Bio-Inspired Anomaly Detection with Cursory Dendritic Cell Algorithm

Subjects: Computer Science, Artificial Intelligence

Contributor: Rui Pinto

The autonomous and adaptable identification of anomalies in industrial contexts, particularly in the physical processes of Cyber-Physical Production Systems (CPPS), requires using critical technologies to identify failures correctly. Most of the existing solutions in the anomaly detection research area do not consider such systems' dynamics. Immune-based models, such as the Dendritic Cell Algorithm (DCA), may provide a rich source of inspiration for detecting anomalies. Cursory Dendritic Cell Algorithm (CDCA), is a novel variation of the Dendritic Cell Algorithm (DCA), developed to be flexible and monitor physical industrial processes continually while detecting anomalies in an online fashion.

Artificial Immune Systems Dendritic Cell Algorithm Cyber-Physical Production Systems Anomaly Detection

1. Introduction

Industry 4.0 (I4.0) is a highly appealing paradigm established originally in Germany and spread throughout Europe. It is characterized as the merger of the Internet and adaptable items ("smart" [1] machinery and goods) with existing digitized industry. Access to the Internet through Cyber-Physical Production Systems (CPPS) is one of the primary facilitators of I4.0, and this combination is commonly referred to as the Industrial Internet of Things (IIoT). The use of the Internet of Things (IoT) in I4.0 enables devices to be more flexible in how they interface and behave, as well as the integration of significantly more complex architectures in smart and autonomic industrial scenarios, such as components that process massive amounts of data (Big Data), and distributed computation in the Cloud/Edge layer.

This synergy among technologies enables CPPS components in I4.0 to be autonomic since components can organize themselves in a network based on the context in which they are inserted, can monitor their environment, assess potential problems, and adapt, resulting in a decentralized system [1]. As time passes, CPPS tend to have more and more devices linked in a network, greater Internet access, and less human involvement [2], as CPPS is often viewed as a vital element in the notion of IIoT and the Fourth Industrial Revolution.

At their foundation, CPPS are a combination of computing and physical processes that are aligned with communication networks [3]. While typical embedded systems integrate computing with real-world interactions, CPPS place a particular emphasis on inter-device communication, according to Jazdi [3]. They are frequently integrated into feedback loops in which the cyber component (control, communication and computation in general) and physical component (the physical process, measurements by sensors and actuator impacts) are tightly coupled, just as with embedded systems, but with the added component of subsystem interactions due to their ability to communicate with each other. This increases the complexity and dynamism of such systems, which are frequently challenging to manage in centralized and prescribed techniques.

A CPPS is thought to be a safety-critical system in many of its application scenarios, where a variation in conventional behavior might result in substantial economic losses and material damage, if not human lives. Over the years, several attacks have tackled Industrial Control Systems (ICS), which had a considerable impact on the physical processes. Some known examples are the *Stuxnet worm* [4], the *Ukraine SCADA attack* [5], the *Mirai attack* [6], Maroochy Water Services in Australia (2000), German Steel Mill Thyssenkrupp (2014), and Norsk Hydro Aluminium (2019) [7].

To address this problem, designing robust CPPS is critical, mainly when under equipment faults, caused by attacks or not. Furthermore, due to the rise of automation and system dynamics, increased complexity makes assaults and naturally occurring errors more difficult to detect, analyze, and resolve by a human operator. This motivates the need for CPPS to identify such anomalies and take relevant action. Furthermore, since the reaction to mitigate the impact of anomalies should be fast, CPPS have real-time needs [8].

2. Anomaly Detection

Anomaly detection refers to algorithms that analyze data to identify anomalies or outliers. An anomaly may appear gradually in manufacturing processes, such as tool/wheel wear. It might be sudden, such as tool breakage, or it could be avoidable, such as excessive vibration/chatter. So, process anomaly detection is critical since knowledge of tool wear is required for scheduling tool changes, detection of tool breakage is required for salvaging the workpiece and the machine, and recognizing chatter is required for initiating remedial action [9].

An anomaly detection algorithm can be trained using previous information or typical operating circumstances. This is why process monitoring is so important, which refers to the capture, manipulation, and analysis of sensor readings to identify the condition of a process. In either instance, a profile of what is to be regarded as usual is constructed. The algorithm then accepts input data and determines whether it is consistent with its normal behavior or an outlier. A process set of variables (e.g., force, speed, motion, temperature, pressure, power, acoustic emission, feed motor current, and so on) is measured, processed online, and compared to their predicted values for this purpose. Any variations from predicted values are blamed on process anomalies [10].

A process anomaly is a deviation from normal process behavior linked to various sources. It is defined as an unallowable divergence of at least one system characteristic property (feature) from the accepted or normal operating state. Facing an anomaly, if no recovery decisions are made, it may result in malfunctions or failures, which are long-term disruptions in a system’s capacity or performance to fulfill a necessary function under specified operating parameters.

Anomalies can express themselves in three ways, according to Chandola et al. [11]:

- Point anomaly: this is the most basic situation in which a single data instance is abnormal compared to the rest—for example, if a data instance has a greater value than those of all other data instances.
- Contextual anomaly: a data instance that is out of place in its context. The definition of context varies depending on the situation. A recorded average yearly temperature of 9 °C in the Amazon rainforest, for example, can be considered as a contextual anomaly because it is abnormal in the context in which it is recorded (here, the context is longitude and latitude). However, such an average yearly temperature would not be anomalous in all contexts (e.g., Europe).
- Collective anomaly: this occurs when a connected data group is deviant compared to the total dataset. For example, suppose a periodic behavior that repeats for every constant time interval abruptly changes for a single period. In this case, that period alone is not anomalous, nor is any point in the context deviating from the sequence. However, the entire period behaves inconsistently when compared to surrounding data points, despite possibly not having any significantly high values or a high rate of change.

As previously stated, anomaly detection seeks to find data items that are out of the ordinary compared to “normal” ones. The first issue is defining what is normal and abnormal in a way that is objective and calculable. This definition changes according to the technique adopted by a particular algorithm. There are three primary techniques to establishing what defines an anomaly: (i) distance based: outliers are defined by their distance from other data points; (ii) density based: outliers are data points that are found in lower density locations; (iii) rank based: outliers are data points whose nearest neighbors have other data points as nearest neighbors.

Furthermore, anomaly detection algorithms can be further classified as hard computing based and soft computing based. Hard computing includes probabilistic learning, knowledge, and ML-based techniques, while soft computing includes branches such as fuzzy logic, genetic algorithms, ant colony optimization, and artificial immune systems. Finally, there are combination learners, which include hybrid and ensemble-based algorithms.

Data have a known distribution and parameters in parametric anomaly detection (e.g., normal distribution characterized by mean and standard deviation). On the other hand, prior knowledge of data may be known in non-parametric techniques, but their distribution is unknown. This can be accomplished using: (i) a supervised algorithm, in which all data points used have a known classification through the use of labels (e.g., “anomalous” and “normal”); (ii) an unsupervised algorithm, in which data points have no labeling and data patterns and relationships are learned; (iii) a semi-supervised algorithm, by employing a small amount of labeled data and applying that knowledge to attempt learning on additional, unlabeled data, where the latter constitutes the majority of all data, which can be accomplished by applying what was learned in a supervised setting, for example, to label the unlabeled data.

Finally, the application environment in which anomaly detection happens can be online or offline. Offline anomaly detection refers to the examination of comprehensive and collected data at any point in the past, with loose limits on the methods employed since they can be more complicated and do not need to provide rapid reaction times. On the other hand, online anomaly detection is either real-time or near real time and is frequently coupled with data streaming, resulting in a continuous process of data gathering and anomaly detection.

2.1. Anomaly Detection in CPPS

As previously said, the I4.0 paradigm emergence has some difficulties. Using anomaly detection in this context entails identifying essential components of these issues and the standard anomaly detection in time series problems. These issues vary depending on the unique context of industrial application, but a good anomaly detection technique must include the following features:

- Generalization: It is impossible to forecast the kinds of anomalies that will arise and how they will emerge in a time series. For example, if a time series is highly periodic, techniques that account for seasonal trends will do very well in identifying deviations. Nevertheless, the same strategy would perform poorly in unstable time series. A decent anomaly detection technique must be consistently successful in an agnostic way to time series typical behavior and presume that the intended behavior is dynamic.

- Time performance: In industrial applications, anomaly detection must be discovered within usable time, since late detection can result in massive losses and jeopardize safety.

- Resource efficiency: According to the I4.0 concept, CPPS components are anticipated to be self-monitoring. As a result, components must execute anomaly detection decentralized, which raises issues about resource utilization. These components are frequently resource constrained, both in themselves and in their use of network infrastructures (i.e., communication itself must be efficient).

- Little need for labeled data: There is a scarcity of publicly accessible datasets for industrial settings under assault. Many techniques to appropriate learning need massive volumes of data, which are not feasible in this scenario. Furthermore, depending on the labeled data, techniques may eventually have difficulties adapting effectively if the system since the model would overfit the data from which it has learned, which would be contrary to the feature indicated in the first bullet point.

- Robustness and safety: Components must be able to adapt to unfavorable situations by continuing regular operation without jeopardizing the equipment’s or human life’s safety.

These difficulties are not trivial, and to the best of knowledge, no solution seeks to handle all of them simultaneously. Instead, they focus on a subset of them most of the time. Some of these challenges motivate the design and development of AIS applied to anomaly detection.

2.2. AIS for Anomaly Detection

According to Tokarev et al. [12], AIS is one of the most promising current solutions to the problem of anomaly detection. These approaches offer a great degree of generality, are usually computationally efficient, and consume fewer resources. Costa Silva et al. [13] defend that the use of immune-inspired approaches for anomaly detection has been adopted in the literature because of its analogy with body resistance in the human immune system provided against agents which causes diseases. According to them, the applicability of AIS in anomaly detection consists mainly of the self–nonself discrimination and the danger models, which refer to the NSA [14] and DCA [15][16].

On the one hand, the NSA is a straightforward and intuitive algorithm for anomaly detection that analyzes the feature space and generates detectors in the nonself region. Because the self data are utilized as a reference, the detectors are positioned outside the self zone. This means that the overall process is similar to supervised and semi-supervised machine learning approaches. NSA has significant problems concerning the system context. Additionally, it scales poorly with data size, as detectors frequently overlap and leave gaps between them, failing to identify several anomalies. Finally, the process to measure the similarity between detectors and data can be pretty expensive.

On the other hand, the DCA is the most notorious danger model-based algorithm, and it relies significantly on the correlation mechanism between the *antigen* and input signals (behavior of the system). Compared with NSA, the DCA is super lightweight and eliminates the need to define normal (self data). However, it requires prior knowledge or modeling of the problem. Otherwise, input signals and *antigens* are not modeled correctly. Consequently, the pre-processing data stage is critical. In some contexts, to calculate the *safe* and *danger* signals, it is necessary to distinguish between self and nonself data. So, despite resembling an unsupervised machine learning approach, there is a strong dependency on the existing expert knowledge about the context of the application. Moreover, in the classification stage, the algorithm requires the entire *DC* population for cell maturation and overall *antigen* classification, gather then having only a particular data point to mature before categorizing said data point. This makes the algorithm unsuitable for online anomaly detection as it is.

In their work, Costa Silva et al. [13] compared both models using the *UCI Breast Cancer Database*, and reached the conclusion that there is no better immune-inspired model than another in terms of anomaly detection. Depending on the context of application and the problem definition, there are cases in which the DCA may overcome the NSA and vice versa. So, considering the features about anomaly detection in CPPS mentioned before, in theory, DCA is more suitable compared with NSA, since it is more lightweight, and in consequence, resource efficient, and was less dependent on labeled data. However, the limitations regarding the overall algorithm's data structures and procedural operations understanding, online application, and initial problem modeling leave room for improvement if this algorithm is intended to be used for anomaly detection in industrial scenarios.

Regarding the overall algorithm's understanding, Gu et al. [17] proposed a formal mathematical definition of the deterministic Dendritic Cell Algorithm (dDCA). This work helped the research community understand the algorithm and avoid ambiguities since previous investigations focused mainly on empirical aspects. Later, Greensmith and

Gale [18] proposed a specification of the dDCA using functional programming (in Haskell programming language), which on the contrary to the previous work, is targeted to a computer science audience.

Considering the limitation of applying DCA in online classification problems, such as anomaly/intrusion detection, Gu et al. [19] proposed a real-time or near real-time analysis component by applying segmentation to the current output of the DCA into slices. The researchers tested the proposed approach in the *SYN scan* dataset, a large real-world dataset based on a medium-scale port-scan of a university computer network. The results show that segmentation applies to the DCA for real-time analysis since, in some cases, segmentation produces better results when compared with the standard algorithm. Later, Yuan and Chen [20] also proposed a real-time analysis to the DCA, where an *antigen* presented by sufficient dendritic cells is immediately assessed. The researchers validated the proposed approach in the *UCI Wisconsin Breast Cancer* dataset. The results are promising, showing that the algorithm achieves considerable accuracy.

More recently, Pinto et al. proposed the incremental Dendritic Cell Algorithm (iDCA) [21] as a method to detect network intrusions in a real-time industrial Machine-to-Machine (M2M) scenario. For validation of the iDCA, two network intrusion detection datasets were used in an online classification scenario, namely, the *KDD Cup99* and the *M2M using OPC UA* datasets. The results show that the approach is a viable solution to detect anomalies in (near) real-time, especially in environments with little a priori system knowledge for intrusion detection.

As for problem modeling in the DCA, as mentioned before, the DCA avoids a model training step, but it requires domain or expert knowledge for data pre-processing. Gu et al. [22] attempted to use the Principal Component Analysis (PCA) technique to categorize input data in DCA automatically. This way, it is possible to avoid manually over-fit the data to the algorithm every time a new problem context is considered, which is undesirable. Experimental results were performed in the *Stress Recognition in Automobile Drivers* dataset and have shown that the application of PCA to the DCA for automated data pre-processing is successful. Later, Chelly and Elouedi [23] reviewed the pre-processing data phase of the DCA while making a comparative study of data reduction techniques within the DCA, based mainly on Rough Set Theory (RST) usage in the pre-processing phase. The researchers used multiple real-valued and binary-classification datasets. The results show that different DCA versions outperform known Machine Learning classifiers in the literature, namely Support Vector Machine (SVM), Artificial Neural Network (ANN), and Decision Tree (DTree), in terms of the overall classification accuracy.

3. Cursory Dendritic Cell Algorithm

The development of the CDCA aims at solving or reducing the impact of the main limitations of DCA identified earlier. Thus, in this work, three different contributions are proposed. Firstly, researchers propose an object-oriented specification (using Python) to abstract the algorithm into its main mechanisms and decompose it into a more modular structure, such that specific components of the algorithm can be updated more independently. Secondly, researchers propose a near-real-time analysis component, which allows the algorithm to present intermediate classification results to data online, in contrast with its application to entire sets of data. Thirdly, researchers enable data-driven approaches to extract signals in the pre-processing stage of the DCA. The problem modeling process will not require as much expert knowledge as currently.

3.1. Modular Dendritic Cell Algorithm

DCA specification and implementation, using an object-oriented approach, was motivated by the need to adjust and update the algorithm’s source code. With an object-oriented implementation, experimentation and changes to the original algorithm become easier to develop and test.

The decomposition of the DCA presented was carried out by identifying the main steps of the algorithm and abstracting those steps into separate classes (modules). The rationale for each class is according to the following steps: (i) the *antigens* and values are received by the system; (ii) their signals are extracted; (iii) the extracted signals are distributed into the *DC* population; (iv) each *DC* decides how it processes these *antigens* and signals; and (v) the *antigens* are somehow impacted by this processing. Considering this rationale, the following classes are defined:

- **Interface**: to keep the usage of the algorithm simple, a single interface is used to interact with the system as a whole. Therefore, the interface can receive *antigens*, and output *antigen* classification results. Indeed, the *Interface*, as implemented, orchestrates the remaining modules.

- **Signal Extractor**: defines how signals are extracted from data. It receives *antigens* and associated data, while outputs *safe*, *danger*, *PAMP*, and inflammation signals from received data and the *antigen* they are associated with.

- **Sampler**: decides on a strategy for signals and *antigen* distribution in the *DC* population, i.e., which *DCs* are attributed to which *antigens* and signals. It receives the *antigens* and signals from the *Signal Extractor* and chooses specific *DCs* to receive them.

- **DC Population**: the *DCs* themselves can receive *antigens* and signals. Received *antigens* are stored locally by each *DC*. Each *DC* also has access to the *antigen* repertoire (see next module), allowing the collected *antigen*’s state to be updated by the *DC* that collects it.

- **Antigen Repertoire**: a database structure holding all *Antigens* that are given to the system. It might be relevant to store in the *Antigen Repertoire* changes in the *antigens*’ state, for instance, when a *DC* migrates, or store information about which mature/semi-mature *DC* is associated with a given *antigen* collected. The *Interface* can query the *Antigen Repertoire* to achieve classification.

Each component can be updated separately with this modular approach, and different strategies in different modules can be implemented.

3.2. Online Dendritic Cell Algorithm

As previously mentioned, DCA does not support an online anomaly detection capability. The DCA can internally process input data in real-time, but it requires an analysis stage, which is performed offline. To reduce the impact of this limitation, the DCA is updated to allow intermediate classification of data and thus, enable near real-time anomaly detection.

The classification of *antigens* is accomplished at the end of the algorithm's data-flow, by calculating the *MCAV* value, which consists of the ratio of matured *DCs* that collect a given *antigen*, to the total number of times an *antigen* is collected. Thus, the classification method was updated in two ways: (i) each time a *DC* matures, all *antigens* it has collected have their *MCAV* values derived; (ii) each *antigen* has three possible states: inconclusive, safe, or dangerous.

All *antigens* begin as inconclusive. If their *MCAV* value increases beyond the predetermined threshold, the *antigen* is immediately classified as dangerous, i.e., anomalous. If all *DCs* that collected the *antigen* have matured, but the *MCAV* value has not yet increased beyond the predetermined threshold, the *antigen* is classified as safe. The introduction of an "inconclusive" state leads to a lack of immediate classification. Classification will only occur after a sufficient number of *DCs* have migrated. On the other hand, classification of "dangerous" *antigens* is achieved faster, as this classification is attributed to the *antigen* as soon as the *MCAV* increases above the threshold, rather than waiting for the entire data set to be processed. The classification of "safe" *antigens* is also faster than waiting for the entire dataset but is slower than the classification of "dangerous" *antigens*.

3.3. Data-Driven Dendritic Cell Algorithm

Regarding the difficulty of problem modeling, considering the dependency of previous expert knowledge about the problem and application context, the pre-processing stage of the DCA was updated to become more data-driven. This difficulty pertains mainly to the signal extraction procedure, as the DCA does not define how signals are extracted. Thus, a data-driven approach to signal extraction is suitable for solving this limitation. For this, it was considered K-Means, as a scalable and inductive clustering algorithm, since it checks essential requirements, given the context of CPPS, such as:

- Suitable for multi-dimensional data processing;
- Able to process new, unseen, data;
- Usable in an incremental, per-point basis;
- Provides a real-valued score suitable for usage as signals;
- Scalable with data dimension and size;
- Provides real-time processing, such that its usage for online anomaly detection is not prohibitive.

K-Means is an algorithm that finds centroids for each cluster of data such that the variance in each cluster is minimized. The number of clusters is user defined. Therefore, each cluster can be represented in a single point which constitutes the centroid, and future points can be included in clusters according to their distance to each cluster's centroid.

In order to learn what is normal, K-Means first computes the centroids of normal data only. Once this process is completed, the centroids of data are stored, and the maximum distance for each centroid to the farthest data point that belongs to its cluster is also stored. These distances will be used as radii, forming a hypersphere per cluster. Future data points to be classified are compared to these hyperspheres, measuring their distance to the hypersphere's surface. If the data point is inside the hypersphere, the distance to its surface constitutes a *safe* signal. Otherwise, its distance constitutes a *danger* signal. Furthermore, if new normal behavior occurs, more hyperspheres can be produced.

3.4. Proposal Overview

The online adaptation was implemented in the *DC Population* and in the *Antigen Repertoire* modules. In the former, *DCs* directly notify *antigens* of their collection and maturation, and in the latter, *MCAV* values are updated incrementally. Regarding the data-driven adaptation, the *Signal Extractor* module is addressed, such that the K-Means approach is used for the extraction of *danger* and *safe* signals from data.

Considering the contributions described previously, the overall CDCA works as such:

- Training phase:

  (i)The *Interface* receives a predefined number of normal data points, one by one, which will be used as a baseline for normal behavior.
  (ii)The *Signal Extractor* receives these data points, one by one, and applies a standard K-Means approach for clustering the data points. Note that, at this stage, a classification is not possible since there is not yet a normal baseline.
  (iii)Eventually, when sufficient training points are received, the *Signal Extractor* computes each cluster's maximum distance from its centroid to each point considered in the cluster, which will serve as radii for each cluster's hypersphere. The centroids and radii are stored, but the training data points are not.
- Anomaly Detection phase:

  (i)The *Interface* receives each data point, i.e., an *antigen*, one at a time.
  (ii)The *Signal Extractor* receives each point, and outputs its *danger* and *safe* signals, depending on whether they are inside a hypersphere or not, and how far they are from the closest hypersphere surface. *PAMP* and Inflammation signals are not used in CDCA.
  (iii)Upon receiving the *antigen* and resulting signals, the *Sampler* samples *DCs* in a sliding window approach (n-contiguous cells).

(iv)Each *DC* receives the signals and computes their costimulatory value (used for migration) and k value (used to decide whether they migrate as mature or semi-mature). Upon receiving an *antigen*, each *DC* notifies the respective *antigen* in the *Antigen Repertoire* of its collection, incrementing its record of the number of times collected.

(v)Over time, *DCs* mature, and notify their *antigens* of the times they matured as mature *DC*. *DCs* that mature as semi-mature also notify *antigens*.

(vi)When *antigens* in the *Antigen Repertoire* achieve high *MCAVs*, they notify the *Interface* of their state change, from “inconclusive” to “dangerous”, i.r., an anomaly has been detected. When *antigens* in the *Antigen Repertoire* have the sum of *mDC* and *smDC* to be equal to the times they have been collected, their state changes from “inconclusive” to “safe”, i.e., the data point is considered normal. *Antigens* that have notified the *Interface* are removed from the *Antigen Repertoire*.

References

1. Lasi, H.; Fettke, P.; Kemper, H.G.; Feld, T.; Hoffmann, M. Industry 4.0. Bus. Inf. Syst. Eng. 2014, 6, 239–242.
2. Xu, H.; Yu, W.; Griffith, D.; Golmie, N. A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective. IEEE Access 2018, 6, 78238–78259.
3. Jazdi, N. Cyber physical systems in the context of Industry 4.0. In Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–24 May 2014; pp. 1–4.
4. Sharma, M.; Elmiligi, H.; Gebali, F. Network Security and Privacy Evaluation Scheme for Cyber Physical Systems (CPS). In Handbook of Big Data Privacy; Springer International Publishing: Cham, Switzerland, 2020; pp. 191–217.
5. Langner, R. Stuxnet: Dissecting a Cyberwarfare Weapon. IEEE Secur. Priv. 2011, 9, 49–51.
6. Sinanović, H.; Mrdovic, S. Analysis of Mirai malicious software. In Proceedings of the 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 21–23 September 2017; pp. 1–5.
7. Oueslati, N.E.; Mrabet, H.; Jemai, A.; Alhomoud, A. Comparative Study of the Common Cyber-physical Attacks in Industry 4.0. In Proceedings of the 2019 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC), Tunis, Tunisia, 20–22 December 2019; pp. 1–7.
8. Mitchell, R.; Chen, I.R. A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. ACM Comput. Surv. 2014, 46, 55:1–55:29.
9. Ulsoy, A.G. Monitoring and Control of Machining. In Condition Monitoring and Control for Intelligent Manufacturing; Springer: London, UK, 2006; pp. 1–32.

10. Bayar, N.; Darmoul, S.; Hajri-Gabouj, S.; Pierreval, H. Fault detection, diagnosis and recovery using Artificial Immune Systems: A review. Eng. Appl. Artif. Intell. 2015, 46, 43–57.
11. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly Detection: A Survey. ACM Comput. Surv. 2009, 41, 1–58.
12. Tokarev, V.; Sychugov, A.; Anchishkin, A. Detection of Anomalies in the Information Networks of Industrial Automation Systems Based on Artificial Immune Detectors. In Proceedings of the 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 8–14 September 2019; pp. 1–5.
13. Costa Silva, G.; Palhares, R.M.; Caminhas, W.M. A Transitional View of Immune Inspired Techniques for Anomaly Detection. In Intelligent Data Engineering and Automated Learning—IDEAL 2012; Yin, H., Costa, J.A.F., Barreto, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 568–577.
14. Textor, J. A Comparative Study of Negative Selection Based Anomaly Detection in Sequence Data. In Artificial Immune Systems; Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 28–41.
15. Greensmith, J.; Twycross, J.; Aickelin, U. Dendritic Cells for Anomaly Detection. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 664–671.
16. Greensmith, J.; Aickelin, U.; Tedesco, G. Information fusion for anomaly detection with the dendritic cell algorithm. Inf. Fusion 2010, 11, 21–34.
17. Gu, F.; Greensmith, J.; Aickelin, U. Theoretical formulation and analysis of the deterministic dendritic cell algorithm. Biosystems 2013, 111, 127–135.
18. Greensmith, J.; Gale, M.B. The Functional Dendritic Cell Algorithm: A formal specification with Haskell. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1787–1794.
19. Gu, F.; Greensmith, J.; Aickelin, U. Integrating Real-Time Analysis with the Dendritic Cell Algorithm through Segmentation. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, Montreal, QC, Canada, 8–12 July 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 1203–1210.
20. Yuan, S.; Chen, Q.j. A Dendritic Cell Algorithm for real-time anomaly detection. In Proceedings of the 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), Zhangjiajie, China, 25–27 May 2012; Volume 1, pp. 448–451.
21. Pinto, R.; Gonçalves, G.; Delsing, J.; Tovar, E. Incremental Dendritic Cell Algorithm for Intrusion Detection in Cyber-Physical Production Systems. In Intelligent Computing; Arai, K., Ed.; Springer International Publishing: Cham, Switzerland, 2021; pp. 664–680.

22. Gu, F.; Greensmith, J.; Oates, R.; Aickelin, U. Pca 4 dca: The application of principal component analysis to the dendritic cell algorithm. SSRN Electron. J. 2009.

23. Chelly, Z.; Elouedi, Z. A study of the data pre-processing module of the dendritic cell evolutionary algorithm. In Proceedings of the 2014 International Conference on Control, Decision and Information Technologies (CoDIT), Metz, France, 3–5 November 2014; pp. 634–639.