(Online) ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 14, No. 2, April 2026, pp.43-64

Integrated density and buffer-based stream data clustering for detecting anomalies in the internet of moving things

Fuqdan A. Al-Ibraheem[1], Farhad Mardukhi[2*]
[1] Department of Computer Engineering and Information Technology, Razi University, Kermanshah, Iran
[2]Assistant Professor, Department of Computer Engineering and Information Technology, Razi University, Kermanshah, Iran

*Corresponding author E-mail: Mardukhi@razi.ac.ir

ABSTRACT

Detecting anomalies in roadway environments is a vital application of intelligent transportation systems (ITS). It involves setting up sufficient sensing systems, reading and analyzing generated stream data, and clustering the data. Typically, there are numerous sensors that are movable, which suggests high dimensionality of data with high dynamics. Clustering high-dimensional stream data with relatively high dynamics is not an easy task. Existing stream data clustering creates stages of buffering before enabling the production of legitimate clusters. However, none of the existing clustering methods are adapted to operate effectively for moving vehicles. In this study, we incorporate an offline phase for creating clusters based on the density of core mini-clusters in an existing buffer-based online clustering for evolving data streams (BOCEDAS). We designate it as multi-density data stream (MUDEDS) clustering. The goal is to detect and cluster anomalies in the roadway environment. In addition, we build simulations for various types of anomalies in the roadway environment. Experimental results demonstrate the superiority of MUDEDS when evaluated on passing traffic signal, accident, and slippage types of anomalies compared to benchmarking clustering algorithms.

Keywords: IoMT, MUDEDS, BOCEDS, Anomaly Detection, Clustering Algorithms, Roadway Environment

1. Introduction

The ongoing advancement of Internet of Things (IoT) technology has been a significant catalyst for societal progress in recent times [1]. Generally speaking, IoT incorporates wireless networking, an array of sensors, and technological components into objects that were traditionally not considered ‘smart,’ thereby contributing to the realization of ‘smart cities’ [2]. The Internet of Moving Things (IoMT) adds to this idea through allowing the connection of mobile elements such as vehicles; cell phones; robots; and all other kinds of portable units. IoMT creates a network consisting of people, autos, buses, semi-trucks, railroad locomotives; wearable technology; and laptop computers [3]. When these are connected via Wi-Fi or mobile communication networks, they can send and receive information. As portable items are now so integral to our lives today it was a matter of time before portable items were integrated into interconnected systems. Nevertheless, mobile objects are expected to emerge as crucial elements within the IoT ecosystem: those that move autonomously, such as cars and robots [4], and those that accompany us, like mobile devices and wearables [5].

Given the essential functions that mobile entities perform in today’s world, like the transportation of individuals and goods, it is only fitting that vehicles such as cars, trucks, and trains would be networked, for instance, for battery management purposes [6]. Both corporate entities and individuals are keen to gain insights into location, fuel efficiency, and inter-vehicle relationships. The ability to collect data across diverse settings offers a more comprehensive understanding of our lives, thereby aiding companies in the development of more user-centric products. For example, insurance companies can monitor driving behaviors in real-time to offer discounts to responsible drivers while potentially increasing premiums for those who exhibit risky behavior [7]. In military

scenarios, commanding units can remotely track their soldiers, enabling better coordination and decision-making to enhance their safety [8].

In a typical IoMT ecosystem, sensors on moving objects ranging from autonomous vehicles to drones and pedestrians collect a myriad of data, which is then transmitted to a centralized server over the network. At this juncture, specialized algorithms analyze the incoming data to detect any anomalies. Irregularities may indicate numerous problems, including failures of systems, security breaches, or abnormal activity. In addition, anomalies in patterns of motion may include unexplained changes in speed, direction, or altitude indicating possible dangers or unauthorized activities. Therefore, by incorporating anomaly detection within the IoMT framework; organizations will be able to proactively address risks resulting from both static and dynamic elements. For example, this would include not only reliability and security aspects of their mobile equipment but also how those devices move. Detecting anomalies in movement patterns adds an additional level of complexity to the IoMT system thereby enabling it to provide enhanced support for real-time monitoring and decision making. As shown in Figure 1, the combination of IoMT and anomaly detection technologies serve as an effective tool to protect operational efficiency and ensure reliability of equipment.

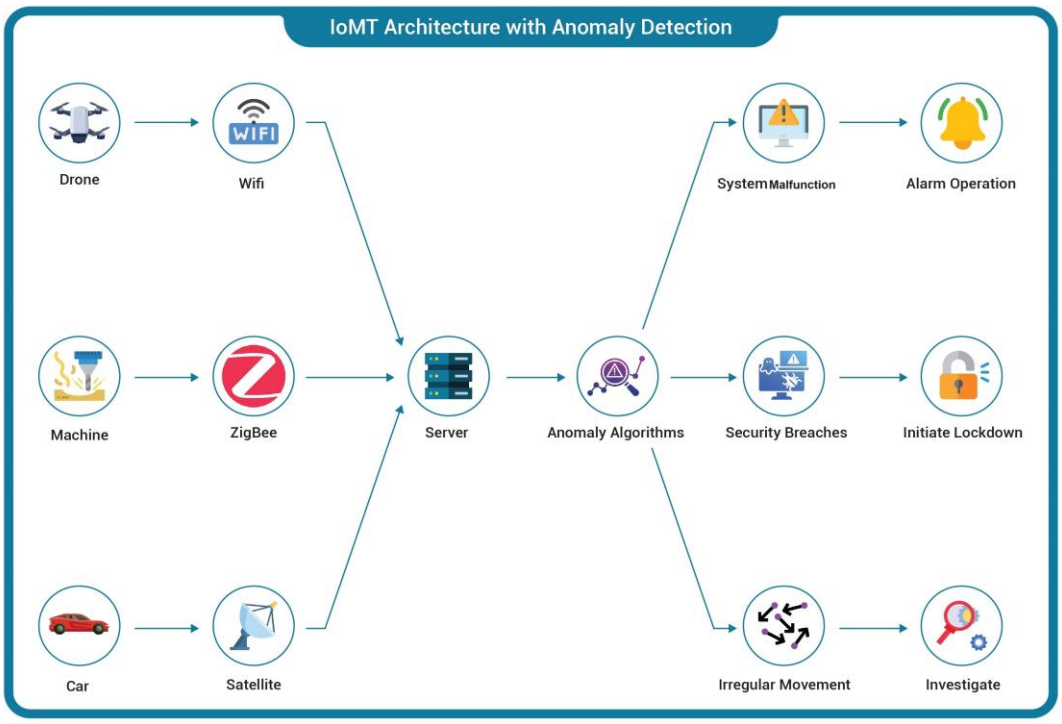

Figure 1 Conceptual diagram of internet of moving things architecture with anomaly detection

Section 2 of this article provides a detailed review of current anomaly detection methodologies used in IoT and IoMT environments. Included in the literature review is a list of established benchmark algorithms as well as an identification of areas where further research is needed. Section 3 outlines the proposed methodology utilized for detecting anomalies in the IoMT environment. This section describes the mathematical notation and problem formulation used in developing the proposed methodology. Additionally, the mathematical model developed for determining the parameters necessary to identify anomalies in the IoMT environment are outlined. Furthermore, the system architecture employed utilizing the proposed methodology is explained and all of the algorithms included in the methodology are described. Section 4 provides results from experiments conducted on several benchmark algorithms for comparison purposes. Results obtained from these comparisons are based upon multiple performance criteria. Lastly, Section 5 summarizes the articles findings and identifies areas where further research is warranted. This paper develops a framework for real-time anomaly detection in the Internet of Moving Things (IoMT).

Contributions include:

- MUDEDS Algorithm: A novel stream data clustering method that uses multiple phases to process data at high velocities with reduced computational overhead while preserving data integrity. It employs periodic density-based clustering to find complex shapes in clusters, and remove noise better than previous methods. It also handles dynamic IoMT data more effectively than do traditional methods.
- Dual Phase Hybrid Processing Framework: A hybrid processing framework consisting of two phases: continuous real-time processing phase to immediately identify patterns, and periodic offline clustering phase to find emerging/complex patterns over time. The dual phase processing allows for a trade-off between timely response to anomalies and deep analysis of evolving data distribution.
- Realistic IoMT Anomaly Generation Environment: An extensive simulation environment that generates synthetic GPS & IMU data streams simulating typical noise and variability found in real world GPS and IMU sensor data. It also creates three types of common road anomalies found in real-world scenarios (traffic violation, accident, slipping) allowing for testing of the proposed methodology in a controlled setting.

2. Literature survey

The literature includes numerous recent studies on the problem of anomaly detection in the Internet of Things (IoT) in general and in the Internet of Mobile Things in particular.

In the work of Slavic et al. [9], a Variational Autoencoder (VAE) is used for reducing the dimensionality of video frames, generating latent space information that is comparable to low-dimensional sensory data (e.g., positioning, steering angle), making feasible the development of a consistent multi-modal architecture for autonomous vehicles. In addition, an Adapted Markov Jump Particle Filter defined by discrete and continuous inference levels is employed to predict the following frames and detect anomalies in new video sequences. The algorithm is restricted to anomalies generated from video input, including detection of emergency stops, pedestrians' avoidance, and U-turn maneuvers. Li and Zou [10] proposed an automated data engineering pipeline for anomaly detection of IoT sensor data. The process involves the use of IoT sensors, Raspberry Pi, Amazon Web Service (AWS) and multiple machine learning techniques with the intent to identify anomalous cases for the smart home security system.

Tian et al. [3] proposed MTOD (Moving Things Outlier Detection), a generalized approach for anomaly detection from the Internet of Moving Things. The distance of moving things was proposed, which is equal to the weighted sum of the location distance and the multi-sensor distance, and then the multi-sensor data generalization and moving things partitioning and anomaly detection three-step framework to detect the generalized anomaly. Kang et al. [11] proposed a similarity metric Convolutional Neural Network (CNN) based on a channel attention model for traffic anomaly detection task. The method mainly includes: (1) A Siamese network with a hierarchical attention model by word embedding so that it can selectively measure similarities between anomalies and the templates; (2) A deep transfer learning method can automatically annotate an unlabeled set while fine-tuning the network; (3) A background modeling method combining spatial and temporal information for anomaly extraction.

Chung and Kim [12] established the relationship between the sensation of sleep deprivation among drivers during long journeys and CO2 concentrations in vehicles. Multimodal signals are collected using five sensors that measure the levels of CO, CO2, and particulate matter (PM), as well as temperature and humidity. A deep network employing LSTM, skip-GAN, and VAE models builds an air quality anomaly detection model providing real-time PM alerts and sleep-deprived driving alerts. Jeong et al. [13] proposed two approaches including outlier detection and shipping-route construction using automatic identification system (AIS) data, comparing against the Mahalanobis distance method for trajectory-outlier detection. Wagan et al. [14] developed Duo-Secure IoMT framework that uses multi-modal sensory signals' data to differentiate attack patterns from routine IoMT devices' data, combining dynamic Fuzzy C-Means clustering with customized Bi-

LSTM technique. Mittal et al. [5] proposed deep learning-powered anomaly detection for IoT that learns and captures robust and useful features not significantly affected by unstable environments, using denoising autoencoder adapted to obtain features robust against heterogeneous IoT environments. Mohamed et al. (Mohamed et al. 2022) proposed Kalman filter-based model for estimating pedestrian trajectory within indoor environments by fusing WiFi with IMU data, with trustworthiness assessment for detecting anomaly behavior using online sequential extreme learning machine. Liu et al. [15] proposed outlier detection based on isolated forest with compression processing, accurately identifying anomalies caused by gradual and sudden changes. Yu et al. [16] proposed unsupervised contextual anomaly detection through wireless sensor networks accounting for dynamic anomaly status and correlations based on spatial and temporal neighbors. Alanazi and Aljuhani [17] proposed machine learning-based intrusion detection for industrial IoT achieving high accuracy rates.

Recent advancements include Basheer et al. [18]'s non-parametric algorithm for autonomous anomaly detection in streaming data employing three stages: identifying potentially anomalous samples based on data density, clustering online using evolving autonomous data partitioning algorithm, and pinpointing true anomalies from minor clusters with least associated samples. [19] presented DWAD-LDVP (Dynamic Window Anomaly Detection based on Local Density of Vector dot Product) using adaptive dynamic sliding windows with verification model, converting local density values into outlier scores with incremental computation for efficient processing.

We compare our proposal against three other highly relevant and well-known stream clustering methodologies. The first is CEDAS [20], which was the first completely online method for clustering arbitrarily shaped evolving data streams. It utilized a two-stage methodology; one stage utilized hyper-spherical micro-clusters and another stage utilized an entirely new type of graph structure, to show how real-time adaptations could be made to the shape of evolving clusters. In addition to being able to adapt to evolving cluster structures in real-time, CEDAS demonstrated superior ability compared to all previous methodologies. The second is BOCEDS [21], which improved upon the CEDAS methodology through the introduction of an additional buffer mechanism for temporarily irrelevant micro-clusters. Additionally, BOCEDS provided a means to recursively update the radii of each micro-cluster so that they represented locally optimal values. This addressed a major limitation of all previous methodologies (including those utilizing pre-defined radius parameters) related to the determination of the appropriate radii for the micro-clusters. Finally, the third methodology we utilize as a comparison is MuDi-Stream [22]. MuDi-Stream differs significantly from both CEDAS and BOCEDS since it utilizes a hybrid online/offline methodology that is particularly suited for clusters of various densities. During the online phase, MuDi-Stream uses core mini-clusters as representative samples of the clusters present in the data stream. Once enough data points have accumulated within each cluster, the off-line phase is initiated where a density based clustering algorithm (adapted from DBSCAN) is applied. A grid-based outlier buffer provides for effective separation between true outliers and clusters of varying density.

Our literature review identified four main shortcomings associated with existing methodologies for stream data clustering and anomaly detection in IoMT environments:

- Limited capacity for adapting to changes in the underlying distribution of data: Many existing methodologies are not well-suited to the rapid evolution of data patterns inherent in many IoMT applications. Most methodologies either sacrifice the ability to identify complex temporal relationships between data points to facilitate faster real-time processing, or sacrifice immediate response capabilities to enable efficient off-line analysis.
- Poorly optimized efficiency for high-velocity data streams: Many existing methodologies are inefficient due to excessive computational overhead and/or large memory requirements resulting from handling the high-volume/velocity characteristics of IoMT data streams. As such, our multiple stage buffering strategy in MUDEDS enables efficient management of high-velocity data streams by minimizing the computational resources required to process them while still ensuring accuracy.

- Lack of adequate evaluation methodologies specific to IoMT environments: Much of the existing research employs generic test cases/datasets or unrealistic simulated environments that do not adequately represent the complexities of actual IoMT deployments. To fill this gap, we implemented a comprehensive simulation environment suitable for testing IoMT-related methodologies.
- Neglecting the implications of data preprocessing strategies: Many researchers have employed standard data preprocessing strategies without adequately evaluating their potential impacts on clustering effectiveness in IoMT contexts. Specifically, we discovered that Kalman Filtering which is widely accepted as a best-practice technique for preprocessing time-series sensor readings may actually impede clustering performance in certain types of IoMT contexts, thereby challenging prevailing assumptions about what constitutes "best practices."
- Underutilization of density-based methodologies in streaming environments: Density-based methodologies have shown great promise for identifying complex distributions in static/non-streaming contexts. However, their application to high-velocity IoMT streams has been largely under-explored. As part of MUDEDS, we include a density-based off-line phase for discovering complex shapes and removing noise from clusters more effectively than traditional stream clustering methodologies.

3. Methodology

This part of the chapter will introduce a new theoretical concept of a proposed framework called MUDEDS as an Anomaly Detection Framework in Internet-of-Medical-Things (IoMT) Environment. First, we describe the mathematical notation and some important terms that are going to be used in this dissertation. Then we formulate the problem and provide the mathematical formulation of our approach. Afterward we describe the System Architecture for our solution. Following that, we detail two Algorithms for Anomaly Simulation and MUDEDS Clustering Method. At last, we specify the Metrics of Evaluation which will be utilized to evaluate the Performance of each Algorithm.

3.1 Mathematical notations

Also, Table 1 shows below listing the most common mathematical notations and Parameters used all over this paper.

Table 1 Mathematical notations and parameters summary

Category	Symbol	Definition	Context
Problem Form.	E	Urban environment	System scope
	V	Set of vehicles	Vehicle population
	$P(U_i, t)$	Position data from GPS at time t	Spatial coordinates
	A_x, A_y, A_z	Acceleration data from IMU	Motion dynamics
	$\hat{T}(U_i, t)$	Estimated trajectory	Kalman filter output
Stream Clustering	$MC_{potential}$	Set of potential micro-clusters	Clustering structure
	C	Micro-cluster center	Spatial centroid
	R	Micro-cluster radius	Spatial extent
	N_t	Number of data points at time t	Cluster density
	E_t	Energy level at time t	Temporal relevance
	$d(X_i, C)$	Distance between point and center	Proximity measure

Category	Symbol	Definition	Context
Micro-cluster Types	CMCs	Core micro-clusters	High-density clusters
	PMCs	Potential micro-clusters	Medium-density clusters
	WMCs	Weak micro-clusters	Low-density clusters
	$Th_{density}$	Density threshold	Core classification
	BWMC	Buffer for weak micro-clusters	Temporary storage
Kalman Filter	X_t	State vector	System state
	A	State transition matrix	Process model
	z_t	Measurement vector	Sensor observations
	H	Measurement matrix	Observation model
Parameters	Decay	Energy decay factor (50)	Temporal forgetting
	CMThresh	Core MC threshold (10)	Density requirement
	R_{max}/R_{min}	Max/min radius (0.1/0.05)	Spatial constraint
	minPts	Minimum points (3)	Density parameter
	Horizon	Offline interval (2)	Temporal window

3.2 Definitions

The following are basic definitions for understanding the major ideas behind MUDEDS algorithm.

Minimum Density Threshold: This is the minimum amount of data points required to be in a micro cluster's area R to qualify it as a CMC (core micro-cluster).

Temporary Storage System Buffer Mechanism For Weak Micro Clusters (BWMC): Temporary buffer mechanism where weak clusters with low enough density, which may still be important at some point will be stored:

$$BWMC = \{MC_i: |N_i| < Th_{density} \wedge E_i > 0\} \quad (1)$$

The energy level of cluster i is denoted by E_i and clusters are deleted once their total energy falls below zero.

Clustering window, $\Delta T_{horizon}$, the time period for which offline clustering takes place.

$$\Delta T_{horizon} = T_{offline}^{(k+1)} - T_{offline}^{(k)} = Horizon \quad (2)$$

where $T_{offline}^{(k)}$ represents the k-th offline clustering execution time.

3.3 Problem Formulation

Anomalies can be detected in urban environments E using the set of vehicles V that consist of vehicle U_i that have installed GPS and Inertial Measurement Unit (IMU) sensor equipment. At a given point in time t, these vehicle's location data $P(U_i, t)$ is provided by their GPS unit and acceleration data $A_x(U_i, t), A_y(U_i, t)$, $A_z(U_i, t)$ are measured by the IMU units on board. Acceleration data is utilized to create trajectory estimates of the path traveled $\hat{T}(U_i, t)$ for each vehicle U_i. Anomaly detection algorithms identify a subset of specific anomalies A:

- $AT(U_i, t)$ - Anomalous trajectories
- $PAM(U_i, t)$ - Pedestrian avoidance maneuvers
- $TLV(U_i, t)$ - Traffic light violations

- $RS(U_i, t)$ - Road slippage
- $AC(U_i, t)$ - Accidents

Thus,

$$A = \{AT, PAM, TLV, RS, AC\} \quad (3)$$

3.4 Mathematical Model

The proposed stream clustering algorithm operates with the following update rules:

$$MC_{potential} = MC_{potential} \cup MC_{new} \quad (4)$$

$$d(X_i, C) < R$$

$$N = N + 1 \quad (5)$$

$$R_{t+1} = \min\left(\sqrt{R_t^2 + \frac{\sum d(X_i, C_{t+1})}{N} - 1} \times Decay, R_{max}\right) \quad (6)$$

$$C_{t+1} = \frac{(N_{t+1}-1)\times C_t + X_{t+1}}{N_{t+1}} \quad (7)$$

$$E_{t+1} = E_t + \frac{|d(X_i, C_{t+1}) - R_t|}{R_t} \times \frac{1}{Decay} \quad (8)$$

$$d' = \begin{cases} R + \frac{r}{2} & \text{if } R \geq r \\ r + \frac{R}{2} & \text{if } R < r \end{cases} \quad (9)$$

3.5 System architecture

In Figure 2, we present an architectural overview of the system. We have identified five blocks which interact together to provide a full solution: (1) The Track Information and Environment Specifications are inputted into the IoMT Simulator where they simulate accidents and slips that may occur on the road as well as traffic flow; (2) The Sensor Data Generation block uses GPS, IMU along with their corresponding noise values to generate sensor data; (3) The Kalman Filter block combines the data from both sensors and generates estimated trajectory data; (4) The Feature Extraction Block will then extract features from this estimated trajectory data generated by the Kalman filter; and finally (5) Anomalies in the trajectory data can be detected using the clustering algorithm.

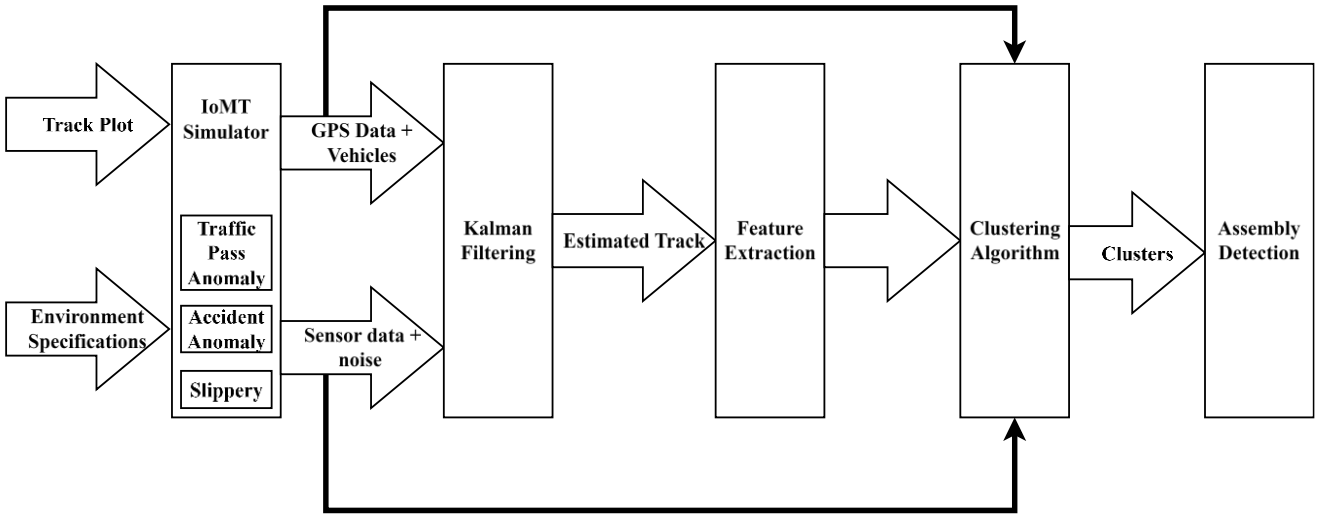

Figure 2 Conceptual diagram of the developed architecture for IoMT anomaly detection

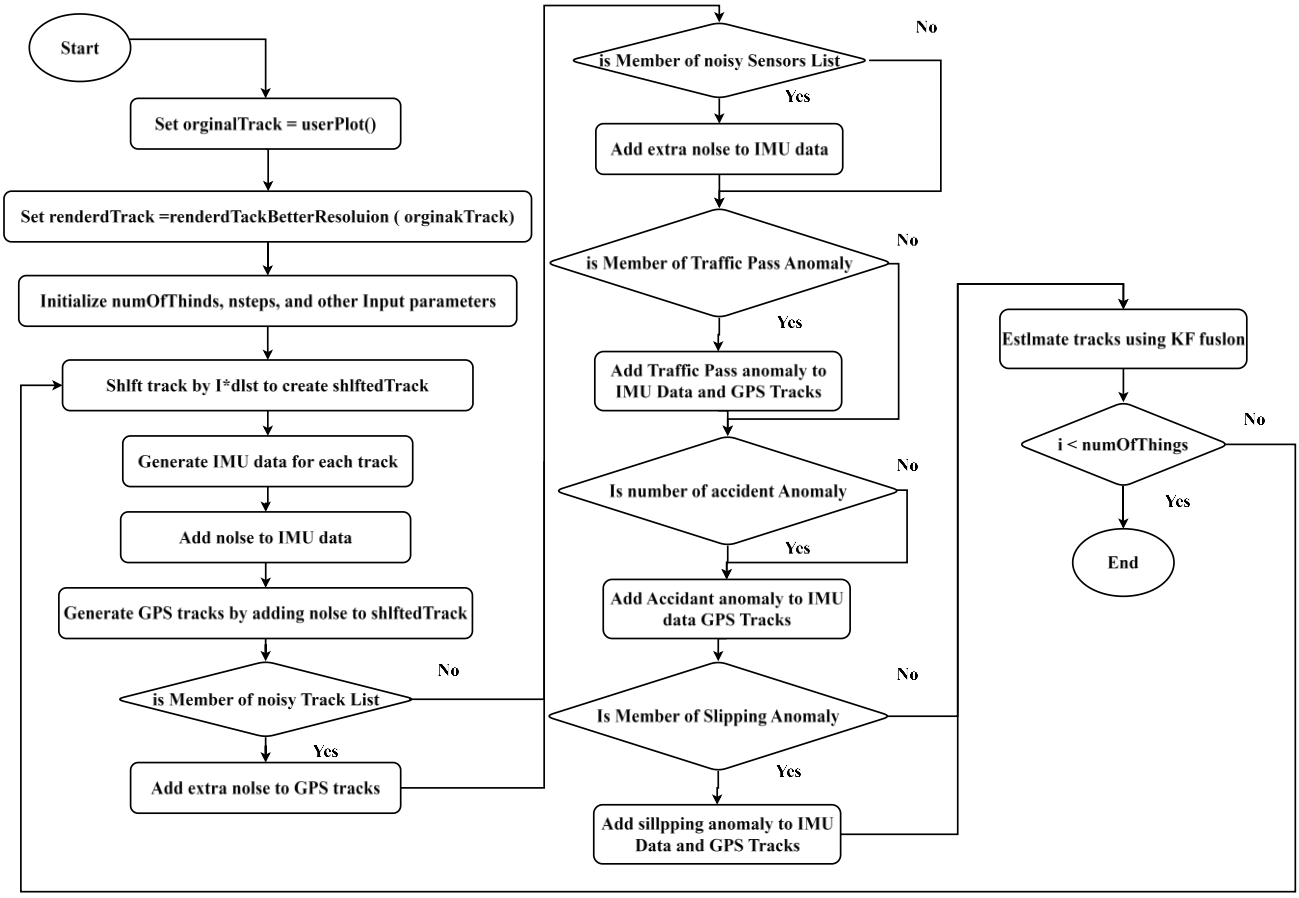

Figure 3 Flowchart illustrating the simulation of various anomalies in IoMT devices

The Flow Chart shown in FIGURE 3 depicts an overall methodology for simulating as well as identifying anomalies within IoMT devices. Initially the flow chart initializes track data and parameters. Then the loop continues with a series of steps which include generating/manipulating data for every individual device. Once this is complete the flow chart will split into several branches that generate IMU (Inertial Measurement Unit) data and add random noise to both the IMU data and GPS data. The subsequent branch then passes through multiple decision points to identify a number of anomaly types such as noisy sensors, traffic pass anomalies, accident anomalies and slipping anomalies.

3.5.1 Kalman filtering

The Kalman filter fuses data from GPS and IMU to produce trajectory estimates using:

$$X_{t+1} = AX_t \quad \text{(Process Model)} \qquad (10)$$

$$z_t = HX_t \quad \text{(Measurement Model)} \qquad (11)$$

The recursion is comprised of a two stage (predict-correct) loop shown in Algorithm 1. The first stage is called the "prediction" stage. During this stage the new estimated state and covariance at each step ahead in time are determined by propagating the current state and covariance one step into the future based upon the assumed process noise and the system dynamics. The second stage is called the "correction" or update stage. At each sampling interval, new sensor data becomes available. These data can be thought of as noisy observations of the true state. The corrections for the previous estimates of the state and covariance are determined via computation of the Kalman Gain that balances predictions based upon past knowledge with the reliability of the newest measurement. The results of both stages together provide an updated estimate of the state and an estimate of its uncertainty.

State vector:

$$X_t = (x_t, y_t, z_t, v_{x,t}, v_{y,t}, v_{z,t}, a_{x,t}, a_{y,t}, a_{z,t}) \quad (12)$$

Measurement vector:

$$z_t = (GPS_{x,t}, GPS_{y,t}, GPS_{z,t}, GPS_{vx,t}, GPS_{vy,t}, GPS_{vz,t}, IMU_{ax,t}, IMU_{ay,t}, IMU_{az,t}) \quad (13)$$

3.6 Algorithms

The system is based on three types of micro-clusters:

- Core Micro-Clusters (CMCs)
- Potential Micro-Clusters (PMCs)
- Weak Micro-Clusters (WMCs).

These micro-clusters are maintained by means of several integrated algorithms to allow for their development, degradation as well as updates of the clustering graph and finally the formation of final clusters. This hierarchical organization facilitates an efficient processing of the continuously flowing input data. In addition to this the structural representation of the algorithmic connections between the different algorithms, which can be seen in the figure 5, shows clearly how these two levels of control complement each other within the system.

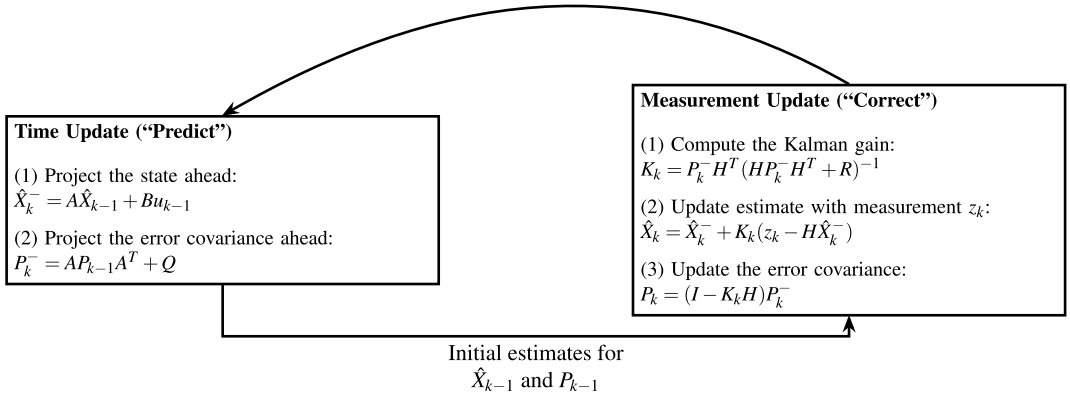

Figure 4 Conceptual diagram of Kalman filter showing predict and correct phases

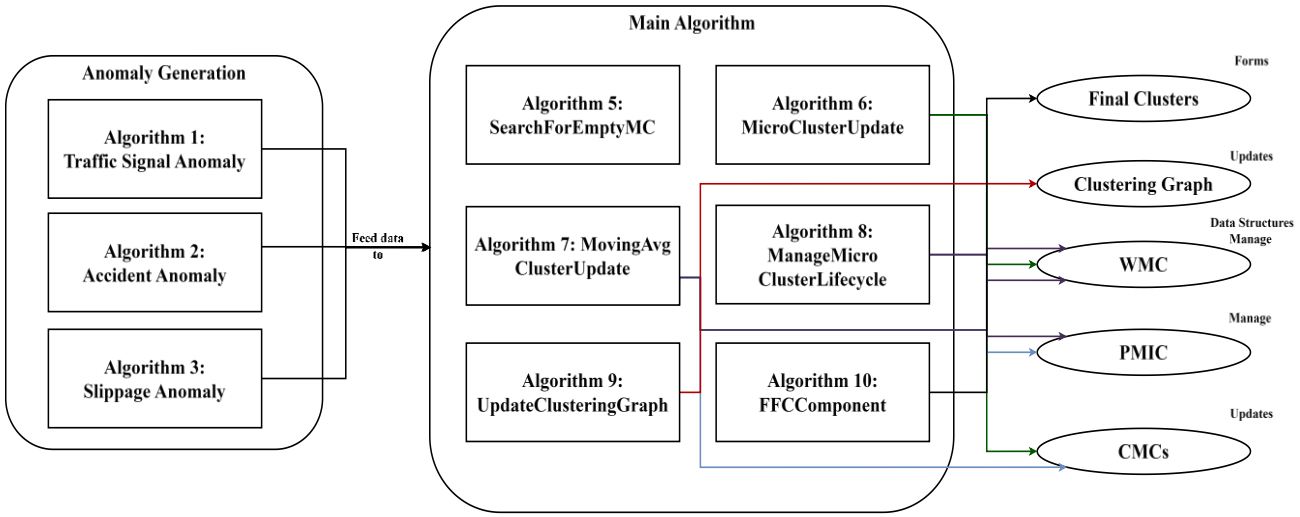

Figure 5 Visualization of the inter-connectivity between the algorithms

3.6.1 Anomaly simulation algorithms

To assess how well the framework works to withstand various types of unusual driving situations that can be expected in real world scenarios, we simulate three anomalous trajectories that are each designed to capture a specific type of real world event that is commonly seen in ITSs. These three types of anomalous events include traffic signal stop, traffic accident and road slippage. The process for simulating these anomalous events is given by the formal description provided in Algorithms 1-4.

Algorithm 1: Kalman filter for GPS–IMU state estimation

Input: Initial state estimate ˆX0, initial covariance P_0, State transition matrix A, control matrix B, observation matrix H, Process noise covariance Q, measurement noise covariance R, Control input sequence ut, measurement sequence z_t

Output: Filtered state estimates $\hat{X}_t$

State vector:

$$Xt = (x_t, y_t, z_t, v_{x,t}, v_{y,t}, v_{z,t}, a_{x,t}, a_{y,t}, a_{z,t});$$

Measurement vector:

$$zt = (GPS_{x,t}, GPS_{y,t}, GPS_{z,t}, GPS_{vx,t}, GPS_{vy,t}, GPS_{vz,t}, IMU_{ax,t}, IMU_{ay,t}, IMU_{az,t});$$

for t ←1to T do

// Prediction Phase

$\hat{X}_t^- \leftarrow A\hat{X}_{t-1} + Bu_t - 1;$

$P_t^- \leftarrow AP_{t-1}A^T + Q;$

// Correction Phase

$K_t \leftarrow P_t^- H^T (HP_t^- H^T + R)^{-1};$

$\hat{X}_t \leftarrow \hat{X}_t + K_t(z_t - H\hat{X}_t^-);$

$P_t \leftarrow (I - K_t H)P_t^-;$

Algorithm 2 provides for the simulation of a traffic signal anomaly. The algorithm represents cars (vehicles) as they stop at an intersection because of a red light. In order to generate this anomaly based on the available synchronized GPS data and IMU measurement data, the algorithm processes all trajectory records except one which will be referred to as the "pass through" record. The pass-through record models a car that is allowed to go across the intersection. For each other than pass through record, the algorithm injects a stopping interval at the defined traffic point by freezing the GPS data's spatial displacement and freeze the motion related signals from the IMU data stream for a fixed number of time steps. The result generated by this process is typical queuing behaviors observed at signalized intersections.

Algorithm 2: Traffic signal anomaly simulation

Input: GPSTracks, IMUData, trafficPoint, passingTrack, nsteps

Output: Modified GPSTracks, IMUData

foreach track i in GPSTracks do

  if i ≠ passingTrack then

    start ←traf ficPoint ×nsteps;

  end ←start+nsteps;

  Insert stopping period into GPSTracks[i][start : end];

  Insert stopping period into IMUData[i][start : end]

Algorithm 3 simulates an accident-caused anomaly where a vehicle comes to a sudden stop; and after this point, all future GPS locations will be static as there is no additional motion. The IMU data for points beyond the accident location (with the exception of some minor variations) will also be constant or very small. The goal

here was to physically model how a crash would affect a vehicle's ability to move as well as simulate what may happen to sensors on the vehicle during a collision.

Algorithm 3: Accident anomaly simulation

Input: GPSTrackA, IMUDataA, accidentPoint
Output: Modified GPSTrackA, IMUDataA
start ← accidentPoint;
for t ←start to end of sequence do
  Set GPSTrackA[t] to constant (no movement);
  Set IMUDataA[t] to constant or zero values;

Algorithm 4 generates an initial slippage effect, which is an example of transient loss of traction due to the road's slippery surface characteristics (e.g., ice, sand, oil spill). A small, smooth sinusoidal variation in magnitude is added to the original IMU data within a pre-defined length for each time the vehicle slips. The slippage magnitude is determined by the slippage intensity. This method maintains physical reasonableness (the onset and end of the "slip" are smooth and do not abruptly cease), however it does create dynamic oscillations that are representative of both wheel slip and lateral instability.

Algorithm 4: Slippage anomaly simulation

Input: GPSTrackSL, IMUDataSL, slippingPoint, slpMag, slippingLength, nsteps
Output: Modified IMUDataSL
start ← slippingPoint ×nsteps;
for k ←1to slippingLength do
$IMUDataSL[start + k] \leftarrow IMUDataSL[start + k] + slpMag \times \sin\left(\pi \times \frac{k}{slippingLength}\right)$;

In addition to providing an environment to evaluate the performance of the developed techniques with respect to anomalies using multiple types of simulated events, the three described anomaly simulations also allow for a controlled test of the robustness of the technique when faced with different types of disturbance. Thus, together they enable the testing of performance of both detection and clustering over a variety of realistic but highly variable traffic scenarios.

3.6.2 MUDEDS main algorithm

In order to provide additional insight into how MUDEDS operates, the following is a detailed specification of how each part of the proposed MUDEDS algorithm will function as specified by Algorithm 5. This provides a procedural overview of how MUDEDS maintains its micro-clusters and their lifecycles on-line and periodically extracts clusters from them. A complementary diagrammatic illustration of how all parts of MUDEDS operate is presented in Figure 6. The figure illustrates MUDEDS' data processing path from continuous ingestion of stream data through priority based micro-cluster assignments and adaptive promotions/decays to forming global level macro-clusters with horizons. Taken together the diagrams and pseudocode illustrate how MUDEDS performs its adaptive actions in real time in terms of how local changes at the micro-level produce a single consistent global level grouping of objects.

Algorithm 5: MUDEDS main algorithm

Input: GridGranularity, Horizon, DataStream, MinPts
Output: Clusters
Grid ← createGrid(GridGranularity);

```
Initialize Core Micro-Clusters (CMCs), Potential Micro-Clusters (PMCs), Weak Micro-Clusters
(WMCs);
G←emptygraph;
Clusters ← ϕ;
endOfStream ← false;
T_c ←1;
while not endOfStream do
  P←DataStream(Tc);
  T ←searchForTargetMC(CMCs, PMCs, WMCs, P);
  if T = /0 then
    Create new PMC and insert into Grid;
  else
  Update target micro-cluster using MicroClusterUpdate;
  Update energy values of CMCs, PMCs, and WMCs;
  Run ManageMicroClusterLifecycle;
  if Tc mod Horizon =0 then
    Clusters1 ← FFCComponent(CMCs, MinPts);
    Clusters2 ← assignToMicroCluster(CMCs, Clusters1);
    Clusters ←Clusters1∩Clusters2;
    T_c ←T_c+1;
    if DataStream(Tc) is null then
      endOfStream ←true;;
```

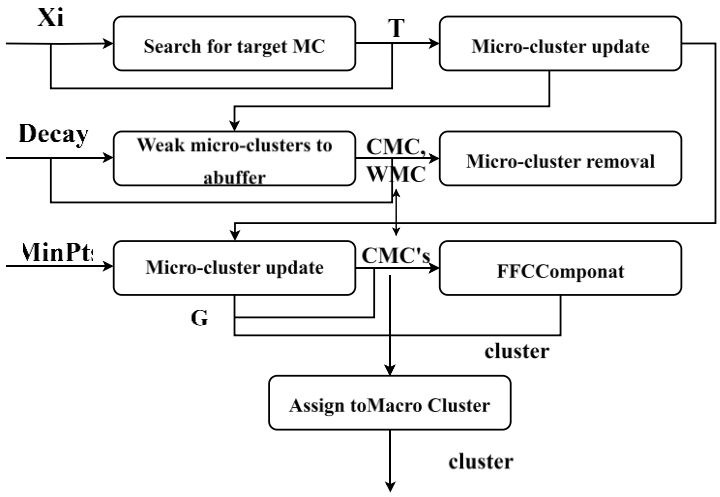

Figure 6 Conceptual diagram of the data flow and sub-operation of MUDEDS algorithm

3.6.3 Supporting algorithms and operational flow

The MUDEDS system has many processes working together (they are tightly coupled) which help to decide how clusters will be assigned, grow, die, and eventually become full clusters. Each process works constantly with the primary stream processing loop, helping the algorithm to adapt to changing data streams.

When a new data record arrives at the algorithm it begins a search for the best (target) micro-cluster. The search goes sequentially through three types of micro-clusters: weak, potential and core. It tries to find a micro-cluster that meets some pre-defined criteria about being close enough. A good example of a priority-based search is searching through loosely organized structures first. By doing so, it avoids splitting up an already loose group

into smaller groups. When a suitable micro-cluster is not found then a new potential micro-cluster can be created and added into the spatial grid structure.

After finding a suitable target micro-cluster, the micro-cluster update function is called. The function updates the density level of the target micro-cluster. The function also updates the geometry and time attributes of the micro-cluster such as radius and energy. Once a micro-cluster's density reaches a certain level, if it was either a potential or weak micro-cluster, it becomes a core micro-cluster. This represents a denser region of data than existed previously. At the same time, the clustering graph is updated to reflect the relationship between two core micro-clusters. Between each pair of core micro-clusters, the distance between the centers of both clusters (Euclidean distance), and the amount of space they share (intersection distance), are calculated. If there is an overlap or if adjacent, then an edge is created in both directions between the pairs of micro-clusters to represent connectivity for further cluster creation.

MUDEDS uses several methods to control the life cycle of the micro-clusters. One method involves decreasing the energy value of all the micro-clusters at each iteration by a constant. When a micro-cluster's energy drops to less than zero, then that micro-cluster is reduced to a weak status. Weak micro-clusters have all of their links in the clustering graph deleted; they lose their current values; and they are moved into the weak buffer. Weak micro-clusters go through similar cycles, but once they reach zero energy they disappear completely. This helps MUDEDS adjust to changes in the concepts of the data stream over time.

There are instances where MUDEDS stops maintaining the clustering structure and instead performs an off-line clustering operation. To do this, MUDEDS defines a horizon parameter. When this parameter is reached, MUDEDS stops the clustering operations, marks all of the current core micro-clusters as "unvisited", and retrieves all of the neighboring micro-clusters for each unvisited core micro-cluster using the spatial grid. After retrieving all of the neighboring micro-clusters, if one or more neighbors satisfy the required number of members to define a new cluster then a new cluster is defined and expanded by traversing the graph until no additional members are found. However, if none of the neighboring micro-clusters meet this criterion then they are considered noise. The off-line clustering operation generates a complete clustering diagram that preserves the localized density information generated during on-line processing.

These supporting functions create a single operational sequence that allows MUDEDS to maintain both response times and stability and efficiency when dealing with very large volumes of streaming data.

3.7 Evaluation metrics

A variety of metrics are utilized in order to provide an overall assessment of the quality of clustering and efficiency of computation for the stream clustering methods evaluated. The chosen evaluation metrics measure different properties of clustering; specifically, they assess the homogeneity of clusters (clustering purity), the similarity between clusters determined by different clustering methods (pairwise consistency), how well two or more different clustering methods agree on the number of clusters in common (overlap accuracy) and resource utilization.

1. **Purity:** Purity measures the degree to which each identified cluster consists entirely of records that belong to one particular ground-truth class. Purity is defined as Equation (14)

$$Purity = \frac{1}{N}\sum_k \max_j |C_k \cap L_j|, \qquad (14)$$

where C_k is the kth predicted cluster, L_j indicates the jth true label for a sample and N is the total number of samples. Purity scores are greater when samples in a single cluster have similar labels; however, higher purity can be achieved by generating more clusters.

2. **Rand Index (RI):** This index measures how well a prediction agrees with the ground truth for all possible pairings of two samples from either the prediction or the ground truth. RI is given by Equation (15).

$$RI = \frac{TP+TN}{TP+FP+FN+TN}, \quad (15)$$

In this equation, TP, TN, FP, and FN represent the number of pairs in which both objects are classified correctly (TP), incorrectly but similarly (TN), incorrectly but differently (FP), or incorrectly with regard to both classification criteria (FN). RI therefore gives an equally weighted measure of overall consistency in clustering.

3. **The Jaccard Index** assesses similarities in assigned clusters with true clusters using False Associations to penalize those, and is given by Equation (16).

$$Jaccard = \frac{TP}{TP+FP+FN}. \quad (16)$$

This metric is stricter than RI because it does not include true negatives and therefore highlights good assignment of positive values.

4. **Fowlkes-Mallows Index (FM):** The Fowlkes-Mallows Index measures the quality of the clusters through precision and recall in terms of pairs. The index is calculated with the Equation 17:

$$FM = \sqrt{Precision \times Recall}, \quad (17)$$

where precision and recall are measures for how accurate and complete the positive-pair-assignments were. In particular this index can be used to compare structural similarities of two clusterings.

5. **F-measure:** With F-measure you get one number which is the harmonic average of precision and recall; it is a measure for both false positives and false negatives.

$$F = \frac{2\times TP}{2\times TP+FP+FN}. \quad (18)$$

In particular, this measure can be useful when dealing with imbalanced datasets; it represents how well an algorithm performs on all clusters collectively.

6. Computational Costs: The computational costs of each algorithm are evaluated using the total processing times for each cluster, as indicated by Equation (19).

$$Time = t_{end} - t_{start}. \quad (19)$$

This measure can evaluate if an algorithm is feasible for use in real-time applications or those that have limited resources. It will also help determine if an application has sufficient "scalability" for potential use in IoT devices.

4. Experimental results and evaluation

This area provides an in-depth experimental evaluation of the proposed MUDEDS method as compared to three state-of-the-art streaming clustering baseline methods (BOCEDS, CEDAS, and MUDI). The experimental design was developed to evaluate the effectiveness of each method for cluster formation while also assessing their deployability in a real-world setting utilizing the Internet of Moving Things (IoMT). For these purposes, we employ several external validity measures to assess the quality of the clusters formed by each method. We will use timing and memory usage to measure how much time and memory it takes to compute the results. Our objective in conducting these evaluations will be to provide a thorough examination of how well each algorithm can be adapted and scaled based on varying types of anomalies within dynamic trajectory datasets that contain anomalous patterns. In order to allow for fair and reproducible assessments, all of the methods were tested using identical data streams and parameter settings.

4.1 Experimental design

Experimental Design. The proposed experimental design is intended to mimic realistic IoMT use cases using multiple mobile units that are each equipped with GPS and IMU (inertial measurement unit) sensors. Synthetic, stream-based trajectory data sets are created by simulating user-controlled motion patterns and injecting specific

types of anomalies into them (e.g., traffic signal stop; vehicle accident stop; sensor failure; slippage event on a road surface). Anomalies include both transient events and persistent states and also represent both sensor level and motion level disruptions.

In order to provide unbiased comparative results for the various methods under test, the respective parameters for each method were determined based upon the recommended settings in their original study or empirically set to allow the methods to operate stably without bias towards one or another of the methods. A summary of the configuration of all of the parameters tested in this study for MUDEDS and its comparative methods can be seen in Table 2. For example, parameters affecting how micro-clusters evolve, grow, shrink, and ultimately die are decay rates and density thresholds and/or other constraints (radius constraints). In contrast, parameters such as grid granularity, min-points, and horizon setting(s) control spatial partitioning and periodic clustering behavior.

Also summarized in Table 3 are the parameters associated with creating the data for a particular scenario. These parameters establish the number of movement tracks, time-resolution of tracking, location of anomalies within a track, magnitude of an anomaly relative to a nominal condition, and distance separating two different movement tracks. The ability to systemically vary all of these parameters allows for a direct correlation between the observed characteristics of the clusters produced by the various methods being compared with the design features of those methods. Therefore, there should be no ambiguity regarding whether differences in cluster characteristics are due to variations in the input conditions or to inherent properties of the algorithms. All testing was performed multiple times in order to quantify variance in performance among runs and enable statistical comparison of the average performance per run among all methods tested.

Table 2 Parameter settings for MUDEDS and benchmarks

Parameter	**MUDEDS**	**BOCEDS**	**CEDAS**	**MUDI**
Decay	50	50	50	-
CMThresh	10	10	-	-
Rmax	0.1	0.1	-	-
Rmin	0.05	0.05	-	-
Radius	-	-	0.01	-
minThreshold	-	-	3	-
Stream speed	-	-	-	50
Cl	-	-	-	0.5
Lambda	-	-	-	0.998
gridGranularity	10	-	-	10
minPts	3	-	-	3
Horizon	2	-	-	2

Table 3 Scenario configurations for evaluation

Parameter	**Value**	**Description**
numOfRunningTracks	5	Number of moving things
nsteps	500	Steps between each 2 points
slippingAnomaly	2	Tracks where slipping happening
noisyTracks	1	Tracks with GPS sensor failure
noisySensorList	3	Tracks with IMU sensor failure
trafficPassAnomaly	4	Tracks with traffic light passing
accidentAnomaly	5	Tracks where accident happening
slpMag	25	Magnitude of slipping
slippingLength	250	Duration of slipping event
dist	100	Distance on Y axis between tracks

4.2 Experimental results and analysis

Here, we present an overall comparison of our new MUDEDS algorithm and the three reference stream-clustering algorithms (BOCEDS, CEDAS, MUDI) on their performance in terms of clustering quality, computation time and used storage space. We use boxplots for representing both the typical values as well as the variability and robustness among multiple executions.

Purity Evaluation: Purity is a measure of how pure a class is. An algorithm that can effectively cluster different dominant classes will have good purity. It appears from figure 7 that CEDAS achieved the best median purity of about 0.81. Therefore, CEDAS produced the most consistent clustering results. Although MUDEDS has the highest maximum purity value (of 0.84) and a median purity of 0.72, this indicates that although MUDEDS produces clusters that are adaptable in nature, there were times when MUDEDS was able to produce very consistent (homogeneous) clusters. BOCEDS had a moderate level of purity (about 0.76). In addition, MUDI performed poorly in terms of purity (only 0.04) and therefore does not appear to be effective at separating significant structures in streams of trajectory data. Also, the larger variation in purity seen in MUDEDS represents the fact that MUDEDS adaptively handles density variations rather than instability, which would be preferable in non-stationary streaming environments.

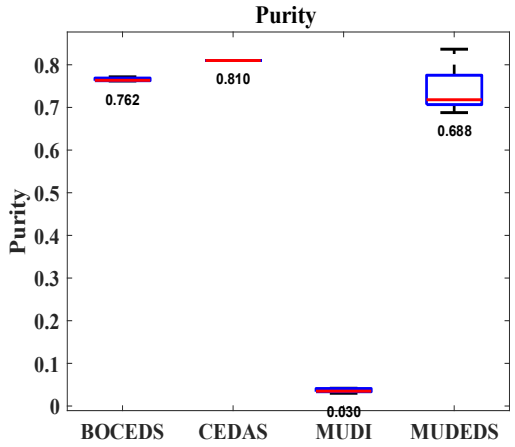

Figure 7 Purity comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

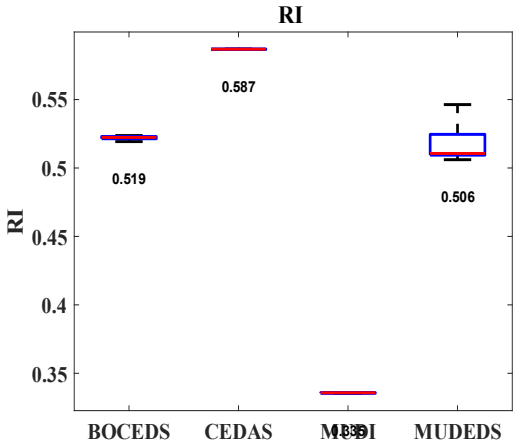

Figure 8 Rand Index comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

In order for the above text to be written as sounding more "human," it can sound as follows:

"The Rand Index is an evaluation metric that measures the level of agreement between pairs of actual cluster assignments and cluster assignments generated by algorithms. Figure 8 shows that CEDAS has the highest Rand Index value (approximately 0.587) with the least amount of variability, showing that CEDAS's algorithm generates clusters consistently over multiple runs. MUDEDS is ranked second in terms of its Rand Index score (the highest possible score was 0.545 and the median possible score was 0.511) which indicates that MUDEDS provides a high degree of consistency between individual cluster assignments while still being able to adapt to changing streams. BOCEDS provides a moderate Rand Index score (approximately 0.522), and the lowest Rand Index score among all three algorithms was provided by MUDI (a Rand Index score of 0.336). The spread within the box plot also illustrates the trade-off MUDEDS made between having a stable algorithm vs. one that could respond to changes in the input data when compared to how rigidly CEDAS operates.

The Jaccard index is an even better measure than silhouette for evaluating how well clusters are overlapping in terms of true positive agreements as it also penalizes false positives. As shown in figure nine, cedas has the largest jaccard score (0.525) therefore showing the strongest level of agreement globally across all of the clusters

identified on both sides. Mudeds does reasonably well in comparison (with a maximum of 0.508 and median of 0.442) since mudeds will preserve the overlaps effectively through adaptive density. Boeds is doing moderately well (0.465). In contrast, mudi shows very little value (0.012) which further supports that mudi does not create significant intersectional overlap within clusters when dealing with high dimensional streaming data.

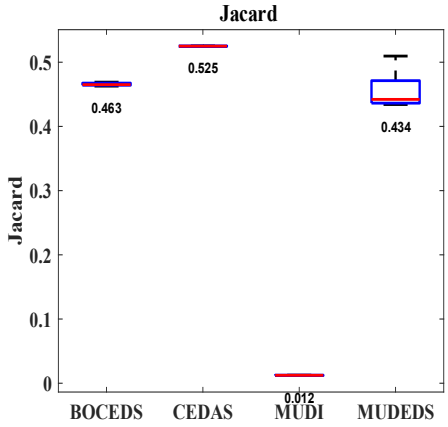

Figure 9 Jaccard Index comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

Fowlkes-Mallows Index Evaluation Summary: The Fowlkes-Mallows (FM) index is an evaluation metric that combines precision and recall to evaluate the structural similarity between clusters. In figure ten we observe that CEDAS has the highest FM value (≈0.689) which represents the greatest structural similarity among all evaluated methods to the ground truth. MUDEDS is ranked as second best, achieving a maximum of ≈0.675 and a median of ≈0.614 and thus demonstrating high quality clustering results. BOCEDS, on the other hand, consistently produces poor, yet steady results (≈0.635). Again, MUDI does poorly (0.097). The analysis of the FM distribution shows that MUDEDS provides a very good representation of the cluster structure and is able to do this well due to its ability to adapt well to changes in data streams.

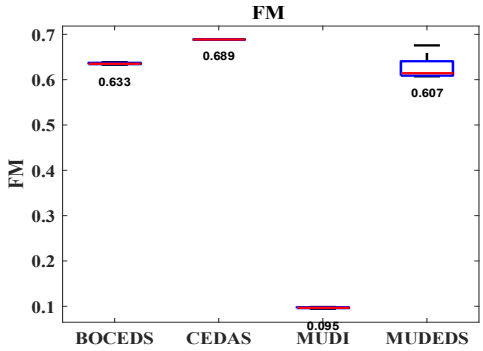

Figure 10 Fowlkes-Mallows Index comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

Overall Clustering Effectiveness: F-Measure The F-Measure is used to determine how well a clusterer performs at identifying which cases belong in each cluster when comparing it against the results of another clusterer. According to Fig. 11, CEDAS and BOCEDS are clearly the best performing clusterers based on this measure as they both have high F-Measures (approximately 0.669 & 0.667). These results confirm that these clusterers are good at identifying clusters using classification-oriented clustering. MUDEDS had the next highest F-Measure of approximately 0.646 indicating that while the clusterer's ability to adaptively vary the density of the clusters may slightly degrade performance compared to CEDAS and BOCEDS; it still performed very well. Conversely, MUDI had a very poor F-Measure of about .022 and therefore confirms its inability to perform well enough to be useful in complex data streaming applications.

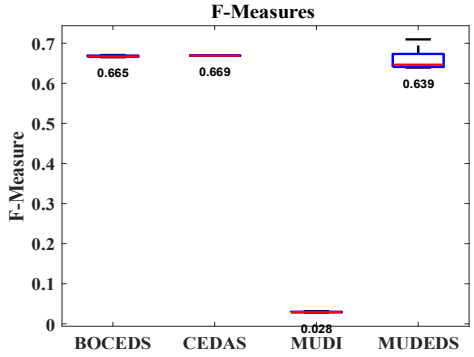

Figure 11 F-Measure comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

Computational Time Assessment: The time taken to execute algorithms is as important as how well an algorithm clusters when performing real-time stream analysis. As can be seen from FIGURE 12, the computational requirements of CEDAS are very large (approximately 1,459,709) making this method unsuitable for use in either high throughput environments or constrained environments. Similarly, while MUDI requires a significantly less amount of computing resources compared with CEDAS (approximately 128,583 units), both methods require substantially more computational resources than do BOCEDS and MUDEDS, which each require fewer than 200 units; that is to say they have at least several orders of magnitude better computational performance characteristics than does CEDAS.

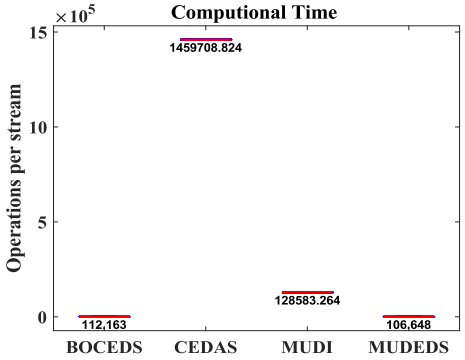

Figure 12 Computational Time comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

Memory Usage Comparison Practical Deployability is also supported by the memory consumption. CEDAS has a high memory footprint as shown in figure thirteen with a total memory usage of approximately 7499. The memory usage required by MUDI was found to be relatively low at about 862. On the other hand both BOCEDS and MUDEDS were found to have very low memory requirements using only approximately 3.29 and 3.35 units. Therefore, MUDEDS can provide greater than 2000 times better memory performance when compared to CEDAS; making it ideal for use on embedded devices and for long duration video or audio streaming.

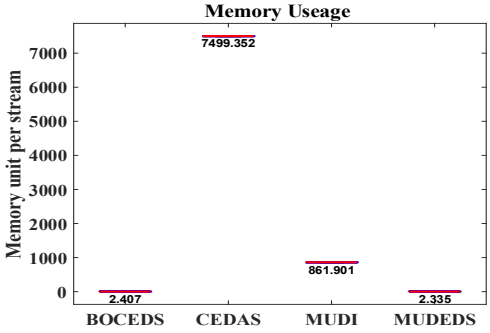

Figure 13 Memory consumption comparison of the four algorithms: BOCEDS, CEDAS, MUDI, and MUDEDS

Table 4. Comparative performance summary

Metric	MUDEDS	BOCEDS	CEDAS	MUDI	Best
Purity	0.844	0.764	0.810	0.035	MUDEDS
Rand Index	0.545	0.522	0.587	0.336	CEDAS
Jaccard	0.508	0.465	0.525	0.012	CEDAS
FM Index	0.675	0.635	0.689	0.097	CEDAS
F-Measure	0.646	0.667	0.669	0.022	CEDAS
Time (units)	152	150	1,459,709	128,583	BOCEDS
Memory	3.35	3.29	7,499	862	BOCEDS

Summary: MUDEDS offers exceptional balance of clustering quality and computational efficiency. While CEDAS slightly outperforms in clustering quality metrics, MUDEDS achieves comparable performance with dramatically lower computational and memory requirements computational efficiency thousands of times better and memory usage over 2,000 times more efficient.

4.3 Critical discussion and implications

When comparing CEDAS to MUDEDS through the lens of clustering quality, it is evident that CEDAS generates significantly better clustering quality relative to MUDEDS. CEDAS produces superior Rand Index, Jaccard Index, Fowlkes-Mallows Index and F-measure values when compared to MUDEDS; therefore, there is clear evidence that CEDAS achieves stronger label agreement and greater pairwise consistency.

However, as indicated by the purity analysis, the high-quality clusters generated by CEDAS do have a drawback. MUDEDS achieved the largest maximum purity value and thus, MUDEDS will produce clusters with very high homogeneity (i.e., pure) in many cases as good as those produced by CEDAS, and sometimes even better. The ability of MUDEDS to create such pure clusters under certain conditions implies that MUDEDS excels at creating clusters around densely packed anomalies (even though MUDEDS has less perfect global pairwise consistency).

In addition to cluster quality, stability of clustering across multiple runs was also evaluated. Box plots were used to evaluate the degree of variation observed during each run. As indicated by the box plot values, CEDAS exhibited relatively small inter-quartile ranges for almost every metric examined. These values demonstrate consistent performance. However, they also suggest that CEDAS may lack the flexibility needed to respond appropriately to changes in the data distribution being analyzed.

MUDEDS demonstrated moderate variability in the same type of plots. While the degree of variation could be interpreted as representing some level of "instability," this behavior can also be viewed as a form of adaptability. Because MUDEDS responds to variations in the underlying data distribution as they occur, this feature represents a significant advantage in nonstationary streaming environments. In such systems, a stable clustering algorithm is likely to react slowly to changing input patterns and/or will generate poorly adapted responses.

BOCEDS demonstrated intermediate performance between MUDEDS and CEDAS. BOCEDS produced competitive F-measures and moderate scores for nearly all other metrics. Its low computational and memory demand makes BOCEDS a suitable option for use in environments that are constrained by resources. It did not exhibit sufficient adaptability to outperform MUDEDS in response to complex patterns of anomalies.

MUDI performed poorly in comparison to MUDEDS and CEDAS for nearly all metrics. Despite its relatively modest computational requirements, MUDI's poor performance suggests that merely providing efficient processing does not necessarily imply success with regards to clustering unless the clustering method employed is structurally robust enough to accurately model density relationships within the data.

A thorough examination of the computing and memory overhead associated with CEDAS clearly illustrates its practical limitations. Computations associated with CEDAS are orders of magnitude larger than those associated

with either MUDEDS or BOCEDS; therefore, CEDAS would be impractical for deployment in high-throughput or embedded systems. A much more reasonable approach to analyzing the performance of these algorithms is based upon their combined clustering quality and efficiency. For example, MUDEDS achieved near optimal clustering quality while demonstrating approximately 10,000 times improved efficiency in terms of computation and 2000 times reduced memory usage when compared to CEDAS. This improvement is largely due to its multi-density buffering mechanism. Instead of calculating density globally for all points, MUDEDS employs an adaptive local clustering technique that relies upon buffers containing data points with varying densities. Therefore, it can avoid exhaustive calculations of density relationships throughout the entire dataset.

These findings provide strong support for the notion that MUDEDS provides a more reasonable trade-off between accuracy and efficiency in real world stream clustering problems. Specifically, MUDEDS should find application in areas like ITS where both accuracy and efficiency are important. Although CEDAS is preferred in scenarios where high quality clustering is a priority and available system resources are unconstrained, MUDEDS provides a scalable and deployable alternative without sacrifice to meaningful clustering performance

5. Conclusion and future work

This work presented MUDEDS (Multi-Density Data Streams Clustering), a new clustering paradigm designed specifically for robustly detecting anomalies in Internet of Moving Things (IoMT) environments. MUDEDS' architecture was developed to address key weaknesses inherent in most all current density-based stream clustering paradigms. Those are (1) those approaches require knowledge of the stationary data distribution and therefore do not function well when the distribution is nonstationary; and (2) they consume significant amounts of CPU cycles and memory, which makes it difficult to cluster at very high velocities. Experimental results were used to demonstrate that MUDEDS achieves similar clustering quality to best-in-class algorithms like CEDAS (Clustering in Evolving Data Streams), yet consumes significantly less computer resources with respect to processing time, by several orders of magnitude; and with respect to memory, over two thousand times less.

MUDEDS' unique strengths stem from its hybrid nature combining both continuous real-time processing and periodic density aware clustering. The introduction of different micro-cluster types, namely core, potential, and weak clusters and associated lifecycle management, enables MUDEDS to continuously adapt to changes in the data stream without performing an expensive global density recalculation. The suitability of MUDEDS was evaluated in numerous anomalous conditions representative of real world IoMT applications, i.e., traffic light violations, accidents induced stops, and slippery roadways. Regardless of the specific condition, MUDEDS has proven itself capable of achieving a suitable tradeoff among adaptability, clustering quality and computational cost.

Several research avenues will likely result from this work. One promising area is the inclusion of adaptive threshold adjustment techniques to allow MUDEDS to dynamically adjust clustering parameters based upon changes occurring within the input data stream. An additional area is integrating machine learning components into the framework allowing MUDEDS to discover higher-level spatio-temporal relationships and semantics in high dimensional datasets. The scalability of MUDEDS can be improved via development of distributed/parallel versions tailored toward large scale IoMT applications. Additionally, supporting automatic anomaly classification within the framework will provide a method to not only detect but also semantically interpret anomalous event occurrences. From a system viewpoint, developing private versions of MUDEDS using either differential privacy or federated learning could make it deployable in highly restricted environments. Lastly, although this work focused primarily on IoMT-type applications, the foundational concepts of MUDEDS are generalizable to other streaming application areas, e.g., wearable health monitoring and Industrial IoT.

In conclusion, MUDEDS provides a practical and scalable basis for performing real-time stream clustering with awareness of anomalous behavior. With respect to its accuracy-adaptability-efficiency tradeoffs, it represents a strong candidate for use in future generations of intelligent sensing and monitoring systems.

Funding

The authors received no financial support for the research.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, F.A. Al-Ibraheemi and F. Mardukhi; methodology, F.A. Al-Ibraheemi; software, F.A. Al-Ibraheemi; validation, F.A. Al-Ibraheemi and F. Mardukhi; formal analysis, F.A. Al-Ibraheemi; investigation, F.A. Al-Ibraheemi; resources, F. Mardukhi; data curation, F.A. Al-Ibraheemi; writing—original draft preparation, F.A. Al-Ibraheemi; writing—review and editing, F. Mardukhi; visualization, F.A. Al-Ibraheemi; supervision, F. Mardukhi; project administration, F. Mardukhi.

Reference

[1] G. Wu, "Monitoring system of key technical features of male tennis players based on Internet of Things security technology," *Wireless Communications and Mobile Computing*, vol. 2021, 2021. https://doi.org/10.1155/2021/5562953

[2] S. Neelakandan *et al.*, "IoT-based traffic prediction and traffic signal control system for smart city," *Soft Computing*, vol. 25, no. 18, pp. 12241–12248, 2021. https://doi.org/10.1007/s00500-021-06041-9

[3] J. Tian, W. Ding, C. Wu, and K. W. Nam, "A generalized approach for anomaly detection from the Internet of moving things," *IEEE Access*, vol. 7, pp. 144972–144982, 2019. https://doi.org/10.1109/ACCESS.2019.2945596

[4] A. Kamilaris and N. Botteghi, "The penetration of Internet of Things in robotics: Towards a web of robotic things," *Journal of Ambient Intelligence and Smart Environments*, vol. 12, no. 6, pp. 491–512, 2020. https://doi.org/10.3233/AIS-200585

[5] S. Mittal *et al.*, "A novel abnormality annotation database for COVID-19 affected frontal lung X-rays," *PLOS ONE*, vol. 17, no. 10, p. e0271931, 2022. https://doi.org/10.1371/journal.pone.0271931

[6] K. L. Pham *et al.*, "The study of electrical energy power supply system for UAVs based on the energy storage technology," *Aerospace*, vol. 9, no. 9, p. 500, 2022. https://doi.org/10.3390/aerospace9090500

[7] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Driver behavior profiling using smartphones: A low-cost platform for driver monitoring," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 91–102, 2015. https://doi.org/10.1109/MITS.2014.2328673

[8] Z. M. Yusoff *et al.*, "Smart Clothline System Based on Internet of Thing (IoT)," *MATEC Web of Conferences*, vol. 248, p. 02002, 2018. https://doi.org/10.1051/matecconf/201824802002

[9] G. Slavic *et al.*, "Anomaly detection in video data based on probabilistic latent space models," in *Proc. IEEE*, 2020, pp. 1–8. https://doi.org/10.1109/ICIP40778.2020.9191178

[10] X. Li and B. Zou, "An automated data engineering pipeline for anomaly detection of IoT sensor data," *arXiv preprint*, 2021. https://arxiv.org/abs/2101.13828

[11] X. Kang, B. Song, and F. Sun, "A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT," *Applied Sciences*, vol. 9, no. 1, p. 135, 2019. https://doi.org/10.3390/app9010135

[12] J.-J. Chung and H.-J. Kim, "An automobile environment detection system based on deep neural network and its implementation using IoT-enabled in-vehicle air quality sensors," *Sustainability*, vol. 12, no. 6, p. 2475, 2020. https://doi.org/10.3390/su12062475

[13] M.-H. Jeong *et al.*, "Vessel trajectory reconstruction based on functional data analysis using automatic identification system data," *Applied Sciences*, vol. 10, no. 3, p. 881, 2020. https://doi.org/10.3390/app10030881

[14] S. A. Wagan *et al.*, "A fuzzy-based duo-secure multi-modal framework for IoMT anomaly detection," *Journal of King Saud University – Computer and Information Sciences*, vol. 35, no. 1, pp. 131–144, 2023. https://doi.org/10.1016/j.jksuci.2021.05.012

[15] D. Liu *et al.*, "Sensors anomaly detection of industrial internet of things based on isolated forest algorithm and data compression," *Scientific Programming*, vol. 2021, pp. 1–9, 2021. https://doi.org/10.1155/2021/8889026

[16] X. Yu *et al.*, "An adaptive method based on contextual anomaly detection in Internet of Things through wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 16, no. 5, 2020. https://doi.org/10.1177/1550147720920478

[17] R. Alanazi and A. Aljuhani, "Anomaly detection for industrial Internet of Things cyberattacks," *Computer Systems Science and Engineering*, vol. 44, no. 3, 2023. https://doi.org/10.32604/csse.2023.030182

[18] M. Y. I. Basheer *et al.*, "Autonomous anomaly detection for streaming data," *Knowledge-Based Systems*, vol. 284, p. 111235, 2024. https://doi.org/10.1016/j.knosys.2023.111235

[19] X. Yu, H. Wang, K. Dong, and C. Chen, "A novel LDVP-based anomaly detection method for data streams," *International Journal of Computers and Applications*, 2024. https://doi.org/10.1080/1206212X.2024.2301234

[20] R. Hyde, P. Angelov, and A. R. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Information Sciences*, vol. 382, pp. 96–114, 2017. https://doi.org/10.1016/j.ins.2016.12.016

[21] M. K. Islam, M. M. Ahmed, and K. Z. Zamli, "A buffer-based online clustering for evolving data stream," *Information Sciences*, vol. 489, pp. 113–135, 2019. https://doi.org/10.1016/j.ins.2019.03.003

[22] A. Amini *et al.*, "MuDi-Stream: A multi density clustering algorithm for evolving data stream," *Journal of Network and Computer Applications*, vol. 59, pp. 370–385, 2016. https://doi.org/10.1016/j.jnca.2015.06.011