 210, Bedford, NH 03110, USA.

Received: 5 September 2017 Accepted: 12 April 2018
Published online: 19 April 2018

### References

1. R Core Team. The R project for statistical computing. https://www.r-project.org/. Accessed 16 Apr 2018.

2. 3rd Generation Partnership Project (3GPP). Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN): definitions (Release 13). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2026. Accessed 16 Apr 2018.
3. 3rd Generation Partnership Project (3GPP). Technical specification group services and system aspects. Telecommunication management. Self-Organizing Networks (SON). Concepts and requirements (Release 14). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2031. Accessed 16 Apr 2018.
4. 3rd Generation Partnership Project (3GPP). Technical specification group services and system aspects. Telecommunication management. Self-Organizing Networks (SON) policy network resource. Model (NRM) Integration Reference Point (IRP). Requirements (Release 11). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2041. Accessed 16 Apr 2018.
5. 3rd Generation Partnership Project (3GPP). Technical specification group services and system sspects. Telecommunication management. Self-Organizing Networks (SON). Self-healing concepts and requirements (Release 14). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2052. Accessed 16 Apr 2018.
6. H Sivaraks, CA Ratanamahatana, Robust and accurate anomaly detection in ECG artifacts using time series motif discovery. Comput. Math. Methods Med., 1–20 (2015). https://doi.org/10.1155/2015/453214
7. K Golmohammadi, OR Zaiane, Time series contextual anomaly detection for detecting market manipulation in stock market. Paper presented at the IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19 2015.
8. Z Jadidi, V Muthukkumarasamy, E Sithirasenan, K Singh, Performance of flow-based anomaly detection in sampled traffic. J. Networks. **10**(9) (2015). https://www.researchgate.net/publication/289556497_Performance_of_Flow-based_Anomaly_Detection_in_Sampled_Traffic
9. H Huang, H Al-Azzawi, H Brani, Network traffic anomaly detection. CoRR, abs/1402.0856. (2014). http://arxiv.org/abs/1402.0856
10. Y Yuan, K Jia, A distributed anomaly detection method of operation energy consumption using smart meter data. Paper presented at the 2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Adelaide, Australia, 23–25 Sept. 2015.
11. DL Goodman, J Hofmeister, R Wagoner, Advanced diagnostics and anomaly detection for railroad safety applications: using a wireless, IoT-enabled measurement system. Paper presented at the IEEE AUTOTESTCON, National Harbor, Maryland, 2–5 November 2015.
12. DH Summerville, KM Zach, Y Chen, Ultra-lightweight deep packet anomaly detection for Internet of Things devices. Paper presented at the 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), Nanjing, China, 14–16 December 2015.
13. V Chandola, A Banerjee, V Kumar, Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 1–58 (2009). https://doi.org/10.1145/1541880.1541882
14. O Ibidunmoye, F Hernandez-Rodriguez, E Elmroth, Performance anomaly detection and bottleneck identification. ACM Comput. Surv. **48**(1), 4–35 (2015). https://doi.org/10.1145/2791120
15. GEP Box, GM Jenkins, Time series analysis: forecasting and control, Revised Edition, (Holden-Day, San Francisco, 1976).
16. R Hyndman, M O'Hara-Wild, C Bergmeir, S Razbash, E Wang, forecast: Forecasting functions for time series and linear models. https://cran.rstudio.com/web/packages/forecast/. Accessed 16 Apr 2018.
17. G Ciocarlie, U Lindqvist, K Nitz, S Novaczki, H Sanneck, in *IEEE Network Operations and Management Symposium (NOMS)*. On the feasibility of deploying cell anomaly detection in operational cellular networks (2014), pp. 1–6
18. M Szmit, A Szmit, Usage of modified Holt-Winters method in the anomaly detection of network traffic: case studies. J. Computer Networks Commun **2012** (2012). https://doi.org/10.1155/2012/192913.
19. Steven K. Thompson, Sampling, Third Edition, (Wiley,Wiley Series in Probability and Statistics, 2012)
20. R Studio Core Team. R studio server download. https://www.rstudio.com/products/rstudio/download-server-2/. Accessed 16 Apr 2018.
21. A Zeileis, G Grothendieck, JA Ryan, JM Ulrich, F Andrews, zoo: S3 infrastructure for regular and irregular time series (Z's ordered observations). https://cran.rstudio.com/web/packages/zoo/. Accessed 16 Apr 2018.
22. H Wickham, W Chang, ggplot2: an implementation of the grammar of graphics. https://cran.rstudio.com/web/packages/ggplot2/. Accessed 16 Apr 2018.
23. H Wickham, reshape: flexibly reshape data. https://cran.rstudio.com/web/packages/reshape/. Accessed 16 Apr 2018.



**Submit your manuscript to a SpringerOpen journal and benefit from:**

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

**Submit your next manuscript at ► springeropen.com**