International Journal of Innovative Research and Scientific Studies, 9(3) 2026, pages: 44-51

IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

Markov-based algorithms for wireless sensor network: Theoretical insights and python implementation

Kian Meng Yap[1,3*], Huang Shen Chua[2], Jiehan Teoh[1], NG WEI JIANG[4]

[1]*Research Centre for Human-Machine Collaboration, School of Computing and Artificial Intelligence, Faculty of Engineering and Technology, Sunway University, Bandar Sunway, 47500 Petaling Jaya, Malaysia.*
[2,4]*Faculty of Artificial Intelligence and Frontier Technologies (FAiFT), UNITAR International University, Kelana Jaya, Selangor, Malaysia.*
[3]*Department of Computing and Information Systems, Sunway University, Selangor 47500, Malaysia.*
[4]*Auco Venture SDN BHD, Lot 1115 Kampung Ribu Rantau, Negeri Sembilan, 71200, Malaysia.*

Corresponding author: Kian Meng Yap (*Email: kmyap@sunway.edu.my*)

Abstract

The study concentrates on improving the dependability and operational efficacy of LoRa-based Wireless Sensor Networks (WSNs), which are extensively utilized in IoT applications, especially for long-range private networks. It seeks to deal with the problems that arise when a single node or communication line fails, which can have a big effect on network performance. The research utilizes a Markovian matrix theoretical framework to examine and simulate the behavior of LoRa-based Wireless Sensor Networks (WSNs), incorporating states such as Sleep (S), Idle (I), Transmit (T), and Receive (R) mode. A Python software program was created to put this model into action, allowing for testing and simulation with 50 fake data sets. The method stresses that the network should always be running, that sensor nodes should be replaced quickly, and that the network should be able to handle failures of individual nodes. The simulations indicate that using the Markov chain model in conjunction with detailed step-by-step math computation may yield a more accurate analysis of the data sets. The methodology also helps you evaluate protocols, change control, look at scalability, and make informed choices about how to build a network. This work offers practical benefits for the design, deployment, and maintenance of LoRa-based WSNs in real-world IoT scenarios. It supports network administrators and engineers in predicting power consumption, designing resilient protocols, scaling networks efficiently, and implementing adaptive control measures to ensure continuous and dependable operation. The integration of Markov chain mathematical modeling with Python-based simulation provides a robust solution for ensuring reliable operation of LoRa-based WSNs. The approach mitigates the impact of node failures, supports rapid recovery, and maintains network integrity.

Keywords: Energy efficiency, Markov's chains, Python, Wireless sensor networks.

DOI: 10.53894/ijirss.v9i3.11333
Funding: The research is supported by the Ministry of Higher Education Fundamental Research (Grant Number: (FRGS) FRGS/1/2021/ICT02/SYUC/02/1).
History: Received: 22 December 2025 / **Revised:** 11 February 2026 / **Accepted:** 16 February 2026 / **Published:** 6 March 2026

Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Transparency: The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.
Publisher: Innovative Research Publishing

1. Introduction

A Long Range (LoRa) wireless module can establish a wide-area network (WAN) [1]. It has become a LoRaWAN. The LoRa wireless module is a very low-power, long-range transmission device that receives incoming data and sends it wirelessly [2]. As a result, smart wireless sensor networks are rapidly evolving in many applications such as monitor agriculture activities, home security and surveillance systems, home appliances, military, education, medical treatments, etc. Long Range (LoRa) wireless technology works with sensors to create hundreds or thousands of sensor nodes. This makes wireless communication and scalable machine-to-machine connectivity more efficient [3]. As industry transitions from Industry 4.0 to Industry 5.0, Industrial Wireless Sensor Networks (IWSN) have been used a lot in smart factories' real-time monitoring and control systems. Sensors are electronic devices that monitors, measures, or detects any physical process or change [2]. Therefore, the system not only transmission data, a fault detection and fault-tolerance (FT) mechanisms are embedded into communication protocols. The Markov model is frequently used in many areas of wireless sensor networks (WSNs) because it is capable of modelling behavior processes and use current information to make predictions about future states. Therefore, it is able to use past data to model the system's failure states. The common applications are energy management and lifetime prediction [4] routing and data transmission [5]. MAC layer protocols [6] security and intrusion detection [7] node mobility and localization [8]. Fault Detection and Network Reliability [9] smart farming [10]. Forecasting discharge and others. The Markov Chain is straightforward to understand, easy to use, and good for predicting things that are either category or state-based. Most researchers merely show the basic math equations and don't explain how to understand or interpret the stages involved in the calculations. Because of this lack of clear instructions, it's hard for readers to understand how the calculations work, repeat the results, or use the same procedure on other problems. This paper provides a concise, step-by-step explanation of a mathematical system that changes from one state to another based on certain probability rules.

2. Overview of Wireless Sensor Networks (WSNs)

Wireless Sensor Networks (WSNs) Matin and Islam [11] are a way to collect data about the environment, like temperature, sound, vibration, pressure, motion, gases, and more, and transfer it to a central point where it can be evaluated. A sensor node has a wireless module and has sensing and computing devices, radio transceivers, and power components. All of the wireless sensor nodes will set up a network communication topology to make a wireless sensor network [12]. Figure 1 demonstrates the design and configuration of a wireless sensor network system for monitoring plant conditions.

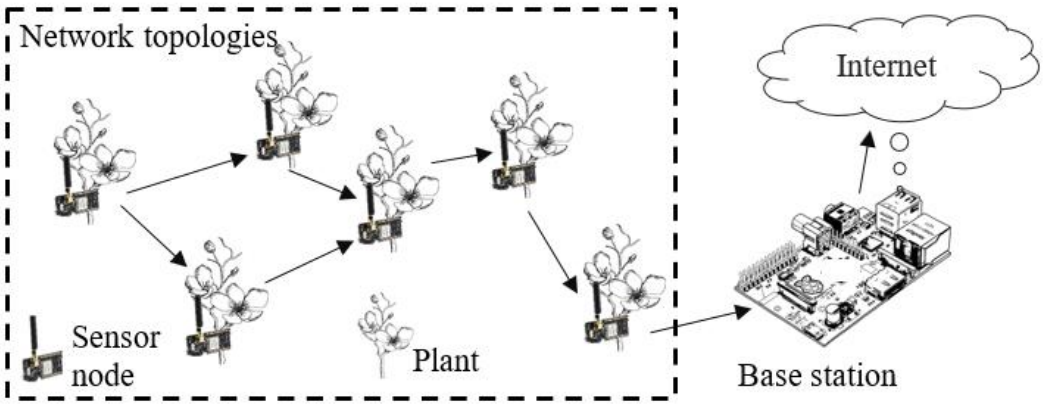

Figure 1.
a wireless sensor network architecture system.

The sensors are install across the plant environment to keep a data collection on like temperature, humidity, soil moisture, light intensity, pressure, and the status of the equipment. There get real-time data and send it wirelessly to a

central control system or cloud platform, so there is no need for complicated wiring. Wireless sensor networks aid with precision farming by keeping an eye on soil moisture, nutrient levels, and weather conditions. The data that was collected helps people make smart decisions about watering, fertilizing, and controlling pests, which results to higher agricultural yields and better use of resources. A base station (or sink node) is the most important component of a cluster that collects, controls, and sends data from other sensor nodes. A base station (BS) can send signals over a longer distance than other nodes. The BS might also be a raspberry pi node or active node that acts as an information sink, or it could be anything else that takes data from the sensor network and sends it to the internet. One common technique to use wireless sensor networks to continue to keep monitoring plants is to connect from a LoRa to another LoRa module to a Raspberry Pi [13]. The Dragino LoRa HAT is a popular LoRa module that may be placed directly on Raspberry Pi boards, for instance. This design lets sensor nodes and a central gateway talk to each other over vast distances with little power, which is good for farming. But for the small capacity of the plant, an EBYTE E32-915T20D can be connected to a Raspberry Pi Pico for low-power long-range environmental monitoring.

3. Theoretical Insights into Markov Models

In the Markov prediction model, the conditions of an event are defined by its state at a specified time. The likelihood of moving to the next state is called the state transfer probability. Each state can potentially shift to another state, allowing us to create a transfer probability matrix using the following matrix Equation 1:

$$P = \begin{bmatrix} P_{11.} & P_{12.} & \cdots & P_{1N.} \\ P_{21.} & P_{22.} & \cdots & P_{2N.} \\ P_{31.} & P_{32.} & \cdots & P_{3N.} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1.} & P_{N2.} & \cdots & P_{NN.} \end{bmatrix} \quad (1)$$

The transition matrix for a Markov chain is defined as a $N \times N$ matrix. Each element of this matrix, denoted as (i, j), is equal to entry $P_{ij} = p[i, j]$.. The matrix P is defined as:

$$\mathbf{0} \leq P_{ij} \leq \mathbf{1}, \mathbf{1} \leq i, j \leq \boldsymbol{N} \quad (2)$$

$$\textstyle\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} P_{ij} = 1, \mathbf{1} \leq i \leq \boldsymbol{N} \quad (3)$$

Whereby P_{ij}, represents the probability of transitioning from $current\ state$ (i) to $next\ state$ (j). The notation $0 \leq P_{ij} \leq 1$ means that every entry in the matrix is a probability and must be between 0 and 1 (inclusive). , $1 \leq i, j \leq N$ indicates that (i) and (j) range over all possible states, where (N) is the total number of states in the Markov chain. From equation 3, This equation means that if you sum all the probabilities of moving from state (i) to every possible state (j), the total must be 1.

These transition system processes were applied in many applications, including resource allocation Amine, et al. [14] channel state prediction [15] traffic management [16] prediction of lifetime model [17] optimisation of sensor network [18]. Energy Efficiency [17] and others. They are all connected by these transitions. To explain the matrix Markov theory, The examples concepts and operations from the work of El Fawal, et al. [18] and Nurgaliyev, et al. [19] were used. The computation example uses the Markov model to describe the operational states of LoRa sensor nodes. These states include Sleep (S), Idle (I), Transmit (T), and Receive (R). Every change requires both time and power energy. The statuses of LoRa can be checked using AT commands [20, 21]. A Markov Transition Matrix gives a mathematical approach to describe how frequently and how consistently these shifts between states happen over time. Let make a 4×4 Markov Transition Probability Matrix for the LoRa node with the 4 states: State 1 = S (Sleep), State 2 = I (Idle), State 3 = T (Transmit), and State 4 = R (Receive). During this observation period, the changes between states are recorded. The total number of transitions from each state to every other state is then counted and put into a count matrix, which is called C. Each element, C_{ij}, in this matrix represents how often the node changes from state (i) to state (j). After gathering the transition data, each row of the C matrix is normalised. This is done by dividing each count by the total of the counts in that row, which changes the raw counts into probabilities. The resulting matrix is the Markov Transition Probability Matrix. Each row in this matrix shows the probability of moving from one state to another, and the total of each row always equals one. The Markov chain state transition is simply divided into the following phases for computing the matrix probabilities:

1) Read and record the incoming observed sequence of the system states (S, I, T, R).
   It predefines S (Sleep) = 0, I (Idle) = 1, T (Transmit) = 2, and R (Receive) =3. The integers 0, 1, 0,1,2, and 3 represent the states.
2) Compute the count transition matrix (There are 16 transition states based on 4x4),
   $C_{ij}(C_{11.}\text{to } C_{44.})$.
   $C_{ij} = \text{Number of transitions from state i to j}$

$$C_{ij} = \sum_{t=0}^{T-2} 1\{N_t = i \ and \ N_{t+1} = j\} \quad (4)$$

Whereby $N_t \in (0,1,2,3)$[mapping S, I, T, and R to intergers]. $1(\cdot)$ is the indicator function

$$1\{\text{condition}\} = \begin{cases} 1, \text{if condition is true} \\ 0, \text{ if condition is false} \end{cases}$$

The build count matrix C_{ij} is shown in Table 1:

Table 1.
The sequence of states is [S, I, T, R], and the build count matrix C can be generated accordingly.

From \ To	S(0)	I(1)	T(2)	R(3)
S(0	C_{00}	C_{01}	C_{02}	C_{03}
I(1)	C_{10}	C_{11}	C_{12}	C_{13}
T(2)	C_{20}	C_{21}	C_{22}	C_{23}
R(3)	C_{30}	C_{31}	C_{32}	C_{33}

In the words, Each C_{ij} counts how many times it goes from I to j in the order.
3) Compute the ratio counting transition into a transition probability matrix.

$$P_{ij} = \frac{C_{ij}}{\sum_{k=0}^{N-1} C_{ik}} \quad (5)$$

Whereby C_{ij} is observed count transition matrix $i - j$ (The number of occurrences of transitions from i to j within the sequence.) $\sum_{k=0}^{N-1} C_{ik}$ is total sum count transition matrix corresponds from current state I to next state k (example : $S \to S$, $S \to I, S \to R, S \to R$) .

Therefore, N = 4, and the transition matrix PPP must be a 4×4 matrix. Let assume probabilities of the matrix be:

$$p = \begin{bmatrix} 0.80 & 0.20 & 0.00 & 0.00 \\ 0.10 & 0.60 & 0.20 & 0.10 \\ 0.00 & 0.90 & 0.10 & 0.00 \\ 0.10 & 0.80 & 0.00 & 0.10 \end{bmatrix} \quad (6)$$

The current state is represented by the rows, while the next state is represented by the columns. The order of the states is [S, I, T, R] and build count matrix C are listed in table 1.0. To understand how to calculate probabilities, the following key ideas are needed: Discrete-Time Markov Chains (DTMCs), Transition Probability Matrices, and Empirical Probability or Relative Frequency. To create a 4×4 Markov Transition Probability Matrix for a LoRa node with four states—Sleep (S), Idle (I), Transmit (T), and Receive (R)—the node's state changes must first be observed over a set duration.

The current probabilities in the matrix are a vital tool for analysing, predicting, and understanding the behaviour of systems that follow Markov processes, which helps us identify the state probability vector. The state probability vector is determined by equation 6.

$$\pi_t = [P(S), P(I), P(T), P(R)] \quad (7)$$

Then, following a single time step

$$\pi_{t+1} = \pi_t \times P \quad (8)$$

$$\pi_t = \begin{bmatrix} \pi_{S(t)} \\ \pi_{I(t)} \\ \pi_{T(t)} \\ \pi_{R(t)} \end{bmatrix} \quad (9)$$

Whereby, $\pi_t = [P(S), P(I), P(T), P(R)]$ is the probability vector at time t. P is the transition probability matrix. π_{t+1} is the probability vector at time $t + 1$.

In other words, π_t is determined by observing the states of the LoRa node system over a period of time and counting how often each state occurs. These counts are then converted into probabilities. Therefore, to create a new probability vector, all transition probabilities must be employed. The system's four states, represented as S, I, T, and R, are shown in Equation 6.

$$\pi_{t+1} = [\pi_s, \pi_I, \pi_T, \pi_R] \begin{bmatrix} 0.80 & 0.20 & 0.00 & 0.00 \\ 0.10 & 0.60 & 0.20 & 0.10 \\ 0.00 & 0.90 & 0.10 & 0.00 \\ 0.10 & 0.80 & 0.00 & 0.10 \end{bmatrix} \quad (10)$$

The calculation for each expanded element of the new vector is as follows:

$$\pi_{s(t+1)} = 0.80\pi_{s(t)} + 0.10\pi_{I(t)} + 0.00\pi_{T(t)} + 0.10\pi_{R(t)} \quad (11)$$
$$\pi_{I(t+1)} = 0.20\pi_{s(t)} + 0.60\pi_{I(t)} + 0.90\pi_{T(t)} + 0.80\pi_{R(t)} \quad (12)$$
$$\pi_{T(t+1)} = 0.00\pi_{s(t)} + 0.20\pi_{I(t)} + 0.10\pi_{T(t)} + 0.00\pi_{R(t)} \quad (13)$$
$$\pi_{R(t+1)} = 0.00\pi_{s(t)} + 0.10\pi_{I(t)} + 0.00\pi_{T(t)} + 0.10\pi_{R(t)} \quad (14)$$

where, $\pi_{s(t)} + \pi_{I(t)} + \pi_{T(t)} + \pi_{R(t)} = 1$.

When a row vector π_t is multiplied by the matrix $\boldsymbol{P}$, the j^{th} component of the resulting vector π_{t+1} is calculated as the weighted sum of the j column of $\boldsymbol{P}$. The weights used in this calculation are determined by the elements of the vector π_t is shown in Equation 11.

$$\pi_{j(t+1)} = \sum_{i \in (S,I,T,R)} \pi_{i(t)} P_{i,j} \quad (15)$$

These short-term state distributions can be utilized to understand or forecast how the LoRa system behaves.

4. Python Implementation of Markov-Based Algorithms

This study explores a four-state Markov model with states' S, I, T, and R. A python script has been developed to:

1. Assess a collected incoming sequence of system state transitions.
2. Calculate the transition count matrix C_{ij}.
3. Transform the counts into a transition probability matrix P_{ij}.

The Figure 2 presents a Markov transition matrix calculation.

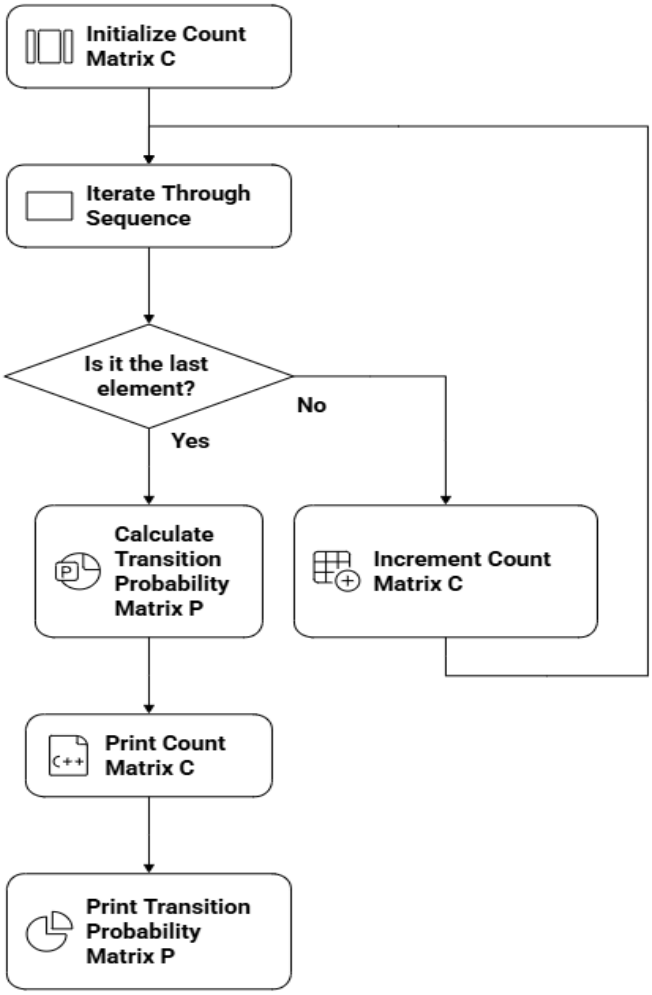

Figure 2.
Markov transition matrix calculation.

The analysis flow chart was done in Python concept to figure out the state transition and transition probability matrices from a series of discrete states. First, the series of observed states was set up, with each number standing for a different state. The total number of possible states was set, and a square count matrix was set up with zeros to keep track of transitions between states. The algorithm went through the sequence over and over again, and for each pair of consecutive states, it added one to the count matrix element that matched that state. This counted how many times the system went from one state to another. After this iteration was finished, the row-wise sums of the count matrix were used to find out how many transitions came from each state. Then, each row of the count matrix was normalized by its sum to create the transition probability matrix. This matrix shows the empirical likelihood of moving from one state to another based on the data that was collected. This method offers a simple and repeatable way to measure how transitions happen in systems with discrete states. Figure 3 shows Markov-Based Algorithms that are based on the theory. This algorithm computes a probability distribution across 50 predefined variable sample sets each time it executes.

```python
import numpy as np
import time
seq = [0,0,1,1,2,2,3,1,0,0,
1,2,3,3,1,0,2,2,3,1,
0,1,1,2,3,0,0,1,2,3,
1,0,2,2,3,3,1,0,1,2,
3,0,0,1,2,3,1,0,2,3]

n_states = 4
C = np.zeros((n_states,n_states), dtype=int)

print (C)
time.sleep(0.5)

for a,b in zip(seq[:-1], seq[1:]):
    C[a,b] += 1
    print( "a is {aa} and b is {bb}." .format(aa=a,bb=b))
    print (C)

P = C / C.sum(axis=1, keepdims=True)
print("Count matrix:\n", C)
print("Transition probability matrix:\n", P)
```

Figure 3.
Python source codes of Markov transition matrix.

Programming from line 1 and line 2, NumPy and time are library functions used for performing matrix math and keeping track of time, respectively. Lines 3 to 7 indicate that the series has 50 transition state data's, with S denoting 0, I denoting 1, T denoting 2, and R denoting 3. Program from lines 9 to 10 defines four states from the memory address of n_states and generates a 4x4 matrix of zeros to accommodate integer values. Line 12 prints the matrix on the shell monitor, and Line 13 delays half a second. Line 14 to 18 counts the state transitions, which total of 49 pairs based on the 4 x 4 matrix of C_{ij}, build the C_{ij} matrix. Line number 20 calculates the transition probability matrix. Table 2 shows the results of the python program for C_{ij} and Table 3 shows results of the python program for P_{ij}.

Table 2.
Results of the python program for C_{ij}.

From \ To	S(0)	I(1)	T(2)	R(3)
S(0)	8	12	4	0
I(1)	8	6	7	4
T(2)	0	1	7	9
R(3)	7	7	0	6

The data shows that there were 8 changes from state 0 to state 0, 12 changes from state 0 to state 1, 4 changes from state 0 to state 2, and 0 changes from state 0 to state 3. The system went from state 1 to state 0 eight times, to state 1 six times, to state 2 seven times, and to state 3 four times. There were no transitions to state 0 while starting from state 2. There was only one transfer to state 1, seven transitions to state 2, and nine transitions to state 3. Lastly, there were seven changes from state 3 to state 0, seven changes from state 3 to state 1, none from state 3 to state 2, and six changes from state 3 to state 3. This matrix gives a full picture of the transition dynamics in the series, showing which transitions between states are the most and least common. The transition probability matrix from Table 3 shows how the system travels between its operational states: Sleep (S), Idle (I), Transmit (T), and Receive (R). When the system is in Sleep mode, it is most likely to switch to Idle mode (approximately 46% of the time), next to Sleep mode (31% of the time), and last to Transmit mode (15% of the time).

Table 3.
Results of the python program for P_{ij}.

From \ To	S(0)	I(1)	T(2)	R(3)
S(0)	0.3077	0.4615	0.1538	0
I(1)	0.3478	0.2609	0.3043	0.1739
T(2)	0	0.0526	0.3684	0.4737
R(3)	0.35	0.35	0	0.3

It's important to note that there are no straight transitions from Sleep to Receive. When the system is Idle, there is a 35% probability that it will go back to Sleep and a 30% chance that it will go to Transmit. There is also a 26% chance that it will stay Idle and a 17% chance that it will go to Receive. When the system is in Transmit mode, it usually goes to Receive (47%) or stays in Transmit mode (37%). There is only a 5% chance that it will go to Idle mode and never go back to Sleep mode. When the system is in Receive mode, it has a 35% chance of going to Sleep or Idle, a 30% chance of staying in Receive mode, and it does not go directly from Receive to Transmit. The matrix shows a logical sequence of operation: Sleep usually leads to Idle, Idle often leads to Transmit, Transmit changes to Receive, and after Receiving, the system goes back to either Sleep or Idle. This form shows that state management is done well, with fewer transitions that are unlikely or caused a lot of energy usage.

5. Discussion

Markov chains are a strong way to represent systems where the chance of each next state only depends on the current state. This is called "memorylessness." This trait makes it much easier to study complicated random processes that show up in many different areas. This study utilized Markov chain theory to mathematically represent state transitions, applying transition matrices and the Chapman-Kolmogorov equations to ascertain the probability distribution of subsequent states. These mathematical techniques help us understand both short-term changes and long-term steady-state behaviors. The example used Python and the NumPy package to put Markov-based methods into practice. Transition matrices were created and changed to test theoretical predictions and model changes in state. This computational method made it possible to quickly make prototypes, see them, and try them out, which improved the entire analysis. Markov chains have several uses that go beyond the scope of this paper. They are used in finance to predict credit risk and market trends. In computer science, they are the basis for search engine and recommendation system algorithms. Markov models are also used in biology to model DNA sequences, queueing theory, and even to predict the weather. So, combining Markov theory, mathematical modelling, and Python-based computation creates a flexible framework that can be used to solve many real-world situations.

6. Conclusions

The Markov chain is a highly important algorithm for improving and optimizing a wide range of process applications. Numerous studies have demonstrated the significant effectiveness of this algorithm in improving and optimizing various process applications, such as data and alarm messages in power line monitoring systems, energy-efficient pipeline monitoring systems, networks for linear monitoring applications, integrated networks for trackside smart weather monitoring, pervasive monitoring of underground environments, prolonging the lifespan of linear wireless sensor networks (LWSNs), and many others.

References

[1] F. Pitu and N. C. Gaitan, "Implementing a wide-area network and low power solution using long-range wide-are a network technology," *Technologies*, vol. 13, no. 1, p. 36, 2025. https://doi.org/10.3390/technologies13010036

[2] S. Syed, A. H. Sial, M. Shafiq, Z. S. Sial, and S. Ahmed, "Smart wireless sensor networks (WSNs): A review on concepts, techniques, and applications," *Sir Syed University Research Journal of Engineering & Technology*, vol. 15, no. 1, pp. 102-108, 2025. https://doi.org/10.33317/ssurj.682

[3] A. E. Hererdia, P. F. Lucero, F. A. Salinas, and A. V. Rodas, "Design and implementation of wireless sensor network with LoRa technology for industrial monitoring," *Latin-American Journal of Computing*, vol. 7, no. 2, pp. 48-63, 2020.

[4] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 2, pp. 2–23, 2006. https://doi.org/10.1109/COMST.2006.315852

[5] L. Abbad, A. Nacer, H. Abbad, M. T. Brahim, and N. Zioui, "A weighted Markov-clustering routing protocol for optimizing energy use in wireless sensor networks," *Egyptian Informatics Journal*, vol. 23, no. 3, pp. 483-497, 2022. https://doi.org/10.1016/j.eij.2022.05.001

[6] S. U. Rehman, S. Berber, and A. Swain, "Performance analysis of CSMA/CA algorithm for wireless sensor network," presented at the In TENCON 2010-2010 IEEE Region 10 Conference (pp. 2012-2017). IEEE, 2010.

[7] O. Salem, K. Alsubhi, A. Mehaoua, and R. Boutaba, "Markov models for anomaly detection in wireless body area networks for secure health monitoring," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 526-540, 2020. https://doi.org/10.1109/JSAC.2020.3020602

[8] C. Koushik, P. Vetrivelan, and E. Chang, "Markov chain-based mobility prediction and relay-node selection for QoS provisioned routing in opportunistic wireless network," *IETE Journal of Research*, vol. 70, no. 3, pp. 2233-2245, 2024. https://doi.org/10.1080/03772063.2023.2178534

[9] T. Arjannikov, S. Diemert, S. Ganti, C. Lampman, and E. C. Wiebe, "Using markov chains to model sensor network reliability," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017.

[10] M. Catelani, L. Ciani, A. Bartolini, C. Del Rio, G. Guidi, and G. Patrizi, "Reliability analysis of wireless sensor network for smart farming applications," *Sensors*, vol. 21, no. 22, p. 7683, 2021. https://doi.org/10.3390/s21227683

[11] M. A. Matin and M. Islam, *Overview of wireless sensor network, in Wireless Sensor Networks – Technology and Protocols, M. A. Matin, Ed*. Rijeka, Croatia: IntechOpen, 2012.

[12] C. Buratti, A. Conti, D. Dardari, and R. Verdone, "An overview on wireless sensor networks technology and evolution," *Sensors*, vol. 9, no. 9, pp. 6869-6896, 2009. https://doi.org/10.3390/s90906869

[13] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail, "Energy-efficient wireless sensor networks for precision agriculture: A review," *Sensors*, vol. 17, no. 8, p. 1781, 2017. https://doi.org/10.3390/s17081781

[14] M. Amine, A. Kobbane, J. Ben-Othman, and M. El Koutbi, "Green network slicing architecture based on 5G-IoT and next-generation technologies," *Applied Sciences*, vol. 15, no. 16, p. 8938, 2025. https://doi.org/10.3390/app15168938

[15] V. Kovtun and K. Grochla, "RETRACTED ARTICLE: Investigation of the competitive nature of eMBB and mMTC 5G services in conditions of limited communication resource," *Scientific Reports*, vol. 12, no. 1, p. 16050, 2022. https://doi.org/10.1038/s41598-022-20135-5

[16] S. Y. Huang, H. H. Cho, Y. C. Chang, J. Y. Yuan, and H. C. Chao, "An efficient spectrum scheduling mechanism using Markov decision chain for 5G mobile network," *IET Communications*, vol. 16, no. 11, pp. 1268-1278, 2022. https://doi.org/10.1049/cmu2.12263

[17] Y. Garbatov, D. Yalamov, and P. Georgiev, "Markov chain analysis of ship energy efficiency," *Energies*, vol. 17, no. 12, p. 3018, 2024. https://doi.org/10.3390/en17123018

[18] A. H. El Fawal, A. Mansour, H. El Ghor, N. A. Ismail, and S. Shamaa, "Modeling emergency traffic using a continuous-time Markov Chain," *Journal of Sensor and Actuator Networks*, vol. 13, no. 6, p. 71, 2024. https://doi.org/10.3390/jsan13060071

[19] M. Nurgaliyev, A. Saymbetov, Y. Yashchyshyn, N. Kuttybay, and D. Tukymbekov, "Prediction of energy consumption for LoRa based wireless sensors network," *Wireless Networks*, vol. 26, no. 5, pp. 3507-3520, 2020. https://doi.org/10.1007/s11276-020-02276-5

[20] L. REYAX and C. Technology, "LoRa AT command guide (RYLR40x / RYLR89x). Taiwan," 2020. https://reyax.com/upload/products_download/download_file/LoRa-AT-Command-RYLR40x_RYLR89x_EN-8.pdf?utm_source=chatgpt.com

[21] S. Electronics, "SparkFun LoRaSerial: AT commands documentation," 2024. https://docs.sparkfun.com/SparkFun_LoRaSerial/at_commands/?utm_source=chatgpt.com