Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

ВІСНИК

Харківського національного університету імені В.Н. Каразіна

Серія
«Математичне моделювання.
Інформаційні технології.
Автоматизовані системи управління»

Випуск 64

Серія заснована 2003 р.

BULLETIN

of V.N. Karazin Kharkiv National University

Series
«Mathematical Modeling.
Information Technology.
Automated Control Systems»

Issue 64

First published in 2003

Харків
2024

Засновник журналу Харківський національний університет імені В. Н. Каразіна, Харків, Україна. Рік заснування 2003. Періодичність: 4 випуски на рік. https://periodicals.karazin.ua/mia

Статті містять дослідження у галузі математичного моделювання та обчислювальних методів, інформаційних технологій, захисту інформації. Висвітлюються нові математичні методи дослідження та керування фізичними, технічними та інформаційними процесами, дослідження з програмування та комп'ютерного моделювання в наукоємних технологіях.

Для викладачів, наукових працівників, аспірантів, працюючих у відповідних або суміжних напрямках.

Наказом Міністерства освіти і науки України від 17.03.2020 № 409 наукове фахове періодичне видання Вісник Харківського національного університету імені В.Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління» включено до Категорії «Б» Переліку наукових фахових видань України за наступними спеціальностями: 113 – Прикладна математика;122 – Комп'ютерні науки та інформаційні технології; 123 – Комп'ютерна інженерія; 125 – Кібербезпека.

Затверджено до друку рішенням Вченої ради Харківського національного університету імені В. Н. Каразіна (протокол № 24 від 25.11.2024 р.)

Редакційна колегія:

Адреса редакційної колегії: 61022, м. Харків, майдан Свободи, 6, Харківський національний університет імені В. Н. Каразіна, к. 534.
Тел. +380 (57) 705-42-81, Email: journal-mia@karazin.ua.

Мова публікації: українська, англійська.

Статті пройшли внутрішнє та зовнішнє рецензування.

Ідентифікатор медіа у Реєстрі суб'єктів у сфері медіа: R30-04456
(Рішення № 1538 від 09.05.2024 р Національної ради України з питань телебачення і радіомовлення. Протокол № 15)

The founder of the Journal is V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
Year of foundation 2003. The journal is published four times a year.
https://periodicals.karazin.ua/mia

The articles are present research in the field of mathematical modeling and computing methods, information technologies, information security. New mathematical methods of research and management of physical, technical and information processes, research on programming and computer modeling in science-intensive technologies are covered.

For teachers, researchers, graduate students working in relevant or related fields.

By the order of the Ministry of Education and Science of Ukraine from 17.03.2020 № 409 scientific professional periodical Bulletin of V.N. Karazin Kharkiv National University series "Mathematical modeling. Information Technologies. Automated control systems" is included in Category "B" of the List of scientific professional publications of Ukraine in the following specialties: 113 – Applied Mathematics, 122 – Computer Science and Information Technology; 123 – Computer engineering; 125 – Cybersecurity.

Approved for publication by the decision of the Academic Council of V.N. Karazin Kharkiv National University (Minutes № 24 of 25.11.2024).

Editorial Address: 61022, Kharkiv, Svobodi sq., 6, V.N. Karazin Kharkiv National University, r. 534.
Phone. +380 (57) 705-42-81, Email: journal-mia@karazin.ua.

Language of publication: Ukrainian, English.

The articles pass internal and external review.

Media identifier in the Register of the field of Media Entities: R30-04456
(Decision № 1538 dated May 9, 2024 of the National Council of Television and Radio Broadcasting of Ukraine, Protocol № 15)

ЗМІСТ

CONTENTS

DOI: https://doi.org/10.26565/2304-6201-2024-64-01

УДК (UDC) 004.7:004.65

Berezovskyi Oleksandr
PhD student, Faculty of Applied Mathematics and Informatics
Ivan Franko National University of Lviv, 1, Universytetska St., Lviv, Ukraine, 79000
e-mail: oleksandr.berezovskyi@lnu.edu.ua
https://orcid.org/0009-0003-2241-3324

Terletskyi Mykola
PhD student, Faculty of Applied Mathematics and Informatics
Ivan Franko National University of Lviv, 1, Universytetska St., Lviv, Ukraine, 79000
e-mail: Mykola.Terletskyi@lnu.edu.ua
https://orcid.org/0009-0002-6369-0793

Distributed Data Storing Based on Distributed Transaction Ledger

The primary trend in the development of modern information technologies is the migration of computations to the cloud, making distributed computing the dominant strategy for information processing. In particular, this poses the challenge of reliable distributed data storage. A well-known approach to solving the problem of distributed data storage is blockchain or, more generally, distributed ledger technology. A key challenge of this technology is creating an effective mechanism for the global numbering of registry records. The complexity of solving this problem results from the fundamental limitations of distributed computing — the inability to accurately synchronize distributed computing processes and the limitations resulting from the CAP theorem for distributed data stores. The authors attempt to circumvent the mentioned limitations based on the hypothesis that such limitations can be overcome by considering both the network topology and narrowing the class of distributed systems to distributed registers. The work is based on methods of modeling distributed computing, particularly the model of space-time diagrams proposed by L. Lamport. This model allows us to introduce such a tool as logical clocks, including Lamport's logical clock algorithm. Unfortunately, Lamport's logical clock algorithm allows assigning a common timestamp to different events if they are concurrent. The paper proposes an algorithm that is a composition of Lamport's clock algorithm and the wave algorithm, which is not only a logical clock but also assigns different timestamps to different events. Thus, this algorithm provides a mechanism for the global numbering of entries of distributed ledger replicas. A problematic issue remains gaps in the series of ledger entry numbers. Thus, the paper proposes an effective mechanism for the global numbering of records of a distributed register and identifies a shortcoming of this mechanism. Further study is to identify specific conditions in terms of network topology that would ensure the absence of the mentioned shortcoming.

Keywords: *ledger, distributed ledger, network topology, distributed computation, logical clock, Lamport's clock*.

Як цитувати: Berezovskyi O., Terletskyi M. Distributed Data Storing Based on Distributed Transaction Ledger. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.6-12. https://doi.org/10.26565/2304-6201-2024-64-01

How to quote: O. Berezovskyi, and M. Terletskyi, "Distributed Data Storing Based on Distributed Transaction Ledger", *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 6-12, 2024. https://doi.org/10.26565/2304-6201-2024-64-01

Introduction

The exponential growth of data has necessitated innovative approaches to data storage and management. While efficient for structured data, traditional centralized database systems face scalability and security challenges as data volumes increase and distribution becomes critical [1]. Distributed data storage systems, on the other hand, offer scalability and fault tolerance but often compromise on data consistency and reliability.

A fundamental challenge in distributed data storage systems is the CAP theorem, which states that it is impossible to simultaneously guarantee Consistency, Availability, and Partition Tolerance for a distributed system [2].

Here, we understand consistency as strong or weak consistency more precisely, we say strong consistency whenever all nodes in the system see the same data at the same time; in contrast, if all nodes will see the same data eventually but there may be delays then we say about weak consistency.

Availability is understanding the system must always be available to process requests, even in the face of failures.

Partition Tolerance means the system must continue to operate correctly even if a system part loses the ability to communicate with its other parts.

The trade-off of these requirements forces system designers to decide which properties to prioritize.

Understanding the CAP theorem is crucial for designing and implementing distributed data storage systems. By carefully considering the trade-offs involved, system architects can make informed decisions about prioritizing these properties depending on their importance for their specific use case.

This research proposes a hybrid approach that combines the strengths of Acidic Database Management Systems (ACID DBMS) and Distributed Ledger Technology (DLT) [3]. By integrating the ACID properties of databases with the distributed and immutable nature of DLTs, we aim to create a robust, scalable, and secure distributed data storage solution [1].

Our approach is based on the assumption of a loss-free transmission of information in the system [4].

We are not expecting the proposed approach to be the panacea for any system development context. Moreover, one of our research goals is to prove that such an expectation is unjustified. But we hope this approach is fruitful for special network topologies under some fairness guarantees.
In this paper, we demonstrate the grounds of our expectations.

1. Distributed Computation

A distributed computation is an information processing performed by at least two processing units interacting via a network [1]. The key constraints on processing unit behavior are that each unit performs some sequential computation, and that message passing is the only interaction between units .

Thus, a distributed computation is determined by a network, whose nodes perform local sequential processes, by these local processes, and by features of the message-passing by processes. Below, we give the mathematical model of a distributed computation presented by O. Barskyi, I. Illin, and A. Zozulia on PhD Symposium of ICTERI 2024 Conference.

Typically, a processing unit that performs a particular local process of a distributed computation, is identified with that process and is called a computation participant.

Considering that, one can classify distributed computation as follows [4].

1. The local behavior of each computation participant corresponds completely to the computation specification, and each sent message will be eventually received.

2. The local behavior of a computation participant can violate the computation specification, but each sent message will be eventually received.

3. The local behavior of each computation participant completely corresponds to the computation specification but there is a nonzero probability of losing a message under transmission.

4. The local behavior of a computation participant can violate the computation specification and there is a nonzero probability of losing a message under transmission.

Usually, we refer to case 1 as a reliable computation, case 2 as a computation with Byzantine failures, case 3 as a computation with non-Byzantine failures [5], and case 4 as a general-kind computation.

Note that non-Byzantine failures are rather associated with the unreliability of network equipment than with incorrect behavior of computing participants. Therefore, we focus our attention on studying case 1 in the paper.

Model of a Network [6]**.** As it is generally accepted, we understand a network [7] as an oriented or unoriented graph with nodes that refer to participants and edges that indicate the possibility of transmitting data directly from the source of the edge to the target of the edge. The standard constraints for the network graph are that

- There is no loop in the graph. This means that there is no way to transmit data directly to itself.
- For any two nodes, there exists a route connecting these nodes. This constraint is referred to as the connectivity.

Model of a Participant Behavior. The next key component of distributed computation is the set of the computation participants' behaviors. However, if we have n participants in the computation it does

not mean that each participant has unique behavior. As a rule, we can divide all participants into classes of participants having the same behavior. Usually, the number m of these classes is less than n. The references to these classes are called roles as usual. Of course, the number of roles can be equal to one. For example, all participants of a distributed ledger have the same behavior which means the number of roles is one.

Local behavior is a sequence of computation actions and two kinds of additional actions: sending and receiving messages. These kinds of action provide interaction of the computation participants. So, for modeling local behaviors of a distributed computation, one can use any model of single-process computation extended with the mentioned two behavior primitives.

Events of a Computation. It is generally accepted that any action in a distributed computation is atomic. This assumption provides the ability to consider an action as an event, that is, some instantaneous change in the configuration of the system. According to L. Lamport [xx], the set of computational events E is equipped with a partial order called a causal order or a "happen-before" relation. The fact "an event e_1 has happened before an event e_2" is denoted by $e_1 \to e_2$. This relation is defined [8, Definition at p. 559] as follows:

1) if e_1 and e_2 are events corresponding to local actions of the same participant, which are neither sending nor receiving events, and e_1 precedes e_2 in the local ordering of this participant's events then $e_1 \to e_2$;
2) if e_1 is a message sending event, and e_2 is the receiving event of this message then $e_1 \to e_2$;
3) if $e_1 \to e_2$ and $e_2 \to e_3$ then $e_1 \to e_3$.

If both $e_1 \nrightarrow e_2$ and $e_2 \nrightarrow e_1$ are true, then events e_1 and e_2 are called concurrent. In this case, the denotation $e_1 \parallel e_2$ is used.

Logical Clock of a Computation. The concept of a logical clock was introduced for providing a manner to order events of a computation [8]. Firstly, we need a set T of timestamps which are surrogates of physical timepoints. The natural requirement is such a set should be a well-ordered set of the order type ω. It means that $T = \{t_0 < t_1 <\ldots< t_{n-1} < t_n < t_{n+1} <\ldots\}$.

A logical clock is a function $c: E \to T$ such that $c(e_1) < c(e_2)$ whenever $e_1 \to e_2$.

Let us discus a simple but important proposition.

Proposition. Let $c: E \to T$ be a logical clock then if $c(e_1) = c(e_2)$ then $e_1 \parallel e_2$ for any $e_1 \neq e_2 \in E$.

Proof. Indeed, there the following variants only for any $e_1, e_2 \in E$:

1) $e_1 \to e_2$ and $e_2 \to e_1$ is impossible because it contradicts to irreflexivity of the causal order.
2) $e_1 \to e_2$ and $e_2 \nrightarrow e_1$ is impossible because it contradicts to $c(e_1) = c(e_2)$ due to $c(e_1) < c(e_2)$.
3) $e_1 \nrightarrow e_2$ and $e_2 \to e_1$ is impossible because it contradicts to $c(e_1) = c(e_2)$ similarly to the previous.
4) and only $e_1 \nrightarrow e_2$ and $e_2 \nrightarrow e_1$ i.e. $e_1 \parallel e_2$ does not contradict to $c(e_1) = c(e_2)$.

Qed.

Lamport's clock algorithm. L. Lamport proposed the following algorithm to equip each computational event with a timestamp for $T = N$ where N is an ordinal of natural numbers. All participants of this algorithm behave similarly. In other words, the role set of this algorithm is a singleton.

Algorithm of Lamport's clock.

Assume

$p_1, \ldots, p_n (n \geq 2)$ be participants of a computation,

e_i^k denote the *k*-th event in the event sequence of the participant p_i.

Ensure

a logical clock $Lamport: E \to N$.

The algorithm requires to equip each participant p_i with a counter ts_i and initializes it by zero.

Behavior of p_i.

Catch an event e.

If e is an event corresponding to a local action of p_i then $c(e) \leftarrow ts_i$ and $ts_i \leftarrow ts_i + 1$.
If e is a message m sending event then $c(e) \leftarrow ts_i$, $ts_i \leftarrow ts_i + 1$, add ts_i to m and resend m.
If e is a message m receiving event, then $ts_i \leftarrow max(ts, ts_i)$, $c(e) \leftarrow ts_i$, and $ts_i \leftarrow ts_i + 1$.
Jump to the first line.

It is known that this algorithm computes a logical clock, and this clock referred to as $Lamport$ satisfies the following condition

$$Lamport(e) \leq c(e) \text{ for any clock } c: E \to N \text{ and } e \in E. \quad (1)$$

It is because of (see [Tel, p. 62])

$Lamport(e) = 0$ if and only if there is no $e' \in E$ such that $e' \to e$;
$Lamport(e) = n > 0$ if and only if $n = max\big(Lamport(e') \big| e' \in P(e)\big) + 1$
where $P(e) = \{e' \in E \vee e' \to ebutthereisnoe'' \in Esuchthate' \to e'' \to e\}$.

2. Model of Distributed Ledger

Before considering a distributed ledger (DL), we construct the model of a ledger, which is understood as a data store that allows only writing new data units and reading stored data units. In this context, this store can be represented by a finite set S of pairs $(timestamp, data_{unit})$ where the timestamp is a key. We do not fix a set of data units D and consider it as a model parameter. Also, we restrict the set T of timestamp values by ordinals of the order type ω. The concrete ordinal is selected depending on the studied problem. It can be the ordinal of natural numbers with the natural ordering or the ordinal of strings with lexicographic ordering [8]. For storing new data unit d, we choose the key

$$ts = min\{ts' \vee \forall x \in D, (ts', x) \notin S\} \quad (2)$$

and then write the pair (ts, d). Of course, formula (2) is suitable only for specifying a new key value, and not for its calculation, the algorithm of which depends on the specific choice of the set T.

A distributed ledger (DL) is an information system that stores data across network nodes (belonging to a set P) and allows only two types of transactions: writing new data and retrieving written data. It is suggested that each node in the network has its ledger called a replica. It ies assumed that each node in the network has its own ledger, called a replica. A distributed ledger exists physically only as a collection of replicas, so the consistency of these replicas is important.

Replica consistency can be formalized as follows

$$\forall ts \in T, p, q \in P, x_p, x_q \in D, (ts, x_p) \in S_p \to (ts, x_q) \in S_q \to x_p = x_q \quad (3)$$

In formula (3), S_p and S_q refer to the replicas owned by nodes p and q respectively.

Formula (3) does not require that replicas match for all nodes, it only requires that replica keys be not only local but also global keys.

Of course, due to the CAP theorem [2], we cannot expect the existence of an algorithm to record data to all local ledgers consistently. In more detail, DL nodes receive write requests and interact to execute these requests by the nodes' local ledgers modifying consistently.

So, we need a method for assigning the correct timestamp to any request to write a new data unit.

More details, let $p_1, \dots, p_n$ $(n \geq 2)$ be nodes of a DL being studied, w_i^k be k-th request for writing obtained by node p_i, and e_i be another event observed by node p_i (if we need to consider several such events, we use superscripts for these events distinguishing).

Of course, we can use Lamport's clock algorithm to determine the local keys of each DL node.

This approach, of course, ensures the inequality $Lamport(w_i^k) < Lamport(w_i^l)$ if $k < l$.

But, if we order events w_i^k and w_j^l, we can obtain $Lamport(w_i^k) = Lamport(w_j^l)$ due to $w_i^k \parallel w_j^l$.

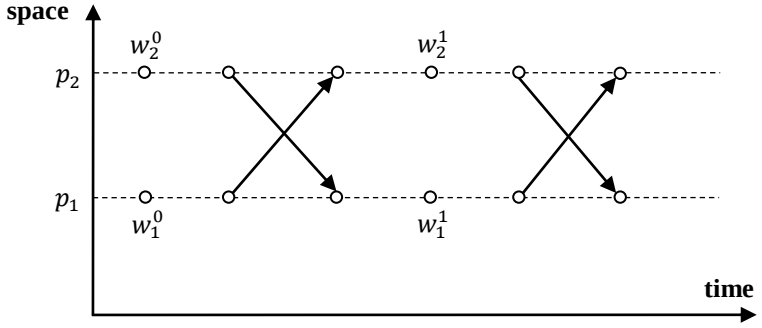


Fig.1. Lamport's timestamp conflict

In Fig. 1 we illustrate Lamport's timestamp conflict. In this figure, the unlabeled circles correspond to the events of sending/receiving notifications concerning timestamps. One can easily see those events w_1^0 and w_2^0 and w_1^1 and w_2^1 of receiving requests for writing a data unit have the same Lamport's timestamp equal respectively 0 and 3.

3. DL Timestamping Algorithm

The example from the previous section not only illustrates an anomaly in the Lamport clock algorithm for distributed ledgers but also allows us to understand how to modify this algorithm to avoid this anomaly.

The algorithm is applied if a wave algorithm is available for the network topology [6, 7]. Moreover, the use of wave algorithms specific to a particular network topology can significantly change the efficiency of the algorithm.

DL timestamping algorithm.

Assume:

$p_1, \dots, p_n$ be nodes of a DL where $n \geq 2$;

$T = N \times \{1, \dots, n\}$ be the timestamp set with the lexicographic order that turns it into a well-ordered set of order type ω

Ensure:

add a data unit with correct timestamp

Behavior of p_i.

Catch an event e.

If e is a request for writing data unit d, then

$ts \leftarrow (Lamport(e), i)$

write d with key ts

start wave by sending notification (ts, d)

If e is an event of notification receiving, then

if a message with ts from (ts, d) has received firstly, then write d with key ts

execute step of the wave algorithm adding Lamport's timestamp to all necessary messages

If e is an event of other kinds, then ignore it.

Jump to the first line.

The proposed algorithm guarantees that timestamps for writing requests received by the same ledger node are ordered by their arrival at this node. Due to Lamport's clock algorithm and the concurrency of the writing requests obtained by different ledger nodes, we can conclude that Lamport's timestamps of such requests can be equal.

Therefore, we add the identifier of the ledger node that has obtained a writing request to Lamport's timestamp of this request. This addition allows us to avoid the anomalies of logical clocks like the anomaly from the example.

Returning to the example, request event w_1^0 obtains the timestamp (0,1), event w_2^0 obtains the timestamp (0,2), event w_1^1 obtains the timestamp (3,1), and event w_2^1 obtains the timestamp (3,2). In other words, we have the following order of request events: w_1^0, w_2^0, w_1^1, w_2^1. However, we must note that the series of timestamp values has gaps. In the example, we do not have entries with timestamps (1,1), (1,2), (2,1), and (2,2) between the entries with the timestamps (0,2) and (3,1) respectively.

4. Conclusion

The previous consideration demonstrated that, under the crucial assumptions of network reliability and strict adherence to the algorithm by all participants in the distributed ledger, the composition of a wave algorithm and Lamport's clock algorithm gives a viable mechanism for establishing global numbering of ledger entries. This mechanism provides a crucial foundation for many distributed ledger applications, offering a consistent and ordered view of the transaction history.

However, the proposed algorithm, as outlined in the "DL Timestamp Algorithm" section, exhibits a significant drawback: it generates a series of timestamps with gaps. This gap, while not necessarily fatal to all applications, can pose challenges in certain scenarios. For instance, in a distributed transaction registry for a complex of ACID databases, consistent ordering and the absence of gaps in timestamps are paramount for maintaining data integrity and ensuring the correct execution of transactions across multiple nodes.

The presence of gaps in the timestamp series necessitates further investigation and refinement of the algorithm. This leads to several key open questions.

1) Robustness against Network Failures: the current algorithm's behavior in the face of network disruptions, both non-Byzantine (e.g., temporary network partitions) and Byzantine (e.g., malicious attacks), needs thorough evaluation. In the context the following are interesting:

- Can the algorithm be modified to maintain its functionality and ensure consistent timestamping even if such failures are present?
- What are the trade-offs between robustness and performance in such scenarios?

2) Algorithmic Complexity: assessing the computational complexity of the algorithm is crucial for practical implementation. In the context the following are interesting:

- How does the complexity scale with the number of nodes in the distributed ledger network?
- Are there optimization techniques for reducing the computational overhead while maintaining accuracy and consistency?

3) Physical Time Estimation: while the algorithm provides a logical ordering of events, it's essential to understand the relationship between the generated timestamps and physical time; in other words

- Can we estimate the physical time of execution of a round of the algorithm with reasonable accuracy?

This information is crucial for performance analysis, debugging, and potentially for applications that require real-time constraints.

Addressing these open questions is crucial for further development and practical application of the proposed global numbering mechanism.

Now let us outline some points for future research directions:

- Exploring alternative approaches to global numbering that may be more resilient to network failures.
- Investigating techniques for minimizing the occurrence of gaps in the timestamp series.
- Developing analytical models to predict the performance and behavior of the algorithm in different network environments.

By addressing these challenges and refining the proposed algorithm, we can pave the way for more robust and efficient distributed ledger technologies with applications in various domains, from financial systems to e-government systems.

REFERENCES

1. Françoise Baude, Denis Caromel, Nathalie Furmento, David Sagnol, Optimizing remote method invocation with communication–computation overlap. Future Generation Computer Systems, Vol. 18, Issue 6, pp. 769-778б ISSN 0167-739X, 2002.
   URL: https://www.sciencedirect.com/science/article/pii/S0167739X02000493
2. Brewer, Eric A. “CAP twelve years later: How the "rules" have changed.” Computer 45 (2012): 23-29. URL: https://ieeexplore.ieee.org/document/6133253
3. Hussein, Z., Salama, M.A. & El-Rahman, S.A. Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms. Cybersecurity 6, 30 (2023). URL: https://doi.org/10.1186/s42400-023-00163-y
4. N. Olifer, V. Olifer, Computer Networks: Principles, Technologies and Protocols for Network Design. Wiley, 2006, 1008 p.
   URL: https://www.wiley.com/en-br/Computer+Networks%3A+Principles%2C+Technologies+and+ Protocols+for+Network+Design-p-9780470869826
5. Gong, Li, Patrick Lincoln and John M. Rushby. “Byzantine Agreement with Authentication: Observations and Applications in Tolerating Hybrid and Link Faults”. In: Dependable Computing and Fault Tolerant Systems. Vol. 10 (1995), pp. 139-157.
   URL: https://www.csl.sri.com/papers/dcca95/dcca95.pdf
6. I. V. Stetsenko, Systems modeling: textbook. Cherkasy: CSTU, 2010, 399 p. [in Ukrainian]

7. U.I. Losev, K.M. Rukkas, S.I. Shmatkov. Computer Networks: textbook. Kharkiv: V.N. Karazin Kharkiv National University, 2013, 248 p. [in Ukrainian] http://www.its.kpi.ua/itm/lgloba/Lists/publications/Attachments/15/(11)--TUD_IBIS_Shill_Globa_NTUU_KPI_camera_ready.pdf
8. L. Lamport. “Time, clocks, and the ordering of events in a distributed system.” CACM, 21(7) (1978), pp. 558-565. URL: https://dl.acm.org/doi/pdf/10.1145/359545.359563
9. W. Fokkink. “Distributed Algorithm: An Intuitive Approach. The MIT Press, 2013. URL: https://mitpress.mit.edu/9780262037662/distributed-algorithms/
10. G. Tel. “Introduction to Distributed Algorithms”. The Cambridge University Press, 2000. URL: https://www.amazon.com/Introduction-Distributed-Algorithms-Gerard-Tel/dp/0521470692

Березовський Олександр Валерійович *Аспірант*
Львівський національний університет імені Івана Франка, вул. Університетська, 1, Львів, 79000

Терлецький Микола Віталійович *Аспірант*
Львівський національний університет імені Івана Франка, вул. Університетська, 1, Львів, 79000

Розподілене зберігання даних на основі технології розподіленого реєстру

Основною тенденцією розвитку сучасних інформаційних технологій є перенесення обчислень в хмару, що робить розподілені обчислення домінуючою стратегією обробки інформації. Зокрема, це ставить задачу надійного розподіленого зберігання інформації. Відомим підходом до вирішення проблеми розподіленого зберігання даних є блокчейн або, у більш загальному випадку, технологія розподіленого реєстру. Ключовою проблемою цієї технології є створення ефективного механізму глобальної нумерації записів реєстру. Складність вирішення проблеми є наслідком фундаментальних обмежень розподілених обчислень - відсутністю можливості точної синхронізації процесів розподіленого обчислення та обемежень, що є наслідками CAP теореми для розподілених сховищ даних. Виходячи з гіпотези про те, що такі обмеження можуть бути подолані шляхом врахування як особливостей топології мережі, так і звуженням класу розподілених систем до розподілених реєстрів, автори намагаються обійти зазначені обмеження. В основі роботи лежать методи моделювання розподілених обчислень, зокрема, модель просторово-часових діаграм, запропонована Л. Лемпортом. Ця модель дозволяє ввести такий інструмент, як логічні годинники, включно з алгоритмом логічного годинника Л. Лемпорта. Нажаль, алгоритм логічного годинника Л. Лемпорта допускає приписування спільної позначки часу для різних подій за умови їх конкурентності. В роботі запропановано алгоритм, який є композицією алгоритму годинника Л. Лемпорта та хвильового алгоритму, який не тільки є логічним годинником, але й приписує різні позначки часу різним подіям. Таким чином, цей алгоритм дає механізм глобальної нумерації записів реплік розподіленого реєстру. Проблемним питанням залишається наявність лакун серед номерів записів реєстру. Отже, в роботі запропонований ефективний механізм глобальної нумерації записів розподіленого реєстру та виявлений недолік цього механізму. Подальшим розвитком дослідження є з’ясування специфічних умов в термінах топології мережі, які забезпечували б відсутність зазначеного недоліка.

Ключові слова: *реєстр, розподілений реєстр, мережева топологія, розподілене обчислення, логічний годинник, годинник Лемпорта.*

Надійшла у першій редакції 12.08.2024, в останній - 19.09.2024.
The first version has been received on 12.08.2024, the final version - on 19.09.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-02

УДК (UDC) 519.9

Бондаренко Костянтин Володимирович *студент, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: bondarenko.konstantin.privat@gmail.com
https://orcid.org/0009-0003-9911-7507

Стрілець Вікторія Євгенівна *к.т.н., доцент кафедри комп'ютерних систем та робототехніки, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: viktoria.strilets@karazin.ua
https://orcid.org/0000-0002-2475-1496

Шевченко Дмитро Олександрович *аспірант, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: dimyich24@gmail.com
http://orcid.org/0000-0002-7897-250X

Прогнозування економічних показників за допомогою моделі LSTM

Актуальність. Прогнозування економічних показників, зокрема цін на нафту, є критично важливим для різних галузей, таких як енергетика, фінанси, виробництво, транспорт та урядова політика. Точні прогнози сприяють прийняттю обґрунтованих рішень та оптимізації ресурсів. В умовах високої волатильності цін на нафту традиційні методи прогнозування, такі як ARIMA, часто не забезпечують належної точності, тому використання сучасних методів, таких як LSTM, є необхідним.
Метою дослідження є підвищення точності прогнозування економічних показників, зокрема цін на нафту марки Brent з використанням моделі LSTM та порівняння її точності з традиційними методами, зокрема ARIMA.
Методи дослідження. Для дослідження було використано два основні методи прогнозування часових рядів: ARIMA та LSTM. Було проведено розвідувальний аналіз даних, підготовку даних, побудову моделей, їх налаштування та оцінку за допомогою метрик MSE, MAE та RMSE. Дані про ціни на нафту марки Brent були оброблені та нормалізовані перед подачею в моделі.
Результати. Модель LSTM показала значно вищу точність прогнозування порівняно з ARIMA. Значення метрик для LSTM (MSE = 0.003, MAE = 0.055, RMSE = 0.055) суттєво перевищують відповідні значення для ARIMA (MSE = 12.59, MAE = 2.84, RMSE = 3.55). LSTM краще обробляє нелінійні залежності та волатильність даних, що робить її оптимальним вибором для довгострокового прогнозування.
Висновки. Використання LSTM для прогнозування економічних показників, зокрема цін на нафту, є більш ефективним у порівнянні з традиційними методами. Модель демонструє здатність до точного моделювання складних залежностей та адаптації до волатильності ринку, що робить її надійним інструментом для прогнозування в сучасних умовах.

***Ключові слова**: прогнозування економічних показників, LSTM, ARIMA, ціни на нафту, машинне навчання, часові ряди, волатильність.*

Як цитувати: Бондаренко К. В., Стрілець В. Є., Шевченко Д. О. Прогнозування економічних показників за допомогою моделі LSTM. *Вісник Харківського національного університету імені В.Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.13-24. https://doi.org/10.26565/2304-6201-2024-64-02

How to quote: K. V. Bondarenko, V. Y. Strilets, D. O. Shevchenko, "Forecasting economic indicators using the LSTM model" *Bulletin of V.N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 13-24, 2024. https://doi.org/10.26565/2304-6201-2024-64-02[in Ukrainian]

1 Вступ

Прогнозування економічних показників таких, як ціни на нафту є важливим аспектом в економіці, фінансах та багатьох інших галузях. Ціна на нафту впливає на вартість енергоносіїв, транспорту, виробництва та багатьох інших сфер діяльності. Точне прогнозування цін на нафту допомагає підприємствам та урядам приймати обґрунтовані рішення та планувати свої дії. Наприклад, прогнозування цін на нафту має важливе значення для:

1. Планування виробництва енергетичними компаніями. Енергетичні компанії використовують прогнози цін на нафту для планування обсягу видобутку та виробництва. Це допомагає оптимізувати ресурси та зменшити витрати.

2. Управління запасами ресурсів. Прогнозування допомагає ефективно управляти сировини, що забезпечує безперервність виробництва.

3. Інвестиційні стратегії. Інвестори використовують прогнози цін на нафту для розробки стратегій торгівлі та інвестування. Це допомагає зменшити ризики та підвищити прибутковість інвестицій.

4. Планування витрат у транспортній логістиці. Компанії, що займаються транспортом, використовують прогнози цін на нафту для планування своїх витрат на паливо. Це допомагає ефективніше розподіляти бюджет та встановлювати ціни на свої послуги.

5. Формування бюджетів компаній і країн. Прогнозування допомагає урядам планувати доходи від нафтогазового сектора, що є важливим для формування державного бюджету.

Таким чином, прогнозування цін на нафту є критично важливим для багатьох галузей економіки та має значний вплив на прийняття рішень на різних рівнях. Використання сучасних методів прогнозування, таких як LSTM, дозволяє отримувати більш точні та надійні прогнози, що сприяє підвищенню ефективності та стабільності роботи підприємств та організацій.

Метою дослідження є підвищення точності прогнозування цін на нафту марки Brent за допомогою розробки відповідних моделей на основі методів глибокого навчання, зокрема довготривалої короткострокової пам'яті (LSTM). Використання LSTM обумовлене його здатністю ефективно обробляти часові ряди та враховувати довготривалі залежності у даних.

У роботі також була здійснена спроба побудови моделі на основі авторегресійної інтегрованої моделі ковзного середнього (ARIMA). Однак результати показали, що ARIMA має обмежену здатність до точного прогнозування в умовах високої волатильності даних про ціни на нафту.

2 Моделювання економічних показників

Під час моделювання або аналізу показникии економічних систем роглядають у вигляді часових рядів [1]. Прогнозування часових рядів є важливим інструментом в аналізі даних, що дозволяє передбачати майбутні значення на основі наявних історичних даних. Цей підхід, що широко використовується в багатьох сферах, таких як економіка, фінанси, енергетика, метеорологія, виробництво та багато інших. Основна мета прогнозування часових рядів полягає у створенні моделі, яка може точно передбачати майбутні значення ряду.

Прогнозування часових рядів є ключовим напрямком досліджень в області машинного навчання, особливо з появою глибоких нейронних мереж, таких як LSTM. Дослідження показують, що LSTM має здатність ефективно моделювати часові залежності та враховувати довготривалі залежності у даних, що робить його більш ефективним у порівнянні з традиційними методами, такими як ARIMA [2].

LSTM (Long Short-Term Memory) – це різновид рекурентних нейронних мереж, розроблений для подолання проблеми довгострокової залежності, яка є характерною для традиційних рекурентних нейронних мереж. LSTM здатна зберігати інформацію на довгі періоди часу, що робить її ефективною для задач прогнозування часових рядів [3].

Дослідження [2, 4-6] показують, що моделі LSTM широко застосовуються в різних галузях, включаючи фінанси, охорону здоров'я та енергетику. Особливо успішно вони використовуються для прогнозування цін на нафту завдяки здатності враховувати нелінійні залежності та волатильність ринку.

Модель ARIMA (Autoregressive Integrated Moving Average) є класичним методом прогнозування часових рядів, що базується на авторегресії, інтеграції та ковзному середньому [2, 7, 8]. ARIMA ефективна для прогнозування стаціонарних часових рядів і широко використовується в економетричному моделюванні [9, 10].

Однак, модель ARIMA має обмеження в умовах високої волатильності та нелінійних залежностей, що часто спостерігаються в реальних даних про ціни на нафту [11].

Згідно з оглядом літератури, сучасні дослідження підтверджують, що методи глибокого навчання, такі як LSTM, мають значні переваги перед традиційними методами, такими як ARIMA, у задачах прогнозування складних часових рядів. Наприклад, у статті [12] розглядаються різні глибокі послідовні моделі та їх застосування для прогнозування часових рядів, включаючи LSTM, що демонструє їх ефективність у порівнянні з традиційними методами.

Це дослідження підкреслює важливість використання глибоких нейронних мереж у сучасних задачах прогнозування, що підтверджує обґрунтованість вибору методу LSTM для даної курсової роботи.

3. Опис даних та розвідувальний аналіз

3.1 Набір даних для дослідження

Для наукового дослідження було використано датасет, що містить дані про ціни на нафту марки Brent. Джерело датасету — відкритий архів історичних цін на нафту, доступний для завантаження з Інтернету (https://www.kaggle.com/datasets/mabusalah/brent-oil-prices).

Датасет охоплює тривалий період, починаючи з 1980х років, що дозволяє аналізувати динаміку змін цін на нафту протягом багатьох років. Це забезпечує достатньо даних для тренування та тестування моделей прогнозування. Дані збираються щоденно, що забезпечує високу точність та деталізацію для аналізу та побудови прогнозних моделей.

Кожен запис у датасеті складається з:

– дата (Date): стовпчик, що містить дату спостереження у форматах "%d-%b-%y" або "%b %d, %Y".

– ціна (Price): стовпчик, що містить ціну на нафту марки Brent у відповідний день.

Дані були очищені від пропущених значень та приведені до єдиного формату дати для забезпечення коректності аналізу. Датасет був відсортований за датою, що дозволило представити дані як послідовність часових рядів.

Значення змінної Price були нормалізовані для подальшого використання у моделях глибокого навчання, що дозволяє поліпшити продуктивність та стабільність моделей.

Використаний датасет забезпечує достатню кількість даних для проведення аналізу та побудови прогнозних моделей. Його основні характеристики, такі як тривалий період покриття, щоденна частота даних та попередня обробка, дозволяють ефективно використовувати сучасні методи машинного навчання для прогнозування цін на нафту марки Brent.

3.2. Розвідувальний аналіз даних

Розвідувальний аналіз даних (EDA) є важливим етапом у підготовці даних для моделювання, оскільки він допомагає зрозуміти основні характеристики та структуру даних [8].

Для датасету цін на нафту марки Brent спочатку був побудований графік часового ряду (рис. 1), щоб візуалізувати зміну цін на нафту протягом часу. Це допомагає ідентифікувати тренди, сезонність та будь-які аномалії у даних.

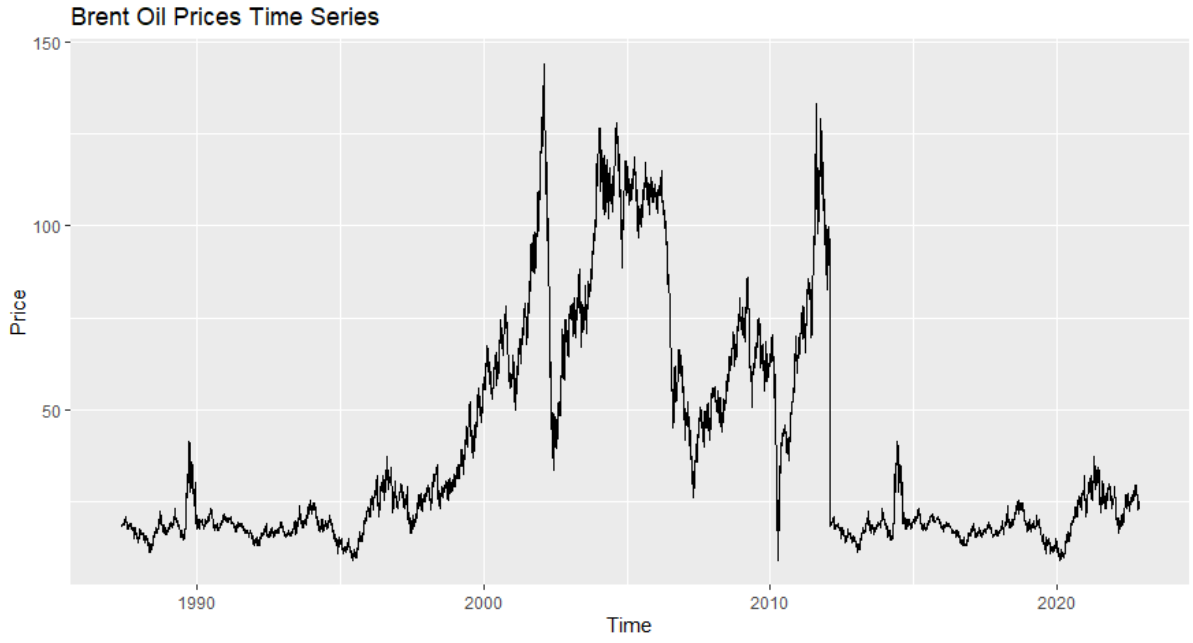


Рисунок 1. Графік часового ряду ціни на нафту

Було обчислено основні статистичні показники, такі як середнє, медіана, стандартне відхилення, мінімум та максимум цін на нафту (табл. 1). Вони показують розподіл даних та їх варіативність.

Таблиця 1. Статистичні показники часового ряду

Min	1st Qu.	Median	Mean	3rd Qu.	Max
9,10	19,05	38,57	48,42	70,09	143,95

Було проведено аналіз на наявність сезонності та довгострокових трендів у даних. Для цього використовувалися методи декомпозиції часових рядів, що дозволяють виділити тренд, сезонну складову та залишкові компоненти (рис. 2).

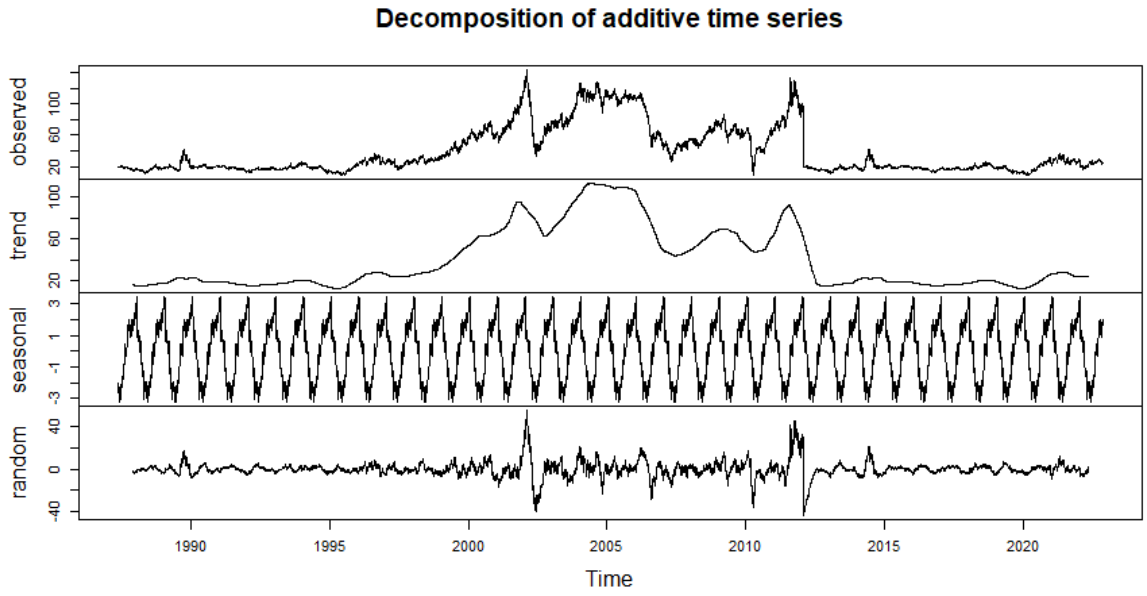


Рисунок 2. Декомпозиція часового ряду

Було побудовано графік автокореляційної функції (ACF) для виявлення автокореляції у даних. Графік показує, що поточні значення цін на нафту залежать від попередніх значень (рис. 3).

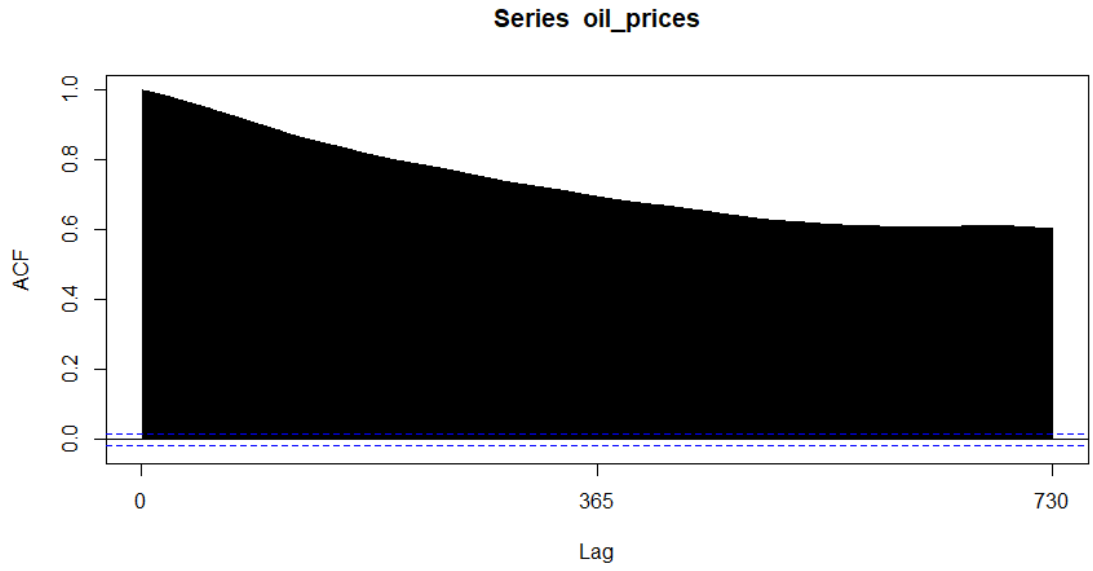


Рисунок 3. Графік автокореляційної функції

Розвідувальний аналіз даних дозволив глибше зрозуміти структуру та характеристики датасету цін на нафту марки Brent. Виявлені тренди, сезонність та автокореляція забезпечують цінну інформацію для подальшого моделювання та прогнозування. Використання EDA допомагає підготувати дані до побудови ефективних моделей машинного навчання, таких як LSTM, що дозволяє отримувати більш точні прогнози.

4 Побудова і оцінка моделей прогнозування

4.1. Модель ARIMA

Модель ARIMA (Autoregressive Integrated Moving Average) є однією з найбільш поширених і широко використовуваних моделей для аналізу та прогнозування часових рядів [7].

Вона поєднує три основні компоненти:

– авторегресію (AR): цей компонент передбачає, що значення часових рядів залежить від попередніх значень. Модель авторегресії включає параметр p, який вказує на кількість попередніх значень, що враховуються при прогнозуванні;

– інтеграцію (I): цей компонент включає різницювання, щоб зробити ряд стаціонарним. Параметр d вказує на кількість разів, які потрібно різницювати ряд для досягнення стаціонарності;

– ковзне середнє (MA): цей компонент враховує попередні похибки прогнозування. Модель ковзного середнього включає параметр q, який визначає кількість попередніх похибок, що використовуються для корекції прогнозу.

Модель ARIMA зазвичай позначається як ARIMA (p, d, q), де p – порядок авторегресії, d – порядок інтеграції, q – порядок ковзного середнього. У роботі значення параметрів p, d, q були отримані із використанням функції auto.arima, яка автоматично їх підібрає.

Модель ARIMA була налаштована з параметрами ARIMA (2,1,2). На її основі змодельовані фактичні, тестові та прогнозовані значення цін на нафту марки Brent за останні 60 днів (рис. 4).

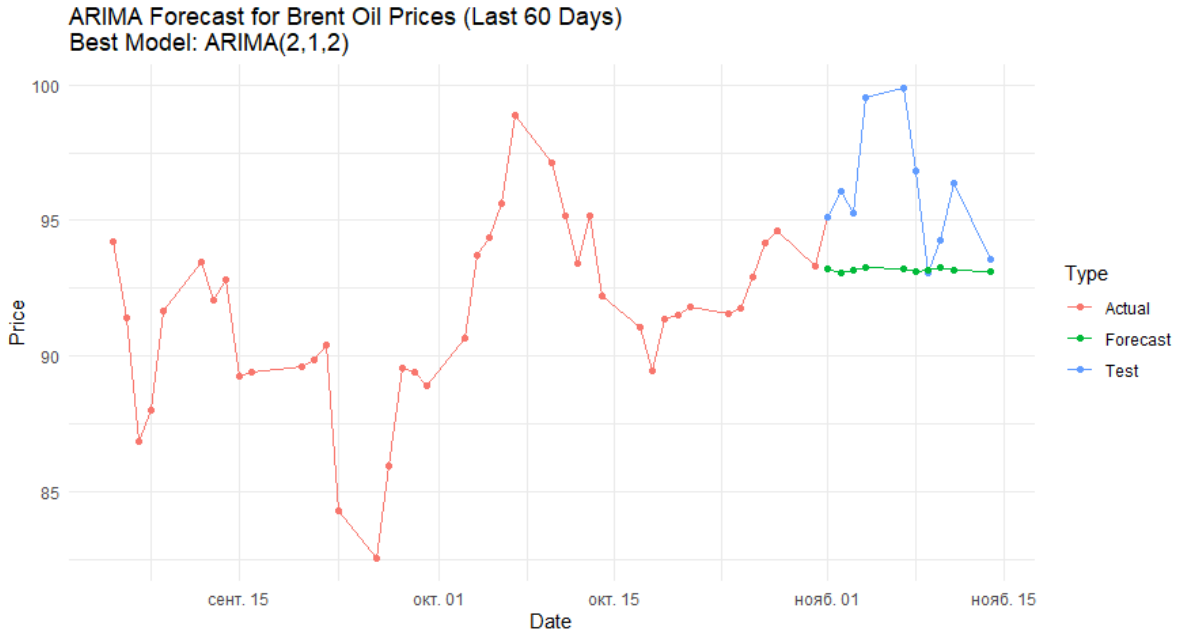


Рисунок 4. Результат прогнозування ціни на нафту на основі моделі ARIMA

Для оцінки точності прогнозів моделі ARIMA були використані три основні метрики: середньоквадратична похибка (MSE), середня абсолютна похибка (MAE) та корінь середньоквадратичної похибки (RMSE). Були отримані такі показники якості моделі ARIMA:

– MSE (Mean Squared Error): 12.59455;
– MAE (Mean Absolute Error): 2.84492;
– RMSE (Root Mean Squared Error): 3.54888.

Високе значення MSE свідчить про те, що помилки прогнозування значні і моделі складно передбачати точні значення цін на нафту. Значення MAE 2.84492 показує, що в середньому модель

помиляється на 2.84 долари при прогнозуванні ціни на нафту. Значення RMSE 3.54888 також свідчить про значні помилки прогнозування.

Графік та метрики показують, що модель ARIMA змогла частково вловити тренди у тестових даних, але її прогнози не були достатньо точними, що підкреслює необхідність використання більш потужних методів, таких як LSTM, для більш точного прогнозування цін на нафту.

Серед причини недостатньої продуктивності моделі ARIMA можна навести:

1. Ціни на нафту характеризуються високою волатильністю і можуть різко змінюватися через різні фактори, такі як політичні події, природні катаклізми або зміни попиту та пропозиції. ARIMA модель може мати труднощі з прогнозуванням таких різких змін.

2. ARIMA модель є лінійною моделлю і не враховує нелінійні залежності між значеннями в часових рядах. Це може призвести до менш точних прогнозів у випадках, коли дані мають складні нелінійні залежності.

3. ARIMA модель вимагає, щоб дані були стаціонарними (мають постійне середнє і дисперсію). Незважаючи на застосування різницювання для досягнення стаціонарності, моделі може бути складно адекватно врахувати довгострокові тренди та сезонність у даних.

Таким чином, результати моделі ARIMA демонструють обмеження лінійних методів для прогнозування складних часових рядів з високою волатильністю та нелінійними залежностями.

4.2. Модель LSTM

LSTM (Long Short-Term Memory) – це різновид рекурентних нейронних мереж (RNN), спеціально розроблений для розв'язання проблеми довгострокової залежності, яка є характерною для традиційних RNN. LSTM-мережі здатні ефективно обробляти часові ряди, зберігаючи інформацію на тривалий період, що робить їх ідеальними для прогнозування часових рядів [13].

Архітектура LSTM:

1. Осередок пам'яті (Memory Cell) – основний компонент LSTM, який зберігає інформацію на довгий час. Осередок пам'яті контролюється трьома основними гейтами (воротами), які дозволяють керувати потоком інформації через нейронну мережу.

2. Гейт забуття (Forget Gate) вирішує, яку частину інформації з осередку пам'яті потрібно забути. Вхід до цього гейту – це комбінація попереднього стану прихованого шару та поточного вхідного сигналу, який проходить через сигмоїдну функцію активації:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f).$$

3. Гейт введення (Input Gate) вирішує, яку нову інформацію потрібно додати до осередку пам'яті. Складається з двох частин: сигмоїдного шару, який вирішує, які значення оновити, та тангенсного шару, який створює нові кандидати на додавання до осередку пам'яті:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i),$$

$$\overline{C_t} = \tanh(W_C * [h_{t-1}, x_t] + b_C).$$

4. Гейт виходу (Output Gate) вирішує, яку частину інформації з осередку пам'яті потрібно вивести. Сигмоїдний шар вирішує, що вивести, а тангенсний шар обмежує значення між -1 та 1:

$$i_t = \sigma(W_0 * [h_{t-1}, x_t] + b_0,$$

$$h_t = o_t * \tanh(C_t).$$

5. Оновлення осередку пам'яті. Новий стан осередку пам'яті формується за допомогою забуття частини старої інформації та додавання нової:

$$C_t = f_t * C_{t-1} + i_t * \overline{C_t}.$$

LSTM може використовуватися для прогнозування часових рядів, оскільки має такі особливості:

– завдяки своїй архітектурі, LSTM може зберігати та використовувати інформацію з попередніх кроків на довгих відстанях у часових рядах, що дозволяє їй ефективно моделювати складні часові залежності4

– на відміну від традиційних RNN, LSTM менш схильна до проблеми градієнтного затухання, що робить її більш стійкою та ефективною при тренуванні на довгих часових рядах;

– LSTM може адаптуватися до різних типів даних та структур часових рядів, що робить її універсальним інструментом для прогнозування;

– завдяки своїй здатності до обробки нелінійних залежностей, LSTM може краще моделювати складні процеси у часових рядах, що забезпечує точніші прогнози порівняно з лінійними моделями, такими як ARIMA.

Таким чином, LSTM є потужним інструментом для прогнозування часових рядів завдяки своїй здатності ефективно обробляти довгострокові та нелінійні залежності, що робить її ідеальною для задач прогнозування цін на нафту марки Brent.

Процес *налаштування гіперпараметрів* є критичним етапом при побудові моделі LSTM, оскільки від правильного вибору гіперпараметрів залежить точність та ефективність моделі [14]. Основні гіперпараметри, які потрібно налаштувати для LSTM, включають кількість блоків (units), кількість епох (epochs), розмір пакета (batch size) та оптимізатор (optimizer).

Кількість блоків (units) LSTM визначає розмір прихованого шару. Більша кількість блоків може дозволити моделі захоплювати більш складні патерни, але також збільшує обчислювальні витрати та ризик перенавчання. Діапазон перевірених значень: 10, 50, 100 блоків.

Кількість епох (epochs) визначає, скільки разів модель пройде через весь тренувальний набір даних. Більша кількість епох може поліпшити точність моделі, але також може призвести до перенавчання. Діапазон перевірених значень: від 1 до 2 епох.

Розмір пакета (batch size) визначає кількість прикладів, що використовуються для одного оновлення ваг під час тренування. Менший розмір пакета може забезпечити більш стабільне оновлення ваг, але збільшує час тренування. Діапазон перевірених значень: 1, 10.

Оптимізатор (optimizer) визначає метод оновлення ваг моделі під час тренування. Вибір оптимізатора може суттєво вплинути на швидкість збіжності та точність моделі. Діапазон перевірених значень: 'adam', 'sgd', 'rmsprop', 'adagrad', 'adadelta', 'adamax', 'nadam', 'ftrl'.

Процес налаштування гіперпараметрів включав ітеративне тестування різних комбінацій гіперпараметрів для визначення кращих налаштувань моделі для прогнозування цін на нафту марки Brent.

Для побудови та навчання моделі LSTM використовувався скрипт на мові програмування R [15]. Використовувались бібліотеки, такі як keras, tensorflow, tidyverse, timetk.

Для оцінки моделі LSTM були використані три основні метрики: MSE, MAE та RMSE. У табл. 2 наведено 5 кращих результатів з цієї таблиці для різних конфігурацій моделі LSTM, відсортовану за показником RMSE:

Таблиця 2. Результати тестування гіперпараметрів моделі

Units	Batch size	Epochs	Optimizer	MSE	MAE	RMSE
50	1	2	adam	0.003056	0.055120	0.055283
10	1	2	adam	0.004808	0.063459	0.069341
100	1	2	nadam	0.006688	0.077512	0.081778
100	1	2	adam	0.009115	0.090693	0.095471

Кращою виявилася модель з конфігурацією: Units: 50, Batch size: 1, Epochs: 2, Optimizer: adam, з найменшим значенням RMSE 0.055283.

Для оцінки та розуміння роботи моделі LSTM були створені декілька графіків, які допомагають проаналізувати продуктивність моделі, помилки прогнозування та відповідність фактичних та прогнозованих значень.

Крива втрат під час навчання (Training Loss Curve) показує, як зменшується значення функції втрат під час навчання моделі на кожній епосі. Графік (рис. 5) демонструє зменшення втрат, що свідчить про покращення моделі під час навчання.

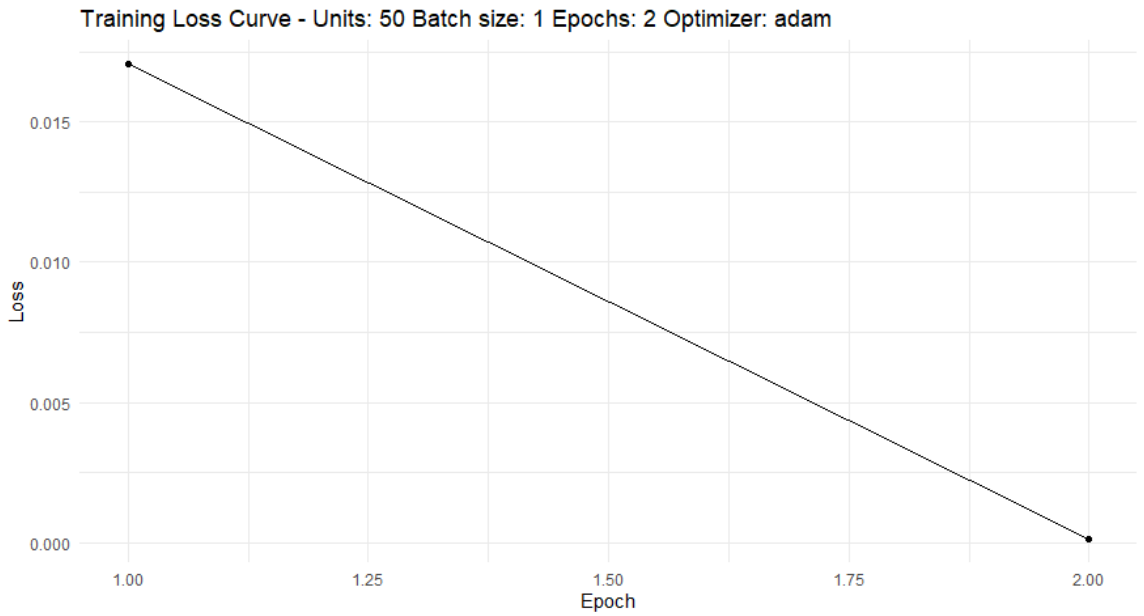


Рисунок 5. Графік кривої втрат

Графік фактичних та прогнозованих значень (Actual vs. Forecasted Values) показує порівняння фактичних значень цін на нафту та прогнозованих значень на основі моделі LSTM. Візуалізація включає прогнози на 100 днів назад (рис. 6), що дозволяє оцінити точність та відповідність моделі.

Графік автокореляцій залишків (ACF of Prediction Errors) показує автокореляцію помилок прогнозування для різних лагів. Він допомагає виявити, чи існує автокореляція між помилками прогнозування, що може вказувати на недоліки моделі. На графіку (рис. 7) видно, що автокореляція помилок швидко зменшується, що вказує на хорошу модель прогнозування.

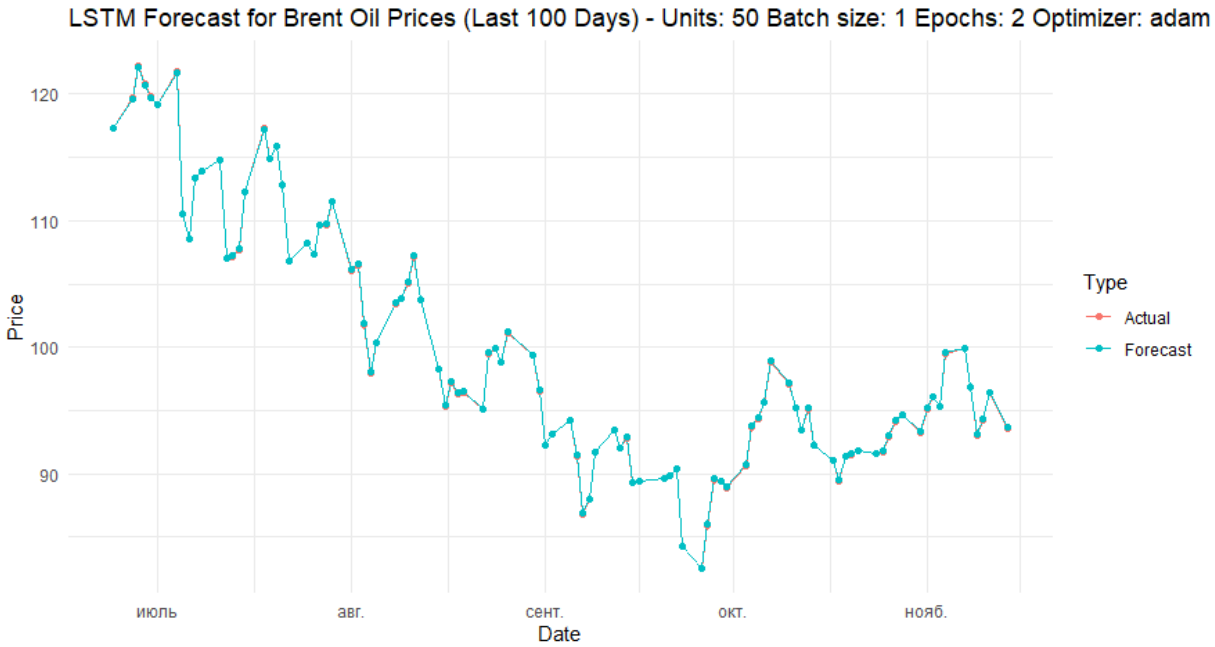


Рисунок 6. Графік прогнозування значень на 100 днів назад

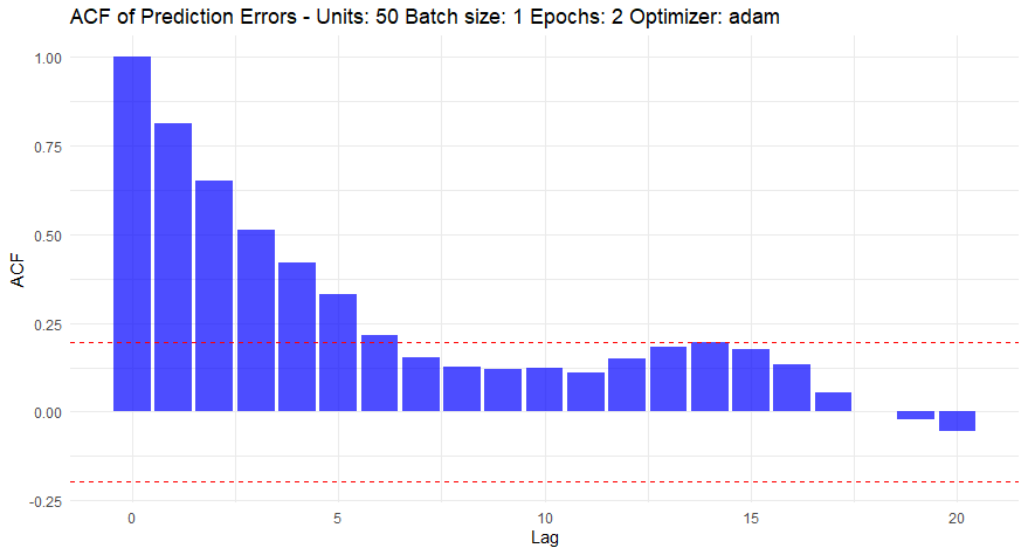


Рисунок 7. Графік автокориляції залишків

На рис. 8 показаний графік прогнозу значень цін на нафту на 100 днів вперед. Він дозволяє оцінити, як модель передбачає майбутні значення на основі історичних даних. Прогнози моделі LSTM на наступні 100 днів демонструють високу точність та відповідність фактичним значенням, що підтверджує ефективність цієї моделі для задач прогнозування.

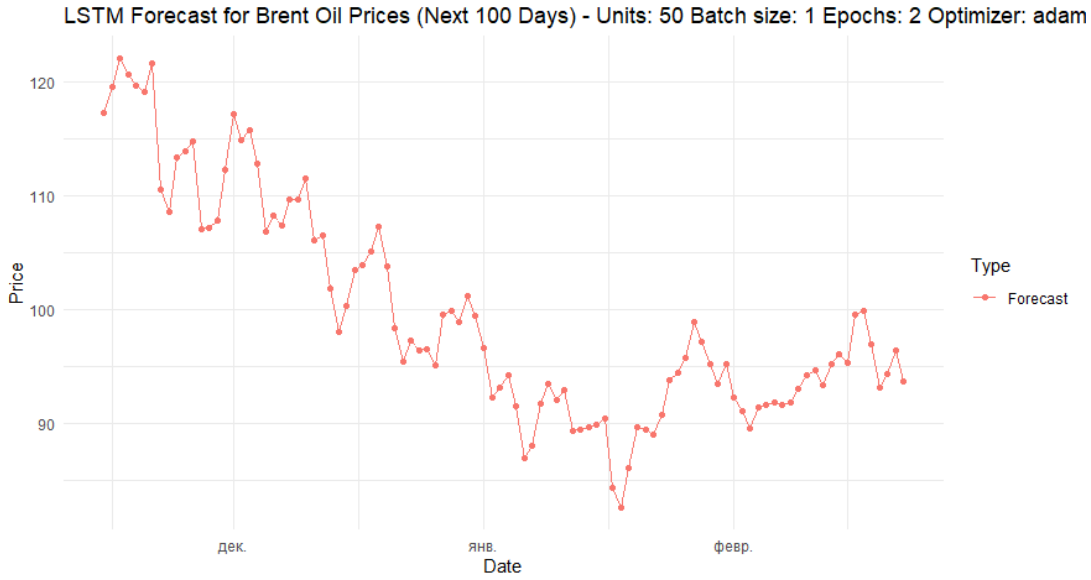


Рисунок 8. Графік прогнозування значень на 100 днів вперед

6. Аналіз результатів

Порівняння продуктивності моделей ARIMA та LSTM було проведено по основним метрикам MSE, MAE та RMSE (табл. 3).

Таблиця 3. Порівняння моделей ARIMA та LSTM

Модель	MSE	MAE	RMSE
ARIMA	12.59455	2.84492	3.54888
LSTM	0.003056	0.055120	0.055283

Модель LSTM має значно менше значення MSE (0.003056) порівняно з моделлю ARIMA (12.59455). Це вказує на те, що модель LSTM має значно менші помилки прогнозування в середньому. Модель LSTM також має значно менше значення MAE (0.055120) порівняно з моделлю ARIMA (2.84492). Це означає, що в середньому помилки прогнозування моделі LSTM є

набагато меншими. І значення RMSE для моделі LSTM (0.055283) є значно меншим порівняно з моделлю ARIMA (3.54888). Оскільки RMSE надає інтерпретоване значення помилки в тих же одиницях, що і прогнозовані значення, можна сказати, що модель LSTM є більш точною.

Результати порівняння моделей показують, що використання сучасних методів глибокого навчання, таких як LSTM, забезпечує значно кращу точність і ефективність прогнозування порівняно з традиційними методами, такими як ARIMA. Це підтверджує необхідність використання більш потужних і адаптивних моделей для складних задач прогнозування у реальних умовах.

На основі порівняння моделей та їх продуктивності, модель LSTM з найкращими показниками точності була обрана як остаточна модель для прогнозування цін на нафту марки Brent. Цей вибір обумовлений значно вищою точністю прогнозів, здатністю обробляти складні нелінійні залежності та адаптацією до високої волатильності ринку. Використання моделі LSTM забезпечує більш надійні та точні прогнози, що є критично важливим для прийняття обґрунтованих рішень у різних галузях економіки та фінансів.

7 Висновки та рекомендації

У роботі було розглянуто процес побудови та оцінки моделей для прогнозування цін на нафту марки Brent з використанням моделей ARIMA та LSTM. На основі проведеного аналізу можна зробити основні висновки:

- Модель ARIMA, хоча і є популярною та широко використовуваною для прогнозування часових рядів, показала недостатньо точні результати для прогнозування цін на нафту марки Brent. Основні обмеження моделі ARIMA включають її лінійність та обмежену здатність обробляти складні, нелінійні залежності у даних.
- Модель LSTM показала значно кращі результати у прогнозуванні цін на нафту порівняно з моделлю ARIMA. Це обумовлено здатністю LSTM моделювати довгострокові залежності та складні патерни у часових рядах. Модель LSTM продемонструвала високу точність прогнозування, адаптацію до волатильності ринку та здатність обробляти нелінійні залежності.

Таким чином, сучасні методи глибокого навчання, такі як LSTM, є значно більш ефективними для прогнозування цін на нафту марки Brent порівняно з традиційними методами, такими як ARIMA. Висока точність прогнозів, здатність моделювати складні залежності та адаптація до волатильності ринку роблять LSTM потужним інструментом для прогнозування у різних галузях економіки та фінансів.

На основі проведеного дослідження та отриманих результатів можна запропонувати наступні рекомендації щодо подальшого покращення моделі LSTM для прогнозування цін на нафту марки Brent, а також напрямків для майбутніх досліджень:

1. Подальше покращення моделі LSTM:

- Провести більш детальний пошук гіперпараметрів, включаючи більший діапазон значень для units, batch size, epochs та різні типи оптимізаторів. Можна використовувати методи, такі як Grid Search або Random Search, для автоматизації цього процесу.
- Додати регуляризацію (наприклад, Dropout) для запобігання перенавчанню моделі. Це допоможе зробити модель більш стійкою та покращити її узагальнення на нових даних.
- Використовувати більш тривалий історичний період для навчання моделі. Збільшення обсягу даних може покращити точність прогнозування, особливо якщо включити дані з різних ринкових умов.
- Включити додаткові ознаки (features), такі як макроекономічні показники, геополітичні події, сезонні фактори тощо. Це може допомогти моделі краще розуміти контекст і робити точніші прогнози.

2. Напрямки для майбутніх досліджень:

- Порівняння з іншими моделями. Порівняти продуктивність моделі LSTM з іншими сучасними моделями глибокого навчання, такими як GRU (Gated Recurrent Unit) або Transformer моделі. Це може допомогти знайти ще більш ефективні підходи до прогнозування.
- Розробити гібридні моделі, які поєднують LSTM з іншими методами машинного навчання або економетричними моделями. Гібридні підходи можуть забезпечити більш точні та стабільні прогнози.

- Аналіз впливу зовнішніх факторів. Дослідити вплив різних зовнішніх факторів на точність прогнозування, таких як зміни в регуляторній політиці, технологічні інновації, зміни в попиті та пропозиції на ринку нафти. Це допоможе краще зрозуміти, як модель реагує на різні умови.
- Працювати над покращенням інтерпретованості моделі LSTM. Це включає розробку методів для пояснення, які фактори впливають на прогнози, що може бути важливим для прийняття бізнес-рішень.
- Впровадити модель у реальне середовище для тестування її продуктивності в реальних умовах. Це дозволить оцінити її практичну цінність та ефективність у прийнятті рішень.

Рекомендації щодо подальшого покращення моделі LSTM та напрямків для майбутніх досліджень дозволяють забезпечити подальший розвиток та вдосконалення методів прогнозування цін на нафту марки Brent. Використання передових технік машинного навчання та глибокого аналізу даних допоможе підвищити точність прогнозів та забезпечити ефективні інструменти для прийняття обґрунтованих рішень у фінансових та економічних галузях.

REFERENCES

1. Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2015). Time Series Analysis: Forecasting and Control (5th ed.). *Wiley.* https://doi.org/10.1002/9781118619193 (дата звернення 12.10.2024)
2. Dozdar Mahdi Ahmed, Masoud Muhammed Hassan and Ramadhan J. Mstafa (2022). A Review on Deep Sequential Models for Forecasting Time Series Data. https://doi.org/10.1155/2022/6596397 (дата звернення 12.10.2024)
3. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735 (дата звернення 12.10.2024)
4. Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. (2018). Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies, 11*(7), 1636. https://doi.org/10.3390/en11071636 (дата звернення 12.10.2024)
5. Nelson, D. M., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN),* 1419-1426. https://doi.org/10.1109/IJCNN.2017.7966019 (дата звернення 12.10.2024)
6. Shahid, F., Zameer, A., & Muneeb, M. (2020). Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM using sequential and statistical data. *Chaos, Solitons & Fractals, 140,* 110212. https://doi.org/10.1016/j.chaos.2020.110212 (дата звернення 12.10.2024)
7. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50,* 159-175. https://doi.org/10.1016/S0925-2312(01)00702-0 (дата звернення 12.10.2024)
8. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE, 13*(3), e0194889. https://doi.org/10.1371/journal.pone.0194889 (дата звернення 12.10.2024)
9. Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting, 36*(1), 75-85. https://doi.org/10.1016/j.ijforecast.2019.03.017 (дата звернення 12.10.2024)
10. Lawrence, M. J., Goodwin, P., O'Connor, M., & Önkal, D. (2006). Judgmental forecasting: A review of progress over the last 25 years. *International Journal of Forecasting, 22*(3), 493-518. https://doi.org/10.1016/j.ijforecast.2006.03.007 (дата звернення 12.10.2024)
11. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). *OTexts.* https://otexts.com/fpp3/ (дата звернення 12.10.2024)
12. Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). A comparison of ARIMA and LSTM in forecasting time series. *IEEE International Conference on Machine Learning and Applications (ICMLA),* 1394-1401. https://doi.org/10.1109/ICMLA.2018.00227 (дата звернення 12.10.2024)

13. Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451-2471. https://doi.org/10.1162/089976600300015015 (дата звернення 12.10.2024)
14. Zhou, Z. H. (2012). Ensemble methods: Foundations and algorithms. *Chapman and Hall/CRC.* https://doi.org/10.1201/b12207 (дата звернення 12.10.2024)
15. Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software, 27*(3), 1-22. https://doi.org/10.18637/jss.v027.i03 (дата звернення 12.10.2024)

Bondarenko Kostiantyn *student of Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

Strilets Viktoriia *Ph.D, associate professor of the Department of Computer Systems and Robotics, Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

Shevchenko Dmytro *PhD student of Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

Forecasting economic indicators using the LSTM model

Relevance. Forecasting economic indicators, particularly oil prices, is critically important for various industries such as energy, finance, manufacturing, transportation, and government policy. Accurate forecasts facilitate informed decision-making and resource optimization. In conditions of high oil price volatility, traditional forecasting methods like ARIMA often do not provide sufficient accuracy, making the use of modern methods such as LSTM necessary.
Objective. The objective of the study is to develop a model for forecasting Brent crude oil prices using the LSTM model and to compare its accuracy with traditional methods, including ARIMA.
Research methods. Two main time series forecasting methods, ARIMA and LSTM, were used in this study. Exploratory data analysis, data preparation, model building, model tuning, and evaluation using MSE, MAE, and RMSE metrics were conducted. Brent crude oil price data was processed and normalized before being fed into the models.
Results. The LSTM model demonstrated significantly higher forecasting accuracy compared to ARIMA. The metrics for LSTM (MSE = 0.003, MAE = 0.055, RMSE = 0.055) significantly outperform the corresponding values for ARIMA (MSE = 12.59, MAE = 2.84, RMSE = 3.55). LSTM better handles nonlinear dependencies and data volatility, making it an optimal choice for long-term forecasting.
Conclusions. The use of LSTM for forecasting economic indicators, particularly oil prices, is more effective compared to traditional methods. The model demonstrates the ability to accurately model complex dependencies and adapt to market volatility, making it a reliable tool for forecasting in modern conditions.
***Keywords**: economic indicator forecasting, LSTM, ARIMA, oil prices, machine learning, time series, volatility.*

Надійшла у першій редакції 10.08.2024, в останній - 01.10.2024.
The first version has been received on 10.08.2024, the final version - on 01.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-03

УДК (UDC) 004.8

Волинець Катерина Андріївна *студентка ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: kateryna.volynets@student.karazin.ua
https://orcid.org/0009-0003-7661-9758

Стрілець Вікторія Євгенівна *к.т.н., доцент кафедри комп'ютерних систем та робототехніки, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: viktoria.strilets@karazin.ua
https://orcid.org/0000-0002-2475-1496

Яковлев Данило В'ячеславович *студент ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 6, Харків, Україна, 61022*
e-mail: yakovlev2020ki11@student.karazin.ua
https://orcid.org/0009-0005-4785-6361

Модель класифікації спам-повідомлень у медичних інформаційних системах

Актуальність. У сучасних медичних інформаційних системах щодень генерується значна кількість текстових записів від пацієнтів, лікарів та персоналу. Для якісної роботи такі систем потребують впровадження моделей і методів аналізу і класифікації текстових даних, зокрема виявленню спамових повідомлень і їх блокуванню. Тому розробка, удосконалення і впровадження моделей і методів класифікації спам-повідомлень є актуальним завданням.
Мета дослідження: покращення якості моделей класифікації спам-повідомлень, що дозволить з високою вірогідністю виявляти і фільтрувати небажані повідомлення у медичних інформаційних системах.
Методи дослідження: методи обробки природної мови, моделювання, машинне навчання, методи класифікації, методи аналізу даних, статистичні методи.
Результати. Були побудовані моделі класифікації спам-повідомлень із використанням таких методів машинного навчання як модель логістичної регресії, модель наївного Баєйсівського класифікатора та модель опорних векторів. Для навчання моделей використаних набір SMS Spam Collection, попередньо підготовлений із використанням CountVectorizer та TF-IDFVectorizer. Усі запропоновані моделі показали високу точність у класифікації спам-повідомлень.
Висновки: розроблені моделі класифікації повідомлень на основі машинного навчання та nlp-підходу успішно визначають небажані повідомлення. Кращою за показниками якості виявилася модель на основі методу опорних векторів з TF-IDF векторизацією, оскільки вона показала найвище значення точності (98.75%) та високу повноту (90.3%) класифікації. Подальші вдосконалення моделей та розширення навчального набору можуть сприяти подальшому покращенню якості розпізнавання спаму.

Ключові слова: *спам-повідомлення, медичні інформаційні системи, машинне навчання, обробка природної мови, класифікація текстових даних.*

Як цитувати: Волинець К. А., Стрілець В. Є., Яковлев Д. В. Модель класифікації спам-повідомлень у медичних інформаційних системах. *Вісник Харківського національного університету імені В.Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.25-31. https://doi.org/10.26565/2304-6201-2024-64-03

How to quote: Volynets K. A., Strilets V. Y., Yakovlev D. V. "The spam-messages classification model in a medical information system". *Bulletin of V.N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp.25-31, 2024. https://doi.org/10.26565/2304-6201-2024-64-03[in Ukrainian]

1 Вступ

У сучасному інформаційному суспільстві обробка великих обсягів текстових даних стає все більш важливим завданням для багатьох галузей, зокрема у сфері медичних послуг. Щодня генерується значна кількість текстових повідомлень у медичних інформаційних системах (МІС), включаючи запити пацієнтів, медичні записи та результати аналізів. Для ефективної роботи таких систем необхідно впровадження автоматизованих методів аналізу та класифікації текстових даних.

Одним із потужних інструментів для вирішення цієї задачі є методи обробки природної мови (NLP), які дозволяють не тільки класифікувати текстові повідомлення, але й виявляти спам або інші нерелевантні дані. На відміну від штучних мов, таких як мови програмування та математичні нотації, природні мови розвивалися в міру переходу від покоління до покоління, і їх важко визначити чіткими правилами [1].

Застосування NLP у медичних інформаційних системах дозволяє не тільки автоматизувати обробку запитів, але й забезпечити високу точність та швидкість аналізу даних, що сприяє поліпшенню якості медичних послуг. Проте, залишається низка проблем, пов'язаних із специфікою медичних текстів, таких як складність медичної термінології та необхідність врахування контексту. Це підкреслює актуальність досліджень у галузі класифікації текстових даних з використанням NLP.

2 Задачі обробки природної мови

Серед ключових праць у галузі NLP для медицини важливе місце займають роботи, присвячені автоматичному узагальненню або виокремленню інформації з електронних медичних записів, що сприяє покращенню управління даними пацієнтів [2]. NLP-моделі в цих роботах показали високу точність у задачах автоматизації обробки результатів аналізів та призначень лікарів.

Однією з основних задач NLP у МІС є класифікація текстових даних, таких як запити пацієнтів і лікарські рекомендації. Методи машинного навчання, як показано у [3], демонструють високу ефективність при класифікації великих обсягів тексту, але потребують ретельної підготовки даних через складність медичної термінології.

Дослідження [4] зосередили увагу на використанні методів опорних векторів (SVM) і байєсових класифікаторів для класифікації медичних повідомлень, підкреслюючи як переваги цих підходів, так і труднощі, зумовлені неоднозначністю медичної мови.

Останні наукові досягнення, такі як [5], продемонстрували ефективність трансформерних моделей, таких як BERT, для класифікації медичних текстів завдяки їхній здатності враховувати контекст. Проте, проблеми, пов'язані з обробкою медичних текстів, які часто містять абревіатури та спеціальні терміни, залишаються актуальними [6].

У роботі розглядається *задача* створення моделі класифікації спам-повідомлень, які надходять до медичних інформаційних систем, із використанням методів машинного навчання, які добре зарекомендували себе в цьому напрямку.

Метою роботи є покращення якості моделей класифікації спам-повідомлень, що дозволить з високою вірогідністю виявляти і фільтрувати небажані повідомлення у медичних інформаційних системах.

3 Методи класифікації текстових даних

Серед методів машинного навчання для розв’язання класифікації текстових даних застосовують такі методи:

1. Метод опорних векторів (SVM) є одним з найпопулярніших методів машинного навчання для класифікації тексту. Він працює шляхом пошуку оптимальної гіперплощини, яка найкращим чином розділяє дані у просторі ознак. SVM добре справляється з наборами даних з великою кількістю ознак та може ефективно працювати з текстовими даними;

2. Байєсівські методи класифікації використовують теорему Байєса для прогнозування класу тексту на основі ймовірностей. Найпоширенішими варіантами байєсівських методів є наївний байєсівський класифікатор (Naive Bayes Classifier), який вважає, що всі ознаки у тексті незалежні між собою при умові класифікації. Цей метод простий у реалізації та зазвичай добре працює для наборів даних з великою кількістю ознак;

3. Нейронні мережі є потужними інструментами для класифікації тексту. Вони можуть ефективно використовуватися для вирішення різноманітних завдань, таких як класифікація

тональності, визначення сутностей тощо. Два основних типи нейронних мереж, які часто використовуються для класифікації текстів:

– згорткові нейронні мережі (CNN) використовуються для аналізу послідовностей даних, таких як текст. Вони здатні автоматично виявляти важливі ознаки у тексті, що робить їх ефективними для класифікації;

– рекурентні нейронні мережі (RNN) працюють з послідовностями довільної довжини та зберігати інформацію про попередні стани. Це робить їх добрим вибором для аналізу тексту, оскільки вони здатні врахувати контекст інформації.

Для дослідження були обрані такі методи машинного навчання, як: логістична регресія, наївний байєсівських класифікатор та метод опорних векторів.

Оскільки у дослідженні розглядається задача класифікації тестових повідомлень, то для створення ефективних моделей варто застосовувати також лексичні методи NLP. Лексичні методи зосереджені на аналізі і обробці окремих слів або токенів у тексті. Вони базуються на лексичних властивостях мови, таких як частота вживання слів, визначення частин мови тощо. До основних лексичних методів відносять:

- токенізацію (tokenization) – процес розбиття тексту на окремі слова або токени. Більшість моделей обробки текстових даних спирається на попередньо виокремлені або позначені слова з тексту, що аналізується [3];

- лематизацію (lemmatization) – процес перетворення слова на його базову форму або лему. Наприклад, слово "running" буде перетворено на "run";

- стемінг (stemming) – відсічення афіксів зі слова, залишаючи лише його корінь. Наприклад, слово "beginning" може бути зведене до "begin".

4 Розробка моделей класифікації спам-повідомлень

4.1 Збір і попередня обробка набору даних

Під час збору та обробки медичних даних для класифікації виникають специфічні проблеми, пов'язані з медичними текстами. Наприклад, медичні дані часто містять спеціалізовані терміни, абревіатури та скорочення, що ускладнює автоматичне розпізнавання та класифікацію. Неоднозначність медичної термінології та її чутливість до контексту вимагає більш ретельної попередньої обробки, включаючи лематизацію та видалення стоп-слів.

Для класифікації текстових даних важливим етапом є підготовка якісного набору даних. Для класифікації спам-повідомлень можна використати SMS Spam Collection Dataset [8], що містить записи текстових повідомлень, позначених як "спам" або "не спам" (рис. 1).

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Рисунок 1 – Фрагмент набору даних

Під час попередньої обробки були використані:

– очищення даних – видалення непотрібних стовпців, видалення пустих записів, перевірка наявності пропущених значень;

– лематизація – приведення слів до початкової форми, видалення стоп-слів, токенізація [7] (розбиття тексту на слова), перетворення тексту в нижній регістр для узгодження.

Також попередньо була проаналізована збалансованість між класами, оскільки нерівномірний розподіл даних між класами може впливати на якість побудованих моделей. Виявилося, що у

наборі даних спостерігається нерівномірний розподіл між класами: 4825 повідомлень позначені як «ham» (не спам) і 747 як «spam» (спам) (рис. 2).

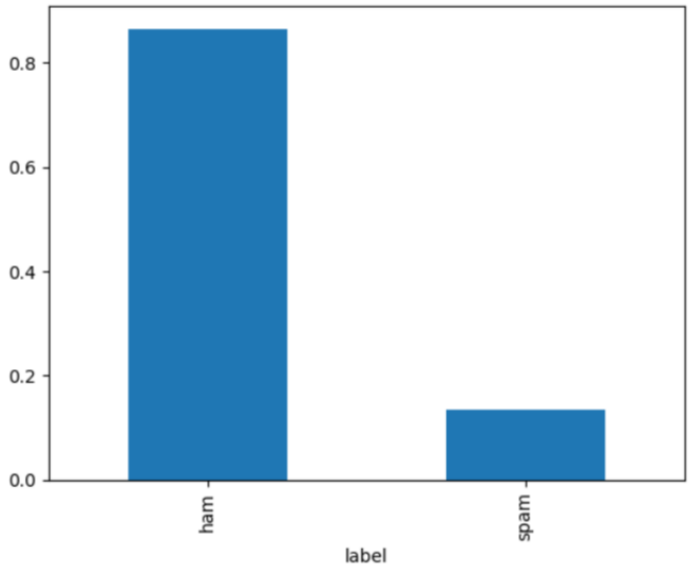


Рисунок 2 – Розподіл класів

Нерівномірність розподілу класів враховувалась під час побудови моделей та аналізі їх якості.

4.2 Створення наборів ознак

Для побудови моделей машинного навчання текстові дані необхідно перетворити у числові вектори. У досліджені були використані:

– CountVectorizer, який перетворює текст у матрицю кількостей, де кожен стовпець відповідає слову з тексту, а кожен рядок — документу (текстовому повідомленню). Кожен елемент матриці вказує кількість разів, коли слово зустрічається у документі;

– TF-IDF (Term Frequency-Inverse Document Frequency) – підхід, який враховує не лише частоту слова у документі, але й його значущість у наборі текстів. Він забезпечує більш точне представлення тексту для класифікації.

За кожним з методів векторизації було побудовано набір даних для навчання і тестування моделей класифікації.

4.3 Створення моделей класифікації та їх налаштування

Для класифікації текстових даних було побудовано декілька моделей машинного навчання: модель логістичної регресії, модель наївного Баєйсівського класифікатора та модель опорних векторів (SVM). Кожна модель була навчена на двох різних наборах ознак: з CountVectorizer та з TF-IDFVectorizer.

Процес налаштування моделей включав підбір гіперпараметрів, таких як регуляризаційні параметри для логістичної регресії та параметр ядра для SVM. Налаштування здійснювалось за допомогою крос-валідації або пошуку по сітці.

5 Аналіз якості побудованих моделей класифікації

Результати на тестовому наборі даних є основною метрикою оцінки продуктивності моделі. Проте також важливо оцінювати результати на навчальному та валідаційному наборах для перевірки наявності перенавчання або недонавчання. Розмір тренувальної і тестової вибірок був 3733 і 1839 відповідно.

Якість моделей на тестовому наборі була оцінена за метриками accuracy, precision, recall (табл. 1).

Таблиця 1. Оцінки якості моделей класифікації

Модель	Accuracy	Precision	Recall
Логістична регресія з CountVectorizer	0.98205	0.99038	0.86919
Логістична регресія з TF-IDF	0.96628	0.99435	0.74261
Наївний Байєс з CountVectorizer	0.98586	0.95670	0.93248
Наївний Байєс з TF-IDF	0.96954	0.99453	0.76793
Метод опорних векторів з CountVectorizer	0.98096	0.97641	0.87341
Метод опорних векторів з TF-IDF	0.98749	1.0	0.90295

Аналізуючи отримані значення показників можна побачити, що модель на основі методу опорних векторів з TF-IDF має найвищий показник точності (98.75%) і precision (100%), показник recall (90.30%) є одним з кращих порівняно з іншими моделями.

Усі розроблені моделі були перевірені на розпізнавання спаму для конкретних повідомлень (рис. 3, 4).

```
Enter message for classificationIs that seriously how you spell his name?
The message is classified as ham.
```

Рисунок 3 – Визначення повідомлення як такого, що не є спамом

```
Enter message for classificationHi! Do you want to earn money? Just call me and claim your reward!
The message is classified as spam.
```

Рисунок 4 – Визначення повідомлення як спам

Протестовані моделі показали здатність до вірної класифікації спам-повідомлень. Але, оскільки початкових набір містив незбалансовані класи, то точність визначення саме спаму буде залежати від подібності аналізованих повідомлень до зразків навчального набору.

6 Розробка альтернативних моделей

Оскільки отримані результати показали високу якість моделей класифікації, було прийняте рішення також побудувати моделі на основі ансамблевих методів Gradient Boosting Classifier та Random Forest Classifier.

Отримані результати (рис. 3) свідчать про те, що моделі Gradient Boosting та Random Forest мають дуже високу якість, але їхні показники різняться в залежності від типу векторизації (CountVectorizer або TF-IDF).

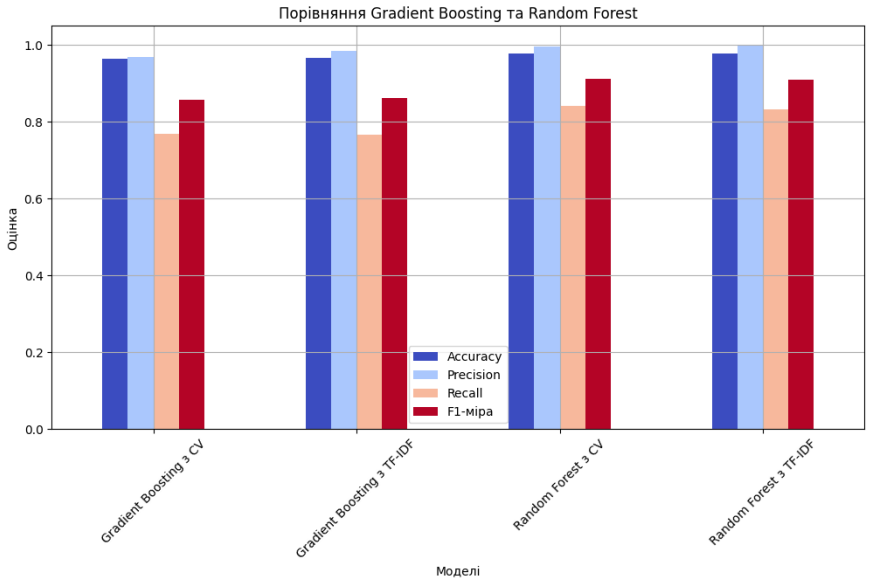


Рисунок 3 – Порівняння якості ансамблевих моделей

Random Forest з TF-IDF демонструє найкращі результати з точки зору точності, але порівнюючи результати, можна зробити висновок, що модель на основі методу опорних векторів (SVM) з TF-IDF показує трохи кращі результати.

Gradient Boosting та Random Forest зазвичай краще працюють на великих наборах даних, тому модель на основі методу опорних векторів з TF-IDF є найефективнішою для задачі класифікації з невеликою кількістю спам-повідомлень.

7 Висновок

У результаті дослідження були побудовані моделі класифікації на основі методів машинного навчання, які мають високі показники якості. Було успішно реалізовано процес класифікації текстових повідомлень з використанням моделей на основі логістичної регресії, наївного Баєйсівського класифікатора та метоуд опорних векторів (SVM). Застосування таких інструментів, як CountVectorizer і TF-IDF, дозволило точно та швидко перетворити текстові дані у числові ознаки, що сприяло підвищенню якості класифікації. Оцінка моделей за метриками якості показала, що метод опорних векторів у поєднанні з TF-IDF демонструє найвищу ефективність.

Моделі на основі CountVectorizer мали вищу F1-міру в порівнянні з моделями на основі TF-IDF:

- F1-міра для CountVectorizer: 0.9211;
- F1-міра для TF-IDF: 0.8208;
- Середня точність перехресної валідації (CountVectorizer): 0.9769.

Це свідчить про те, що CountVectorizer забезпечує більш збалансовану продуктивність між прецизійністю та реколом, що може бути важливим у реальних умовах для уникнення надмірного кількості помилкових позитивних або негативних результатів.

REFERENCES

1. Steven Bird, Ewan Klein, Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media Inc. 2009. 504 p.
2. Wang, M., Sun, Z., Jia, M. et al. Intelligent virtual case learning system based on real medical records and natural language processing. *BMC Med Inform Decis Mak* 22, 60 (2022). https://doi.org/10.1186/s12911-022-01797-7.
3. Robert M. Cronin, Daniel Fabbri, Joshua C. Denny, S. Trent Rosenbloom, Gretchen Purcell Jackson. A comparison of rule-based and machine learning approaches for classifying patient portal messages. *International Journal of Medical Informatics*, Vol. 105, P. 110-120, 2017. https://doi.org/10.1016/j.ijmedinf.2017.06.004
4. Elbattah M., Arnaud É., Gignon M., Dequen G. The Role of Text Analytics in Healthcare: A Review of Recent Developments and Applications. *Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies* (*BIOSTEC 2021*), Volume 5: HEALTHINF, P. 825-832, 2021. DOI: 10.5220/0010414508250832.
5. Turchin Alexander, Masharsky Stanislav, Zitnik Marinka. Comparison of BERT implementations for natural language processing of narrative medical documents. *Informatics in Medicine Unlocked*, 36, 2022. DOI: 101139. 10.1016/j.imu.2022.101139.
6. Zhou Binggui, Yang Guanghua, Shi Zheng, Ma Shaodan. Natural Language Processing for Smart Healthcare. *IEEE Reviews in Biomedical Engineering*, 2021. DOI: 10.48550/arXiv.2110.15803.
7. Jurafsky Daniel, Martin James H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd edition. Prentice Hall, 2019, 621 p. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf (Last accessed: 20.11.2024)
8. Almeida T., Hidalgo J. SMS Spam Collection [Dataset]. *UCI Machine Learning Repository*. 2011. https://doi.org/10.24432/C5CC84.

Volynets Kateryna *student of Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

Strilets Viktoriia *Ph.D, associate professor of the Department of Computer Systems and Robotics, Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

Yakovlev Danylo *student of Education and Research Institute of Computer Sciences and Artificial Intelligence, V.N. Karazin Kharkiv National University, 6 Svobody sq., Kharkiv, Ukraine, 61022*

The spam-messages classification model in a medical information system

Relevance. In modern medical information systems, a significant number of text records are generated daily from the service, doctors and staff. For high-quality work, such systems require the implementation of models and methods for analyzing and classifying text data, in particular, detecting spam messages and blocking them. Therefore, the development, improvement and implementation of models and methods for classifying spam messages is a relevant task.
Research objective: increasing the efficiency of the spam message recognition process in medical information systems; developing and implementing spam classification models based on machine learning methods.
Research methods: natural language processing methods, modeling, machine learning, classification methods, data analysis methods, statistical methods.
Results. Spam message classification models were built using such machine learning methods as the logistic regression model, the national Bayesian classifier model and the support vector model. The SMS Spam Collection set, previously prepared using CountVectorizer and TF-IDFVectorizer, was used to train the models. All proposed models showed high accuracy in spam message classification and the ability to correctly determine the type of message.
Conclusions: The developed message classification models based on machine learning and nlp approach successfully generated unwanted messages. The best model for quality indicators was the model based on the support vector method with TF-IDF vectorization, after which it showed the highest accuracy value (98.75%) and high value of recall (90.3%) of classification. Further improvements of the models and expansion of the training set can contribute to further improvement of the quality of spam recognition.

Keywords: *spam messages, medical information systems, machine learning, natural language processing, text data classification.*

Надійшла у першій редакції 27.08.2024, в останній - 05.10.2024.
The first version has been received on 27.08.2024, the final version - on 05.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-04

УДК (UDC) 519.216

Danilevskyi Mykhailo *PhD student; V.N. Karazin Kharkiv National University, Svobody Square, 4, Kharkiv-22, Ukraine, 61022.*
e-mail: m.danilevskiy@gmail.com
https://orcid.org/0009-0000-0030-2218

Yanovsky Volodymyr *Doctor of Physical and Mathematical Sciences, professor; V. N. Karazin Kharkiv National University, sq. Svobody 4, Kharkiv, Ukraine, 61000; Institute of Single Crystals, National Academy of Sciences of Ukraine, Nauki Ave. 60, Kharkiv, Ukraine, 61001*
e-mail: yanovsky@isc.kharkov.ua
https://orcid.org/0000-0003-0461-749X

Detecting Telephone Subscribers with Abnormal Behaviour Through Network Properties Analysis

Abstract. The use of telephone subscriber networks for fraud, sales of goods and services, and spam leads to annual financial losses of billions of dollars worldwide. The problem was studied back in 1996 and is still relevant today, despite many methods of counteraction and protection. Traditional methods, such as blocking lists and call frequency monitoring, are often ineffective against spammers who bypass these systems by changing their behaviour patterns. **Objective.** The purpose of the work is to study the use of subscriber network properties, such as clustering coefficients, centrality, and average shortest path length, as criteria for identifying subscribers with abnormal behaviour in a dynamic telephone network. **Research methods.** Modeling and numerical experiment. **Results.** The study shows that the global clustering coefficient is a sensitive measure for detecting the presence of spammers in the network. Its value decreases significantly when spammers appear. When classifying subscribers into normal and spammer using the Random Forest model, the most important properties are the local clustering coefficient, average shortest path length, degree and centrality of the subscriber in the network. According to the telephone network modeling data, it was found that with a measurement window size of 4 days, the classifier accuracy (F1 score) and the accuracy of detecting spammers (TPR) reached values of 80%. **Conclusions.** Using the network characteristics of subscribers has a positive effect on the accuracy of detecting subscribers with abnormal behaviour, but it takes time for spammers and normal subscribers to become distinguishable by network characteristics.

Keywords: *malicious phone calls, mobile call graph, telephone spam detection, telephone network, lognormal distribution, degree distribution, clustering coefficient, average shortest path length*

Як цитувати: Danilevskyi M., Yanovsky V. Detecting Telephone Subscribers with Abnormal Behaviour Through Network Properties Analysis. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.32-39. https://doi.org/10.26565/2304-6201-2024-64-04

How to quote: M. Danilevskyi, V. Yanovsky "Detecting Telephone Subscribers with Abnormal Behaviour Through Network Properties Analysis", *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 32-39, 2024. https://doi.org/10.26565/2304-6201-2024-64-04

1 Introduction

Malicious phone calls, encompassing both spam and scams, pose a significant global problem, resulting in billions of dollars in financial losses each year. This issue has been a subject of study since at least 1996 [1] and remains pressing today. Mobile phone spam involves the exploitation of telecommunications networks to execute fraudulent or distracting activities. It is particularly problematic for network providers and users, leading to substantial financial losses and damages. One of the critical challenges in addressing mobile phone spam is the dynamic nature of malicious caller behaviour, which frequently changes and adapts to detection efforts. Although various countermeasures have been implemented, a comprehensive and effective solution to eradicate these activities has yet to be developed. For example, countries such as the United States have introduced measures like the National Do Not Call Registry [2] to mitigate the issue, with fines for violations reaching millions of dollars [3].

Numerous technical approaches have been developed to identify users with abnormal activity within social and, specifically, telephone networks who engage in fraudulent activities or disturb normal users through unsolicited advertisements [4-14]. One common method involves gathering reports from users via mobile applications to generate "blacklists" of known offenders [15]. Another approach includes tracking user behavioural characteristics, such as the number of calls, call timing, duration, and the occurrence of international calls, to create user profiles based on these data points [1, 2, 16]. Additionally, methods of call content analysis – examining both spoken words [17] and the acoustic properties of the voice [18] – have been employed to detect malicious subscribers. Topological features, such as the degree of connectivity, the number of groups a user participates in, interconnections among friends, and the average number of connections within a user's community, have also been utilized in identifying suspicious network users, as demonstrated by the authors of [19]. Another technique proposed in [20] uses the Graph-Based Anomaly Detection (GBAD) system to represent call data as a graph, thus enabling the identification of anomalous structures that deviate from the patterns of normal users. In [21] and [22], neural networks were applied to detect users exhibiting anomalous behaviour. Despite the diverse strategies for identifying malicious subscribers, the issue remains significant and unresolved.

In this paper, we examined potential criteria for identifying subscribers with abnormal behaviour within a mobile telephone network. A computer simulation was developed to replicate a dynamic network of telephone subscribers. The modeling data were used to analyze various properties that characterize subscriber behaviour within the network. Among the indicators evaluated were degree distribution, clustering coefficient, betweenness centrality, average shortest path length, and others. These modeling results facilitated the identification and assessment of network properties that define phone user behaviour. The findings indicate that the global clustering coefficient is effective in detecting the presence of spammers, while properties such as local clustering coefficient, average shortest path length, degree, and betweenness centrality are crucial for subscriber classification.

This approach enables malicious users who may evade traditional spam filters based on call frequency or block lists, including those who have remained inactive for extended periods, frequently change numbers, or exhibit irregular behaviour patterns.

2 Modeling

In this study, we employ a modified version of the dynamic network model of telephone subscribers, as compared to the model described in [27]. The network model consists of nodes representing subscribers and edges representing calls or links between them. A link is established whenever two subscribers engage in a call. The model accounts for variations in call frequency, with typical subscribers making 5 to 7 calls per day, while users exhibiting higher calling activity, such as spammers, may place between 200 and 300 calls daily. The network evolves dynamically over time, with new connections forming and existing connections breaking based on ongoing and completed calls. The dataset generated from the model provides information about calls over a period, enabling an analysis of user properties. In this model, subscribers initiate calls to others on their contact list with a probability defined by a model parameter. Normal users' contact lists contain other normal telephone subscribers such as family members, friends, colleagues etc. Spammers do not have such contact lists and make random calls to a wide range of users. The number of calls follows a lognormal distribution [23]. Additionally, the model assumes that subscribers cannot make or receive other calls during an ongoing call. The duration of each call is determined by the probability of its termination at any given moment, which is also defined by a model parameter.

The model parameters include the total number of subscribers, the parameters of the lognormal distribution for simulating the daily number of calls, the duration of the experiment, the probability of a call ending, the proportion of calls made to contacts from the contact list and their size.

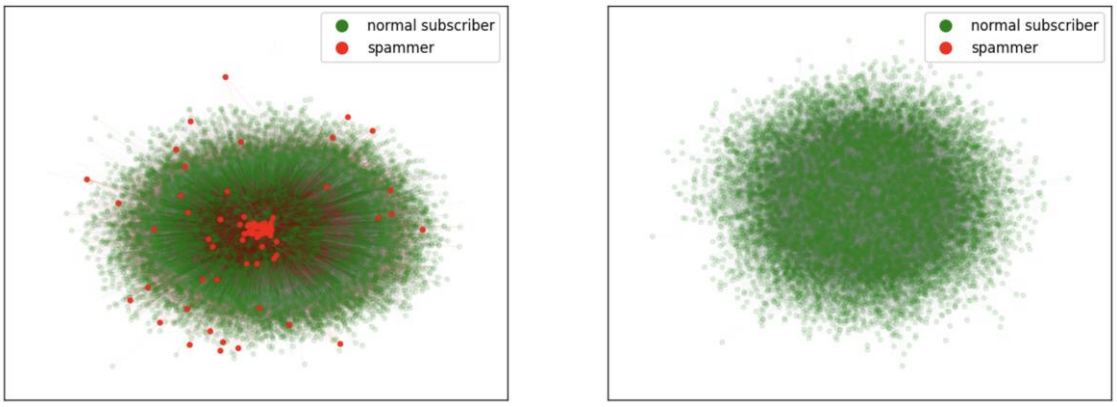
In the numerical experiments, various network properties of subscribers, or network nodes, were measured: node degree, local and global clustering coefficients, node centrality, average shortest path length between subscribers, PageRank [24], and the average number of calls per day [25, 26]. The degree of a node in a graph is defined as the number of edges connected to that node [25], indicating how many other nodes are directly connected to it. The clustering coefficient quantifies how connected a node's neighbours are in the graph [26]. This measure is expressed in two forms: the local clustering coefficient and the global clustering coefficient. Both provide insight into the degree of clustering in the network but at different scales. The local clustering coefficient of a node measures the proportion of

pairs of a node's neighbours that are directly connected to each other, relative to the total number of possible connections among those neighbours. In contrast, the global clustering coefficient measures the general tendency of the network to form closely related groups or clusters. The average shortest path length is defined as the average number of steps along the shortest paths between all possible pairs of nodes in the graph [25].

The employed model enables the analysis of subscriber behaviours and interactions through key network properties. By measuring network properties, we can identify patterns indicative of abnormal behaviour. This approach offers valuable insights for detecting spammers or subscribers with abnormal activity.

3 Detecting the presence of spammers within the network

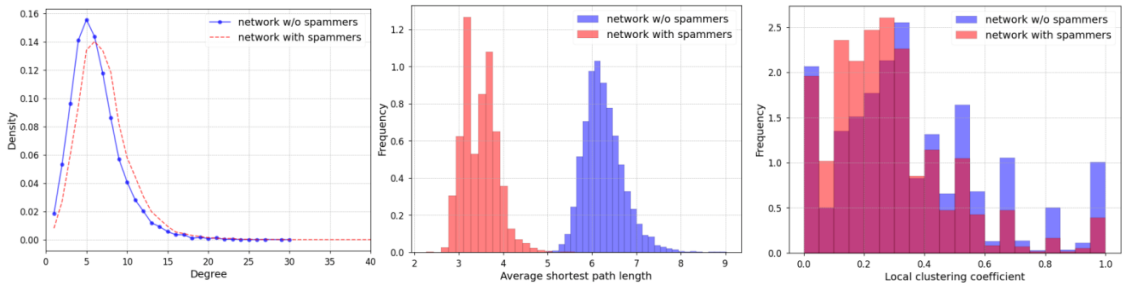
Detecting the presence of spammers in a dynamic telephone network involves analyzing how their behaviour influences key structural properties of the network. In this study, we simulated a dynamic telephone network for one day, consisting of 10,000 subscribers, with 1% (100) of them being spammers. Normal subscribers made 85% of calls to the people in their contact list, approximating real-world network conditions as described in [27]. Meanwhile, spammers called randomly. Based on the simulation data, two networks were created: one containing both normal subscribers and spammers (Fig 1.a), and the other consisting solely of normal subscribers (Fig 1.b).



a) *b)*

Fig. 1 Simulated telephone subscriber networks:
a) with spammers and normal subscribers, b) with normal subscribers

To detect the presence of malicious subscribers in the network, metrics such as node degree, global and local clustering coefficients, and average shortest path length were utilized. Based on the modeling data and the resulting networks, we constructed distributions for the analyzed metrics (Fig. 2) and computed their statistical characteristics (Table 1).



a) *b)* *c)*

Fig.2 Distributions of a) node degrees; b) average shortest path length;
c) local clustering coefficient. Blue color corresponds to the characteristics of a network without spammers, and red color corresponds to a network with spammers.

Table 1 – Characteristics of the studied networks

Network	Degree		Shortest path length		Clustering (local)		Clustering (global)
	mean	sd	mean	sd	mean	sd	
with spammers	8.12	27.06	3.48	0.39	0.28	0.20	0.02
without spammers	6.24	3.14	6.26	0.45	0.36	0.25	0.29
difference	1.88	-	-2.78	-	-0.08	-	-0.28
	23%	-	-80%	-	-28%	-	-1579%

The average degree is higher in the network with spammers (8.12) compared to the network without them (6.24). These users tend to connect to a large number of other nodes, often through actions like sending messages or making calls to numerous users, which increases their degree. A significantly higher average degree or the presence of nodes with abnormally high degrees may indicate the presence of spammers. However, not all of them may be active, and their behaviour can be difficult to distinguish from that of normal subscribers based solely on node degree. The average shortest path length is considerably shorter in the network containing spam entities (3.48) than in the network without them (6.26). By establishing numerous connections, these subscribers create shorter paths across the network, thereby reducing the overall shortest path length. This reduction suggests that the network is becoming more tightly connected, but in a potentially anomalous way, indicative of spam-like behaviour. The local clustering coefficient is slightly lower in the network with spammers (0.28) compared to the network without them (0.36). Such subscribers tend to connect to many unrelated nodes, resulting in a lower local clustering coefficient. This suggests that calls in the network are more random and less structured, resembling a pattern typical of abnormal activity. The global clustering coefficient is significantly lower in the network with spammers (0.02) compared to the network without them (0.29). They disrupt clustering by establishing connections across the network in a broad, random manner, which lowers the global clustering coefficient. A sharp decrease in this metric may serve as a strong indicator of abnormal users in a network, as it reflects a breakdown in the natural clustering and organization of the network.

Although node degree is the most straightforward metric to calculate, its usefulness is limited when spammers frequently change numbers. The shortest path length, while effectively distinguishing between networks with and without spammers, is computationally expensive when calculated for all pairs of subscribers. The local clustering coefficient is also sensitive to the presence of spammers but exhibits larger standard deviations. In contrast, the global clustering coefficient is more sensitive and computationally less expensive than the average shortest path length, making it the most suitable indicator for detecting users with abnormal behavour in a telephone subscriber network.

To test the sensitivity of the global clustering coefficient to the presence of spammers, we simulated their appearance in the network over time. The simulation was conducted with the same parameters as before for a 15-day period, with spam-like activity occurring from the fifth to the eleventh day. Using the obtained data, we calculated the global clustering coefficient for each day and plotted its dynamic behaviour over time (Fig. 3).

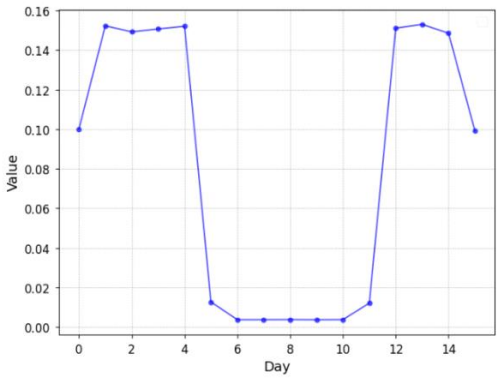


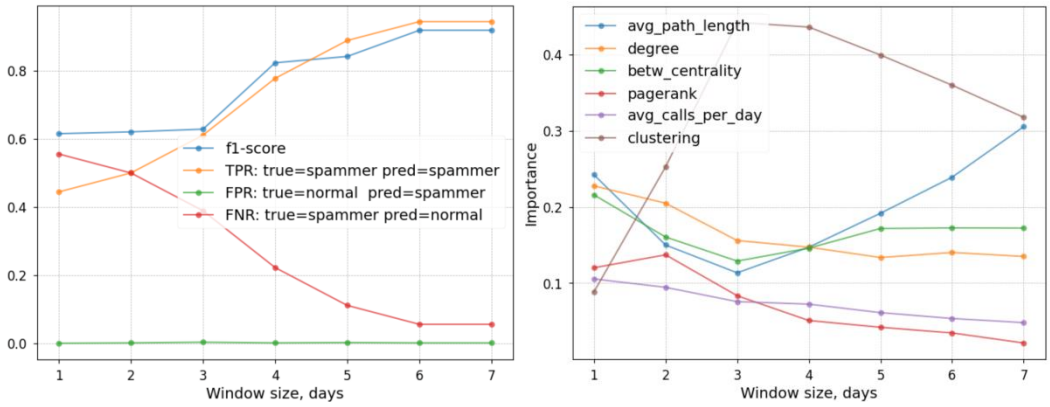
Fig.3 Dynamics of the global clustering coefficient with spammers (5-11 days) and without.

The dependence shown in Fig. 3 illustrates the high sensitivity of the clustering coefficient, with changes of more than an order of magnitude observed, to the presence of spammers in the dynamic

telephone subscriber network. Thus, this coefficient can serve as a clear indicator of abnormal activity within the network. In studies such as [9] and [28], PageRank and the local clustering coefficient have been employed to identify malicious users in systems like Skype and banking networks. These metrics are calculated individually for each user and utilized in classifier models to detect such users. In contrast, the global clustering coefficient can be used to assess the presence of spammer activity across the entire network, before applying classifiers to individual subscribers.

4 Subscriber classification and network properties measurement window size

To classify subscribers and assess the effect of the subscriber properties measurement window size, a numerical experiment was conducted with 10,000 subscribers, 1% of whom were spammers. Random Forest was selected as the classifier due to its minimal requirements for input data distributions, its tolerance for unnormalized or unprocessed data, and its ability to easily generate feature importance values. 80% of the subscribers were included in the training dataset, while the remaining 20% were used for testing. Given that the network of telephone subscribers is dynamic and evolves over time based on subscriber activity, the properties of subscribers also change accordingly. To evaluate the impact of the measurement period on network properties, we constructed seven classifiers, each trained with subscriber properties measured over varying time windows, starting from 1 day and gradually increasing the window size to 7 days. The results of subscriber classification and the importance of subscriber properties for spammer identification are presented in Fig. 4.



a) *b)*

Fig. 4 a) classifier accuracy indicators; b) importance of subscriber properties;

Fig. 4.a. presents the accuracy metrics of the classifier. The F1 score is depicted as a blue curve, the true positive rate (TPR) is shown as an orange curve, the false positive rate (FPR) is represented by a green curve, and the false negative rate (FNR) is indicated by a red curve. Fig. 4.b. illustrates the properties of subscribers and their relative importance within the classification model. The shortest path length (avg_path_length) is shown as a blue curve, the node degree (degree) is represented by an orange curve, node betweenness centrality (betw_centrality) is depicted as a green curve, PageRank is shown as a red curve, the average number of calls per day (avg_calls_per_day) is represented by a purple curve, and the local clustering coefficient (clustering) is shown as a brown curve.

The results of the classifier evaluation demonstrate that the accuracy of spammer detection reaches acceptable levels, with an F1 score greater than 0.80 when the subscriber properties calculation window is set to 4 days. Notably, the number of calls per day is a key distinguishing factor between spammers and normal subscribers; however, it is the least important feature for classification, with its significance diminishing as the window size increases. This can be explained by the fact that some spammers make the same number of calls as normal subscribers, and conversely, some normal subscribers may call excessively. In such cases, when classification based on call frequency is insufficient, network properties provide additional discriminatory power. As shown in Fig. 4b, the three most important properties for classification across all window sizes (1 to 7 days) are node degree, shortest path length, and betweenness centrality. However, for window sizes up to 3 days, these properties, along with the clustering coefficient, are not sufficient for effective spammer detection, as reflected in the lower F1 score and TPR. The F1 score and TPR reach values of 0.81 and 0.79, respectively, only when the window size is 4 days, with the importance of the average shortest path length increasing. As the

window size continues to grow, classifier performance improves, while the relative importance of the local clustering coefficient decreases and the average shortest path length becomes more significant.

5 Conclusion

The identification of telephone subscribers engaged in malicious activities, such as spamming or disturbing normal users, is an important issue in network security. This study explores the potential for detecting the presence of spammers within a telephone subscriber network and classifying individual subscribers. A distinguishing feature of this research is the use of network characteristics as subscriber properties. The study employs a telephone network model to investigate these characteristics. It was found that, to detect the presence of spammers without classifying individual subscribers, the global clustering coefficient is a sufficient metric. This indicator demonstrated high sensitivity with its value decreasing significantly from 0.29 to 0.02 following the emergence of spammers in the network. Furthermore, the study examines the use of subscriber network properties for spammer identification using the Random Forest machine learning model. The importance of network properties was evaluated over seven measurement intervals, ranging from 1 to 7 days. Key properties identified as most important for classification included the local clustering coefficient, average shortest path length, node degree, and betweenness centrality. The study found that, with a measurement window of 4 days, the classifier achieved F1 score of 80%, along with a high spammer detection rate (TPR). These findings suggest that network characteristics significantly enhance model accuracy, although a certain period of observation is required to effectively distinguish between spammers and normal subscribers. Consequently, network characteristics are valuable for identifying dormant or evasive subscribers – those who frequently change numbers, avoid detection through call frequency or blocklists, or remain inactive for extended periods.

REFERENCES

1. N. Davey, S. Field, R. Frank, P. Barson, and G. McAskey, “The detection of fraud in mobile phone networks”, Neural Network World, vol. 6, no. 4, pp. 477–484, 1996.
2. H. Li et al., “A Machine Learning Approach To Prevent Malicious Calls Over Telephony Networks”, in 2018 IEEE Symposium on Security and Privacy (SP), May 2018, pp. 53–69. DOI: 10.1109/SP.2018.00034. (Last accessed: 15.11.2024).
3. “FCC fines illegal robocalling company a record-breaking $300 MILLION after it made more than five billion calls to more than 500M phone numbers in a three-month span | Daily Mail Online.” Accessed: Aug. 16, 2024. URL: https://www.dailymail.co.uk/news/article-12371697/FCC-fines-illegal-robocalling-company-record-breaking-300-MILLION-five-billion-calls-500M-phone-numbers-three-month-span.html (Last accessed: 15.11.2024).
4. N. Jiang et al., “Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis”, in Proceedings of the 10th international conference on Mobile systems, applications, and services, Low Wood Bay Lake District UK: ACM, Jun. 2012, pp. 253–266. DOI: 10.1145/2307636.2307660. (Last accessed: 15.11.2024).
5. H. Tu, A. Doupe, Z. Zhao, and G.-J. Ahn, “SoK: Everyone Hates Robocalls: A Survey of Techniques Against Telephone Spam”, in 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA: IEEE, May 2016, pp. 320–338. DOI: 10.1109/SP.2016.27. (Last accessed: 15.11.2024).
6. P. Patankar, G. Nam, G. Kesidis, and C. R. Das, “Exploring Anti-Spam Models in Large Scale VoIP Systems”, in 28th IEEE International Conference on Distributed Computing Systems (ICDCS 2008), 17-20 June 2008, Beijing, China, IEEE Computer Society, 2008, pp. 85–92. DOI: 10.1109/ICDCS.2008.71. (Last accessed: 15.11.2024).
7. F. Wang, Y. Mo, and B. Huang, “P2P-AVS: P2P Based Cooperative VoIP Spam Filtering”, in Proceedings of the 2007 IEEE Wireless Communications and Networking Conference, USA: IEEE Computer Society, 2007, pp. 3547–3552. DOI: 10.1109/WCNC.2007.650. (Last accessed: 15.11.2024).
8. N. Jiang, Y. Jin, A. Skudlark, and Z.-L. Zhang, “Greystar: fast and accurate detection of SMS spam numbers in large cellular networks using grey phone space”, in Proceedings of the 22nd USENIX Conference on Security, in SEC’13. USA: USENIX Association, 2013, pp. 1–16. DOI: 10.5555/2534766.2534768. (Last accessed: 15.11.2024).

9. A. Leontjeva, M. Goldszmidt, Y. Xie, F. Yu, and M. Abadi, “Early security classification of skype users via machine learning”, in Proceedings of the 2013 ACM workshop on Artificial intelligence and security, Berlin Germany: ACM, Nov. 2013, pp. 35–44. DOI: 10.1145/2517312.2517322. (Last accessed: 15.11.2024).
10. Y. Rebahi, D. Sisalem, and T. Magedanz, “SIP Spam Detection”, in International Conference on Digital Telecommunications (ICDT’06), Cote d’Azur, France: IEEE, 2006, pp. 68–68. DOI: 10.1109/ICDT.2006.69. (Last accessed: 15.11.2024).
11. N. Miramirkhani, O. Starov, and N. Nikiforakis, “Dial One for Scam: A Large-Scale Analysis of Technical Support Scams”, in Proceedings 2017 Network and Distributed System Security Symposium, San Diego, CA: Internet Society, 2017. DOI: 10.14722/ndss.2017.23163. (Last accessed: 15.11.2024).
12. S. Subudhi and S. Panigrahi, “Use of Possibilistic Fuzzy C-means Clustering for Telecom Fraud Detection”, in Computational Intelligence in Data Mining, vol. 556, H. S. Behera and D. P. Mohapatra, Eds., in Advances in Intelligent Systems and Computing, vol. 556. , Singapore: Springer Singapore, 2017, pp. 633–641. DOI: 10.1007/978-981-10-3874-7_60. (Last accessed: 15.11.2024).
13. S. Subudhi and S. Panigrahi, “Quarter-Sphere Support Vector Machine for Fraud Detection in Mobile Telecommunication Networks”, Procedia Computer Science, vol. 48, pp. 353–359, 2015, DOI: 10.1016/j.procs.2015.04.193. (Last accessed: 15.11.2024).
14. R. Zhang and A. Gurtov, “Collaborative Reputation-based Voice Spam Filtering”, in 2009 20th International Workshop on Database and Expert Systems Application, Linz, Austria: IEEE, 2009, pp. 33–37. DOI: 10.1109/DEXA.2009.95. (Last accessed: 15.11.2024).
15. D. Ucci, R. Perdisci, J. Lee, and M. Ahamad, “Building a Collaborative Phone Blacklisting System with Local Differential Privacy”, Jun. 16, 2020, arXiv: arXiv:2006.09287. URL: http://arxiv.org/abs/2006.09287 (Last accessed: 15.11.2024).
16. J. Daka and M. Nyirenda, “Smart Mobile Telecommunication Network Fraud Detection System Using Call Traffic Pattern Analysis and Artificial Neural Network”, vol. 12, pp. 43–50, Apr. 2023, DOI: 10.5923/j.ajis.20221202.01. (Last accessed: 15.11.2024).
17. Q. Zhao, K. Chen, T. Li, Y. Yang, and X. Wang, “Detecting telecommunication fraud by understanding the contents of a call”, Cybersecur, vol. 1, no. 1, p. 8, Dec. 2018, DOI: 10.1186/s42400-018-0008-5. (Last accessed: 15.11.2024).
18. Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria, B. O. Akinyemi, O. H. Odukoya, M. L. Sanni, G. Sewagnon, and G. A. Aderounmu, “Performance Evaluation of Machine Learning based Robocalls Detection Models in Telephony Networks” IJCNIS, vol. 14, no. 6, pp. 37–53, Dec. 2022, DOI: 10.5815/ijcnis.2022.06.04. (Last accessed: 15.11.2024).
19. M. Fire, G. Katz, and Y. Elovici, “Strangers intrusion detection-detecting spammers and fake profiles in social networks based on topology anomalies”, Human journal, vol. 1, no. 1, pp. 26–39, 2012.
20. C. Chaparro and W. Eberle, “Detecting Anomalies in Mobile Telecommunication Networks Using a Graph Based Approach”. URL: https://cdn.aaai.org/ocs/10377/10377-46053-1-PB.pdf. (Last accessed: 15.11.2024).
21. X. Hu, H. Chen, H. Chen, X. Li, J. Zhang, and S. Liu, “Mining Mobile Network Fraudsters with Augmented Graph Neural Networks”, Entropy, vol. 25, no. 1, p. 150, Jan. 2023, DOI: 10.3390/e25010150. (Last accessed: 15.11.2024).
22. S. Ji, J. Li, Q. Yuan, and J. Lu, “Multi-Range Gated Graph Neural Network for Telecommunication Fraud Detection”, in 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–6. DOI: 10.1109/IJCNN48605.2020.9207589. (Last accessed: 15.11.2024).
23. V. Danilevskiy and V. Yanovsky, “Statistical properties of telephone communication network”, arXiv preprint arXiv:2004.03172, 2020. (Last accessed: 15.11.2024).
24. S. Brin and L. Page, “The anatomy of a large-scale hypertextual Web search engine”, Computer Networks and ISDN Systems, vol. 30, no. 1, pp. 107–117, 1998, DOI: https://doi.org/10.1016/S0169-7552(98)00110-X. (Last accessed: 15.11.2024).
25. M. Newman, Networks, vol. 1. Oxford University Press, 2018. DOI: 10.1093/oso/9780198805090.001.0001. (Last accessed: 15.11.2024).

26. D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks”, Nature, vol. 393, no. 6684, pp. 440–442, Jun. 1998, DOI: 10.1038/30918. (Last accessed: 15.11.2024).
27. M. Danilevskyi, V. Yanovsky, and O. Matsiy, “Modeling and Analysis of a Dynamic Network of Telephone Subscribers Considering the Degree of Connectivity by Means of Contact Lists (Unpublished article)”, Bulletin of V.N. Karazin Kharkiv National University, series «Mathematical modeling. Information technology. Automated control systems».
28. C. Azarm, E. Acar, and M. van Zeelt, “On the Potential of Network-Based Features for Fraud Detection”, Feb. 19, 2024, arXiv: arXiv:2402.09495. URL: http://arxiv.org/abs/2402.09495 (Last accessed: 15.11.2024).

Данілевський Михайло Вікторович *аспірант; Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків-22, Україна, 61022*
e-mail: m.danilevskiy@gmail.com
https://orcid.org/0009-0000-0030-2218

Яновський Володимир Володимирович *доктор фізико-математичних наук, професор, професор кафедри штучного інтелекту та програмного забезпечення Харківський національний університет імені В. Н. Каразіна, майдан Свободи 4, Харків-22, Україна, 61022 Завідувач теоретичним відділом, інститут монокристалів НАН України, пр.Науки 60, Харків, Україна, 61001*
e-mail: yanovsky@isc.kharkov.ua
https://orcid.org/0000-0003-0461-749X

Виявлення телефонних абонентів із аномальною поведінкою за допомогою аналізу властивостей мережі

Актуальність. Використання мережі телефонних абонентів з метою шахрайства, продажу товарів та послуг, спаму призводить до щорічних фінансових втрат у мільярди доларів у всьому світі. Проблема вивчалася ще 1996р. і до сьогодні є актуальною, незважаючи на безліч способів протидії та захисту. Традиційні методи, такі як списки блокування та моніторинг частоти викликів, часто неефективні проти спамерів, які обходять ці системи, змінюючи моделі поведінки.

Мета. Метою роботи є вивчення використання мережевих властивостей абонентів, таких як коефіцієнти кластеризації, центральність та середня довжина найкоротшого шляху, як критерії для виявлення абонентів з аномальною поведінкою в динамічній телефонній мережі.

Методи дослідження. Моделювання та чисельний експеримент.

Результати. Дослідження показує, що глобальний коефіцієнт кластеризації є чутливою мірою виявлення присутності спамерів в мережі. Його значення знижується в кілька разів при появі спамерів. При класифікації абонентів на звичайний та спамер за допомогою моделі Random Forest, найважливішими властивостями є локальний коефіцієнт кластеризації, середня довжина найкоротшого шляху, ступінь та центральність абонента в мережі. За даними моделювання мережі телефонних абонентів було виявлено, що при розмірі вікна вимірювання у 4 дні показник точності класифікатора (F1 score) та точності виявлення спамерів (TPR) досягає значень 80%.

Висновки. Використання мережевих характеристик абонентів позитивно впливає на точність виявлення абонентів з аномальною поведінкою, але при цьому вимагає часу, щоб спамери та звичайні абоненти стали помітними за мережевими характеристиками.
Ключові слова: *надокучливі телефонні дзвінки, граф викликів, телефонний спам, телефонна мережа, логнормальний розподіл, розподіл ступенів, коефіцієнт кластеризації, середня довжина найкоротшого шляху*.

Надійшла у першій редакції 02.09.2024, в останній - 09.10.2024.
The first version has been received on 02.09.2024, the final version - on 09.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-05

УДК (UDC) 004.8

Zachepylo Mykhailo *PhD Student, Department of Information Systems named after V. A. Kravets, National Technical University «Kharkiv Polytechnic Institute», 2, Kyrpychova str., 61002, Kharkiv, Ukraine*
e-mail: Mykhailo.Zachepylo@cit.khpi.edu.ua
https://orcid.org/0000-0001-6410-5934

Yushchenko Oleksandr *PhD of Physical and Mathematical Sciences, Professor, professor at the Department of Information Systems named after V. A. Kravets, National Technical University «Kharkiv Polytechnic Institute», 2, Kyrpychova str., 61002, Kharkiv, Ukraine*
e-mail:agyu@kpi.kharkov.ua;
https://orcid.org/0000-0002-0078-3450

Research on survival strategies of artificial life in dynamic environment

This research seeks to develop evolutionary methods for constructing deep neural networks, offering potential improvements to machine learning techniques by modeling adaptive architectures under selective pressures.
Purpose. The goal of the work is to explore the dynamics of neural complexity in artificial life agents exposed to progressively challenging environments.
Research methods. We conducted a two-dimensional simulation to model populations of agents with evolving neural networks and physical forms. The environment progresses from simple conditions to increasingly complex scenarios, including static walls, moving obstacles, hazardous zones, and lethal poisons. Our approach builds on fundamental artificial life systems such as Tierra, Avida, and PolyWorld. The neural architectures evolve based on principles inspired by the NeuroEvolution of Augmenting Topologies. We apply the Tononi–Sporns–Edelman complexity measure to evaluate neural integration and specialization, helping us understand how agents adapt their networks to achieve a balance between global coherence and localized functionality.
Results. Research indicated that while complex environments can temporarily enhance neural sophistication, harsher conditions often favor simpler, more prolific reproductive r-strategies. Effect, populations may create reflex-driven, stimulus-response behaviors instead of developing complex neural structures.
Conclusions. These findings enhance our understanding of adaptive intelligence and guide approaches for designing scalable, matching learning systems in robotics and deep neural network architecture development, contributing to the broader goal of understanding how artificial intelligence should evolve. We propose utilizing a recursive genetic algorithm to optimize these balance challenges, promoting long-term neural adaptation to dynamic environments.

Keywords: *artificial life, neural complexity, evolutionary adaptation, dynamic environments, incremental complexity, open-ended evolution, neuroevolution of augmenting topologies, r-strategy*

Як цитувати: Zachepylo M. O., Yushchenko O. H. Research on survival strategies of artificial life in dynamic environment. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.40-53. https://doi.org/10.26565/2304-6201-2024-64-05

How to quote: Zachepylo M. O., Yushchenko O. H., “Research on survival strategies of artificial life in dynamic environment.”, *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 40-53, 2024. https://doi.org/10.26565/2304-6201-2024-64-05

1. Introduction

Significant progress in deep neural network machine learning has led to revolutionary advancements in technologies such as image recognition, natural language processing, and autonomous systems [1–3]. However, challenges like overfitting, poor generalization, LLM hallucinations, and high computational demands persist, motivating the search for alternative methods of building intelligent multilayered neural networks. The human brain, along with those of higher cetaceans, represents some of the most complex intelligent systems evolved by biological processes, serving as inspiration for simulating the progressive development of artificial neural networks in virtual biocenoses.

The evolution of artificial life (ALife) models has played a crucial role in understanding adaptive behaviors and the emergence of intelligence (noogenesis). Over the decades, researchers have explored diverse systems simulating ecological and evolutionary dynamics. However, significant challenges persist in modeling open-ended evolution, particularly in environments with increasing complexity.

The evolution of Alife models has been instrumental in advancing our understanding of adaptive behaviors. However, previous studies have largely focused on adaptation and have not directly addressed noogenesis within computational ecologies. Over the decades, pioneering systems like Tierra [4] and Avida [5] have offered glimpses into open-ended evolutionary processes. Recent frameworks have built upon foundational systems by exploring morphology, neural complexity, and multi-agent systems, advancing our understanding of these aspects. However, despite these advancements, the challenge of fostering truly open-ended complexity remains unresolved, particularly in dynamic, non-episodic environments that progressively increase in difficulty over time.

This paper addresses a critical gap in ALife research: how does progressive environmental complexification influence the trajectory of morphological, behavioral, and neural complexity in evolving agents? Understanding these dynamics not only advances theoretical knowledge but also has broader implications for designing adaptive artificial systems and improving our grasp of how intelligence might arise in both natural and synthetic contexts. Our study differs from traditional RL setups that reset tasks each episode and from static ALife environments that fail to provide long-term evolutionary pressure. By exploring non-episodic, incrementally complex ecosystems, we aim to illuminate the factors that promote or hinder the emergence of neural complexity and adaptive intelligence. While some studies set the ambitious goal of achieving human-level intelligence [6], we do not aim to produce such outcomes, recognizing this as an unreachable target for now given the current state of ALife research. Instead, we seek to create conditions conducive to the gradual noogenesis of agent populations, offering insights into why complexity sometimes stalls and what might be done to encourage it.

2. Literature review and problem statement

Foundational systems like Tierra [4] and Avida [5] revolutionized our understanding of digital evolution, yet their static environments limited complexity growth. Subsequent studies, such as Karl Sims' evolving virtual creatures [7] and Framsticks [8] extended the exploration of morphology and control. These works primarily focused on objectives like locomotion and simple foraging, without addressing the complexities of open-ended ecosystem dynamics. PolyWorld [9] introduced evolving neural networks to agents in a competitive environment, revealing adaptive behaviors but not exploring progressive complexity. EcoSim [10], JaxLife [6], and multicellularity models [11] further diversified the multi-agent interaction within However, these models often operated in environments that lacked dynamic complexity, limiting insights into long-term evolutionary processes. Hamon et al. [12] incorporated spatiotemporal dynamics in large multi-agent systems, advancing our understanding of distributed interactions. POET [13] investigated paired adaptation in reinforcement learning, illustrating flexibility in task evolution. However, both approaches depend heavily on episodic resets or predefined objectives, restricting their applicability for studying open-ended noogenesis. Efforts by Auerbach and Bongard [14] on how environmental factors can shape morphological complexity, while Giannakakis et al. [15] extended this by highlighting the role of environmental variability in shaping plasticity mechanisms, emphasizing adaptation over time. Notably, this study was not an ALife simulation but rather focused on genetic algorithm optimization in embodied agents, which is important to differentiate. Similarly, Canino-Koning et al. [16] demonstrated that changing environments promote rapid adaptation in Avida, but these instructions-based grid systems differ from more ecology-driven models.

The neural complexity measurements in artificial life simulations were proposed by Yaeger and Sprons [17] and studied over PolyWorld [18], [19]. However, these measurements were conducted in a competitive yet static environment, limiting insights into dynamics under progressively complex conditions.

Despite these achievements, a fundamental question remains unresolved: How does ongoing, continuous environmental complexification, beyond simple predator-prey dynamics, impact the long-term evolutionary dynamics of neural complexity and adaptive behavior? Does increasing environmental complexity consistently drive the evolution of more sophisticated neural architectures, or do resource-efficient strategies, such as simpler yet prolific adaptations, dominate under harsher conditions?

This paper directly addresses this gap. We investigate the interplay between increasing environmental complexity and evolving neural architectures, raising important questions about whether neural network complexity always increases or if other strategies take precedence under harsher conditions.

3. Methods

This section provides an overview of our computational model, evolutionary procedures, and metrics while highlighting additions to previously published work [20]. Our goal remains the simulation of open-ended evolution of artificial creatures in a two-dimensional, progressively complex environment, focusing on neural complexity and adaptive strategies.

3.1. Overview of the Evolution Model

Our model builds on the framework described in [20], which simulates a population of autonomous, embodied agents (artificial "creatures") in a two-dimensional arena. Each creature's characteristics include a connected Physical Body with Neural Network Control. The physical body is represented as a circular entity occupying continuous space, with physics-like rules for movement and interactions. Neural Network Control is governed by an evolving neural network inspired by the NeuroEvolution of Augmenting Topologies (NEAT) [21], with topology and weights subject to mutations. Reproduction is asexual cloning with mutations. In this study, we introduce key updates to the model. Dynamic Environmental Features: The environment evolves over time, incorporating new hazards, spatial obstacles, and resource variability to provide incremental complexity. Additional Sensors: Beyond the basic food sensor, wall sensor, bot sensor, and energy sensor described in [17], creatures now include hazard zone sensors and poison sensors to better navigate increasingly complex environments. Refined Mutation Mechanisms: Mutation probabilities have been recalibrated to include newly added sensors.

3.2. Neural Network Architecture and Complexity

Each creature's control center is a neural network that processes sensory inputs and produces motor outputs. Initially, creatures start with a minimal configuration, typically one food sensor, one energy sensor, one neural node, and a simple pair of effectors (movement and rotation). Over evolutionary time, mutations can add hidden nodes, extra sensors, and more elaborate connections, potentially increasing behavioral sophistication.

We employ the Tononi–Sporns–Endelman (TSE) [22] complexity measure as described and adapted by Yaeger and colleagues [17], [19]. TSE complexity is designed to quantify the balance between integration and segregation in a neural system. Integration refers to the degree of global information sharing among all parts of the system, while segregation reflects how certain subsets of the system maintain specialized, relatively independent patterns of activity. The TSE complexity measure captures how a system's overall integration changes when considering subsets of different sizes.

Consider a neural network composed of n nodes, represented as variables $\{X_i ... X_n\}$. The TSE complexity $C_N(X)$ is defined by comparing the full-system integration against average integrations computed over all subsets of different sizes. Formally (3.1) [22]:

$$C_N(X) = \sum_{k=1}^{n} \left[(k/n) I(X) - \langle I(X_k) \rangle \right], \tag{3.1}$$

where: X – the full set of variables (nodes);

$I(X)$ – the integration of the full system;

$I(X_k)$ – the integration computed over a subset X_k of size k;

$\langle I(X_k) \rangle$ – denotes averaging over all combinations of subsets of size k.

Integration $I(X)$ can be viewed as a multivariate mutual information measure indicating how much of the system's entropy cannot be explained by treating variables as independent (3.2) [22]:

$$I(X) = \sum_{j=1}^{n} H(X_j) - H(X), \tag{3.2}$$

where $H(X_j)$ – the Shannon entropy [23] of a single node X_j

$H(X)$ – a joint entropy of the entire system.

By considering subsets of various sizes k, the measure $C_N(X)$ captures how integration scales and is reduced when parts of the system are examined in isolation. If the system behaves uniformly at all scales, integration values scale linearly, and complexity is low.

Direct computation of all subset combinations is often computationally prohibitive, particularly for large networks. To address this, we employ approximation techniques described by Yaeger [19] and utilize the hardware-accelerated implementation [24]. This implementation leverages JAX, a high-performance library for numerical computation, to significantly speed up the estimation of integration and complexity metrics. The corresponding code is publicly available on https://github.com/WorldThirteen/neural_complexity_jax.

Under the assumption of Gaussian processes, the joint entropy of X can be computed via the covariance matrix **COV** (3.3) [19]:

$$H(X) = \frac{1}{2}\ln(2\pi e)^n |\mathbf{COV}|, \tag{3.3}$$

where $|\mathbf{COV}|$ – the determinant of the covariance matrix of the n nodes' activation patterns.

For our simulations, node activities are recorded during the last 5,000 timesteps of a creature's life. We derive complexity values from these data. By averaging complexity metrics over the population, we gain insights into how environmental pressures influence the evolution of integrated yet specialized neural structures.

This accelerated implementation of TSE complexity enables efficient computation even for large-scale simulations, providing a rigorous framework to examine how creatures' neural architectures balance global coherence with local specialization in response to escalating environmental challenges.

3.3. Environmental Setup and Complexity Increments

The simulation world is a 2D continuous plane with finite dimensions (50,000 x 50,000 units). Initially, this environment is relatively benign, containing only food resources placed at random. Over time, we incrementally introduce complexity to assess how creatures adapt to different survival pressures:

Initial Environment: 500 creatures with minimal neural architecture start in the environment. Food resources are distributed randomly. No hazards, walls, or moving obstacles are present.

Introduction of static walls: we introduce 50 static wall obstacles at random positions. Walls block movement, forcing creatures to navigate around them. The presence of walls tests spatial navigation and obstacle avoidance strategies.

Dynamic obstacles: we add 100 dynamic obstacles that move at a constant velocity (1.5 units/timestep) and bounce off walls. These moving entities require creatures to predict obstacle trajectories and adjust their paths accordingly.

Hazard zones: 100 hazard zones, each a circular area of radius 700 to 2,500 units, are introduced. Entering a hazard zone drains 0.5 energy per simulation tick from a creature, imposing a continuous penalty. Hazard zones test a creature's ability to detect and avoid dangerous regions.

Poison elements: 1,000 poison entities, each with a radius of 90 to 120 units, move at a velocity of 2 units/timestep. Contact with poison is lethal, instantly removing the creature from the population. Poisons represent a severe threat demanding rapid avoidance behaviors.

Environmental elements are updated periodically. For example, walls may randomly change position once every 10,000 timesteps, and hazard zones regenerate every 25,000 timesteps. Visualization of these complexifications can be visible in Fig. 3.1.

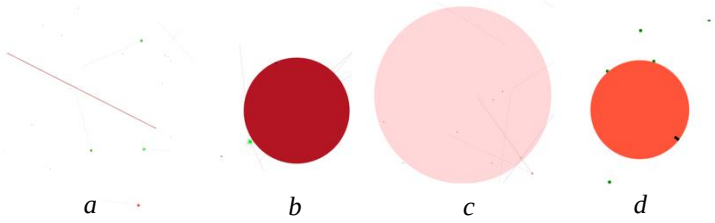


Fig. 3.1. Environmental elements: a – static wall; b – dynamic obstacle; c – hazard zone; d – dynamic poison

The experiment concludes when the population size drops to zero or the simulation reaches a predefined timesteps limit, whichever comes first.

3.4. Simulation Dynamics

The simulation operates in discrete timesteps. During each tick, creatures perceive their environment by reading sensor values and processing these through their neural networks. The resulting neural computations translate sensory inputs into effector outputs, which determine actions such as moving forward, rotating, or remaining stationary. Creatures consume energy continuously and through movement, replenishing it by consuming food (50 energy units per food item). When a creature's energy exceeds 200 units, it reproduces asexually, sharing energy with its offspring. Creatures die if their energy falls below zero or if they collide with poisonous entities, at which point they are removed from the simulation. Meanwhile, the environment evolves dynamically, with obstacles moving, hazard zones regenerating periodically, and poison entities navigating the space.

3.5. Data Collection Protocol

We record data throughout the simulation to analyze evolutionary and behavioral outcomes. Population size – we track the number of living creatures at regular intervals. Mean Lifetime – the average lifespan of creatures is recorded over time. Lifetimes offer insights into survival strategies, reflecting how creatures cope with environmental challenges. Neural Complexity Metrics – using the recorded node activities, we compute the entropy-based complexity measures. These computations are performed offline after the simulation to reduce computational overhead during runtime.

Each metric is saved for post-processing analysis. Visualizations such as population trends, complexity graphs, and network structure diagrams are generated from these recorded data sets.

3.6. Analytical Methods

After the simulations conclude, we use Python-based tools for data analysis. Time-series analysis – we plot population size, mean lifetime, and neural complexity as a function of simulation timesteps. Abrupt changes at known complexity increments (walls, obstacles, hazards, poison) highlight correlations between environmental complexity and evolutionary responses. Complexity computation – the information-theoretic complexity of neural networks is calculated by analyzing node activity logs. Shannon entropy and integration measures are computed for each creature’s neural data. We average values over selected populations and compare complexity trends between simple and complex environmental conditions. Comparative studies – we run multiple experiments under different parameter settings or seeds. Comparisons across runs reveal consistent trends and confirm robustness. Qualitative behavioral analysis – in addition to quantitative metrics, visual observations of creature behavior are qualitatively analyzed. Behavioral patterns such as movement strategies, resource gathering, avoidance of hazards, and interaction with the environment are closely monitored. These qualitative insights help identify emergent behaviors, strategies, and adaptations that complement the numerical results.

4. Results

4.1. Population Dynamics Under Increasing Complexity

Figure 4.1 shows population size over time across the incrementally complex environment as a blue line and simple static environment as an orange line. Initially, when conditions are simpler (0–200,000 timesteps), populations stabilize from the “primordial soup” phase, reflecting abundant resources and minimal threats. Following the introduction of static walls at 200,000 timestep, the populations in complex environments follow the same stabilization trend as populations in simple environments but fluctuate more as creatures must navigate obstacles.

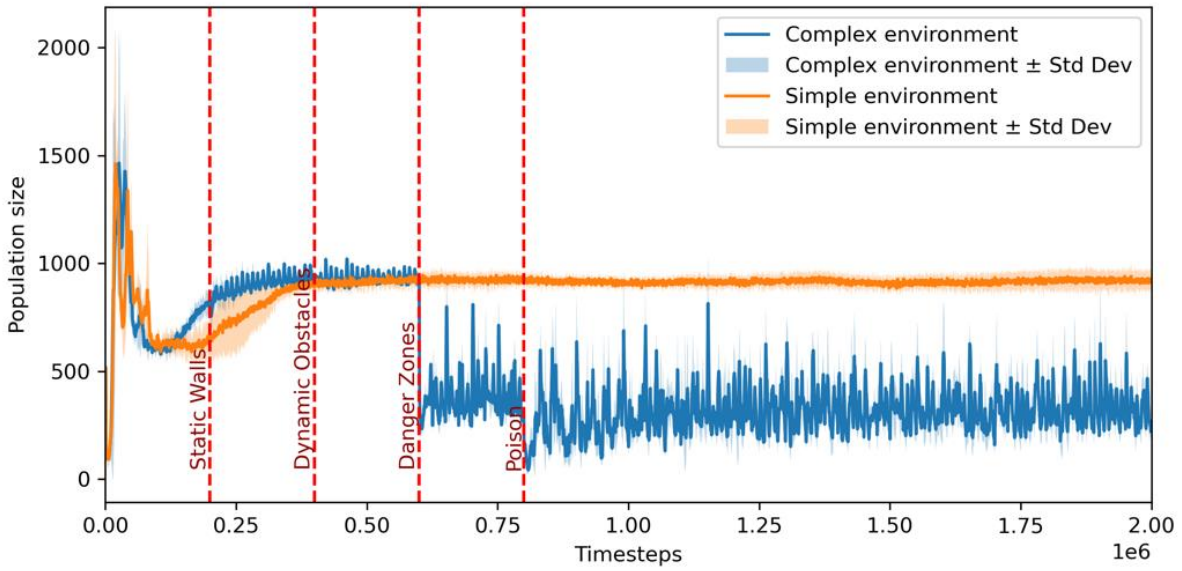


Fig. 4.1. Population size over time

When dynamic obstacles appear at 400,000 timestep, the population size does not indicate a major difference from the previous conditions.

The introduction of hazard zones at 600,000 timestep leads to more significant challenges. As creatures enter these energy-draining regions, their lifespans shorten, reflected by declining population numbers. These zones reward careful navigation and energy management strategies. By this point, the population dynamics show broader oscillations, indicating a struggle between lineages that adapt and those that fail compared to a simple environment that reached a stale state in population size by this time.

Finally, at 800,000 timestep, the environment becomes most lethal with the presence of poison elements. Creatures that encounter poison are instantly removed from the gene pool, drastically increasing selective pressure. The population size (Fig. 2), and the blue line have high fluctuations at a lower threshold or show steep episodic declines as new hazards periodically reshuffle conditions. The result is a population characterized by rapid turnover and intense evolutionary pressure.

4.2. Survival Strategies and Lifespans

To evaluate creature longevity, we tracked mean lifetimes across different environmental complexities (Fig. 4.2, where the blue line indicates an environment with complexities and a simple static environment is depicted as an orange line). In simpler conditions, creatures exhibited extended lifespans, capitalizing on abundant resources and reproducing multiple times. The introduction of static dynamic obstacles at 200,000 and 400,000 timesteps increased lifespans fluctuations.

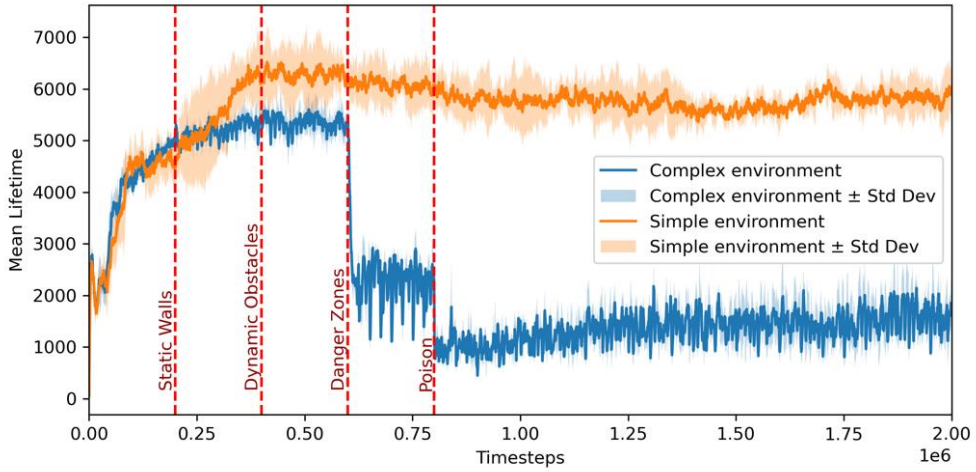
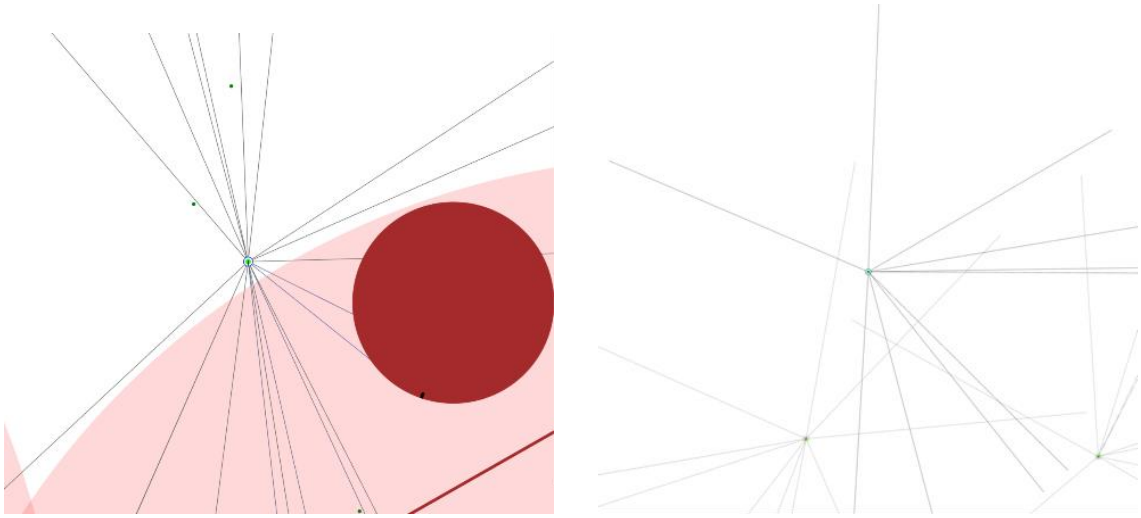


Fig. 4.2. Mean creature lifetime in population over time

Hazard zones introduced at 600,000 timestep reduced mean lifetimes, with only a few lineages evolving avoidance or energy-conservation behaviors. By 800,000 timestep, the addition of poison elements led to the predominance of short-lived, fast-reproducing strategies, minimizing neural and morphological investments. In contrast, in the simpler control environment devoid of these escalating challenges, mean lifetimes remained consistently higher as creatures invested in stable, energy-efficient behaviors, favoring long-term survival and reproduction.

4.3. Morphological and Neural Adaptations

Fig. 4.3, *a* depicts a representative evolved creature morphology under a complex environment. Over many generations, body size and sensor placements can change, reflecting attempts to cope with walls, obstacles, hazard zones, and poison. These lineages often sacrifice stability and complexity for speed and agility, resulting in more streamlined morphologies optimized for quick bursts of foraging and immediate reproduction before encountering lethal conditions.



a *b*

Fig. 4.3. Visual representation of the evolved creature morphology in, circles are bodies, lines are sensors. Where a – in complex environment, b – in simple environment

In contrast, creatures evolved under simpler conditions, illustrated in Fig. 4.3, *b*, often maintain more moderate body sizes and sensor distributions. Without harsh selective pressures, these lineages can afford slower, more energy-efficient foraging strategies. They do not need extreme agility and/or rapid reproduction since they face fewer existential threats. From a neural architecture perspective, Fig. 4.4 compare the neural networks of evolved creatures in complex versus simple environments. Under a complex environment (Fig. 4.4), neural networks may initially expand to incorporate sensors and interneurons for threat detection. However, over time and under extremely lethal conditions, there is a trend toward minimal networks that directly connect critical sensors (e.g., poison or hazard detectors) to effectors for rapid avoidance maneuvers. These networks often remain shallow and specialized, favoring direct stimulus-response paths instead of integrative, deeply hierarchical architectures. This pattern resembles the emergence of direct reflex-like mechanisms, optimized for immediate survival in highly selective environments.

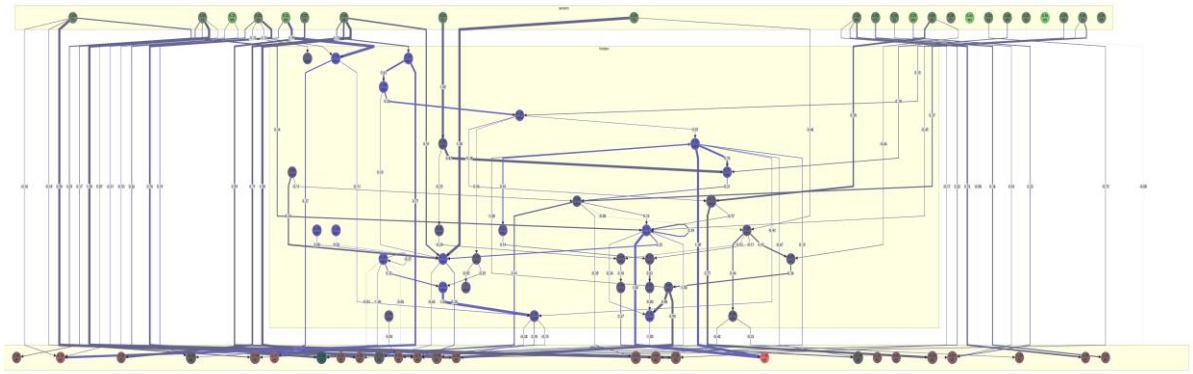


Fig. 4.4. Visual representation of the evolved creature neural network in a complex environment. The top row contains sensor neurons, the bottom row contains effectors, the middle are processing neurons, and the lines are synaptic links.

In simple environments, the neural networks of the creatures often become more elaborate. A sample of such a network is depicted in Fig. 4.5. With less pressure to avoid lethal threats immediately, creatures explore more sophisticated sensor integration, potentially leading to richer behavioral repertoires. These creatures can afford neural complexity that does not yield an immediate survival advantage but may improve long-term foraging efficiency or energy management.

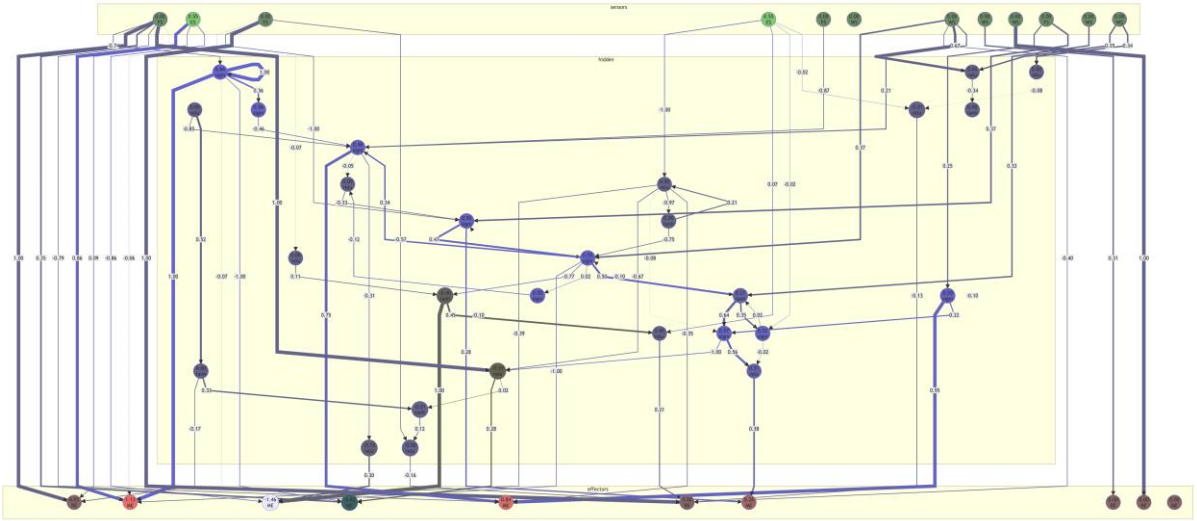


Fig. 4.5. Visual representation of the evolved creature neural network in a simple environment. The top row contains sensor neurons, the bottom row contains effectors, the middle are processing neurons, and the lines are synaptic links.

4.4. Neural Complexity Trends

The mean population neural complexity over time is shown in Fig. 4.6, where orange lines are simulations in a simple environment and blue lines are simulations within environments of grown complexity. Simple environmental complexifications – static walls and dynamic obstacles briefly stimulate neural complexity growth as creatures experiment with more intricate sensorimotor mappings to cope with new obstacles. However, as the environment becomes more hostile (post-600,000 and especially post-800,000 timestep), complexity does not continue to rise. Instead, simpler, more direct strategies dominate, leading to a plateau or even a decline in complexity metrics right after high instability.

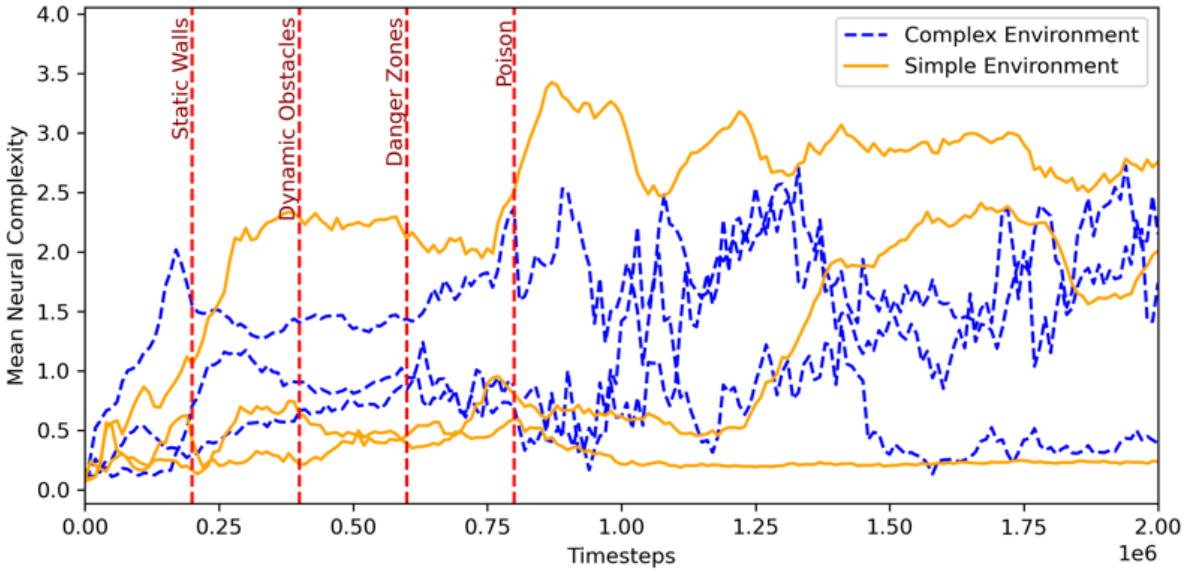


Fig. 4.6. Mean population Neural Complexity over time for simulation runs with complex and simple environments

Figure 4.7, which presents longer simulation runs with matched initial seeds, confirms that complexity is not guaranteed to increase simply by making the environment harder, even if more time is given for the population to adapt. While both complex and simple conditions start similarly, the complex environment lineages eventually diverge, often settling on less complex neural configurations.

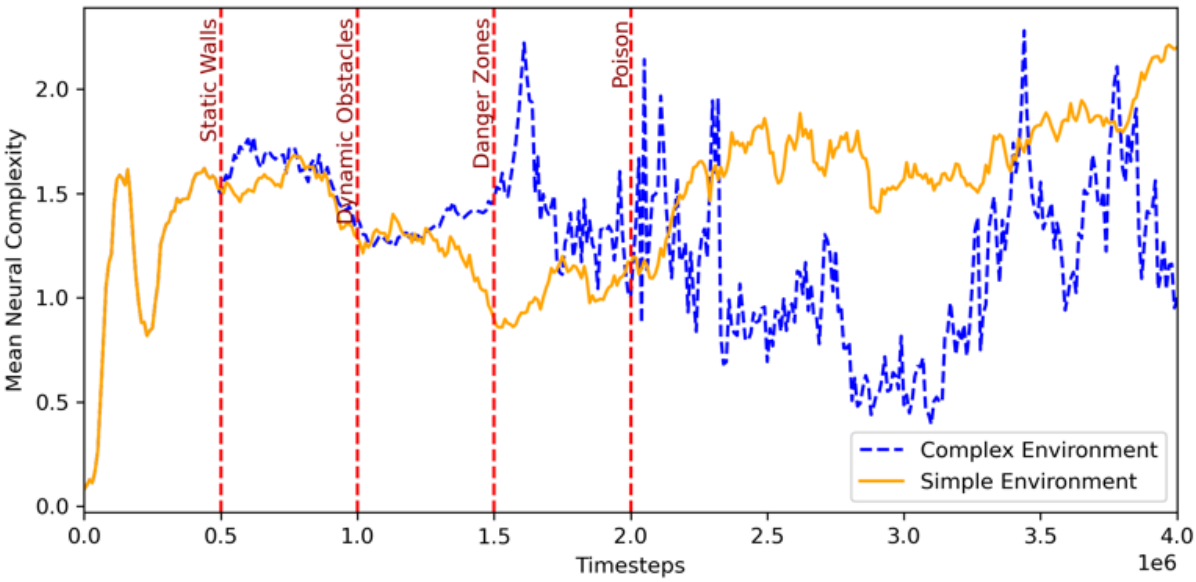


Fig. 4.7. Mean population Neural Complexity over time for simulation runs with complex and simple environments with the same seed

4.5. Generational Turnover and Reproductive Strategies

To understand how lineages maintain themselves under severe selection pressures, we examined the generational patterns of births and deaths over time. Figure 4.8 shows a scatter plot of the generations of newly born creatures per time window. In the complex environment, the high turnover rate is evident. As the environment grows more lethal, lineages adopt rapid reproductive strategies, producing large numbers of offspring in quick succession. This ensures that some descendants survive by sheer volume, even if most die young and fail to evolve more intricate neural control systems.

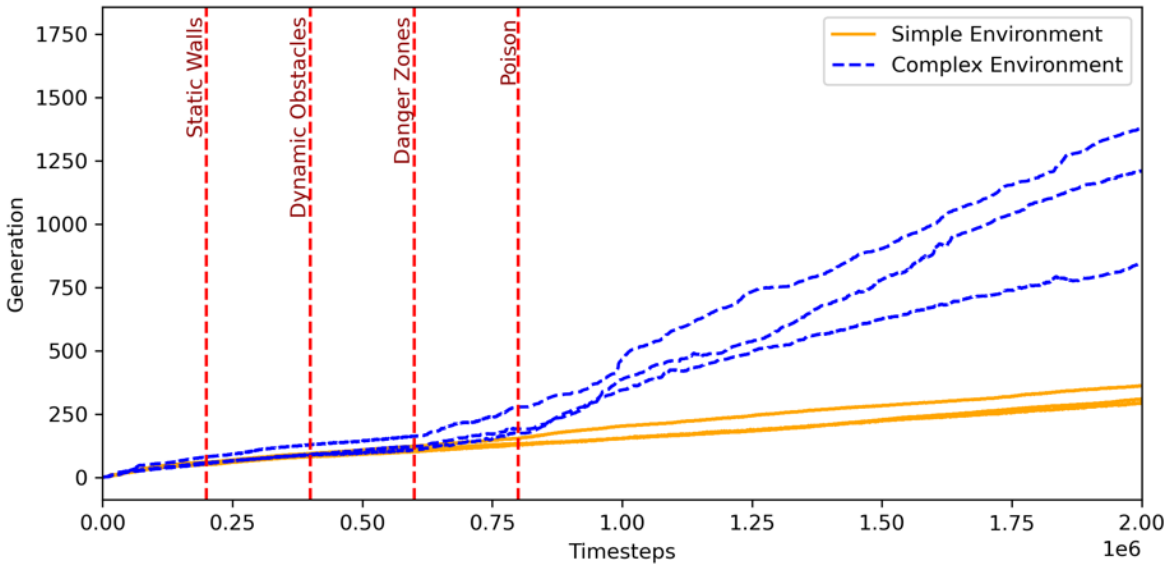


Fig. 4.8. Mean generation of born in its generation per simulation timesteps, complex environment, simple environment

In a simpler environment, generation turnover is more gradual. Creatures reproduce at a moderate pace, often retaining more complex neural networks since their survival is not constantly threatened. This leads to more stable population structures and longer-lived individuals.

4.6. Summary of Findings

Across three repeated experiments under each setting, we found consistent trends. Under escalating complexity (walls at 200,000, dynamic obstacles at 400,000, hazard zones at 600,0000, poisons at 800,000), we observe several trends. Population sizes fluctuate more widely, with pronounced crashes following the introduction of deadly elements (Fig. 4.1). Mean lifetimes decrease significantly, especially once hazard zones and poisons appear (Fig. 4.2). Morphologies shift toward lean, agile forms prioritizing fast reproduction (Fig. 4.3, *a*), whereas simpler environments yield more moderate, stable morphologies (Fig. 4/3, *b*). Neural architectures do not necessarily become more complex; instead, they often remain or even revert to simpler stimulus-response patterns under lethal pressures (Fig. 4.4 vs. Fig. 4.5). Neural complexity measures (Fig. 4.6, Fig. 4.7) show that complexity growth is not guaranteed in more challenging environments. High generational turnover (Fig. 4.8) characterizes populations in harsh conditions, reflecting a strategic pivot away from complexity toward rapid proliferation.

5. Discussion

The results of this study provide a nuanced perspective on the evolutionary trajectories of ALife populations under incrementally increasing environmental complexity. While it is reasonable to hypothesize that adaptation to complex environments would grant individuals with greater cognitive capacity (i.e., higher neural complexity) an adaptive advantage, thereby driving a gradual increase in population-wide neural complexity and fostering noogenesis, the findings suggest that this outcome is not guaranteed.

Initial increases in environmental complexity (e.g., walls, moderate obstacles) occasionally encouraged the emergence of diverse sensorimotor strategies and incremental neural complexity. However, these trends were neither stable nor consistent. As environmental conditions became increasingly lethal, populations often adopted strategies prioritizing survival through quantity rather than quality. Organisms reproduced rapidly, had shorter lifespans, and exhibited simpler stimulus-response neural networks. In essence, rather than evolving toward cognitively complex forms, evolutionary lineages frequently gravitated toward r-strategist solutions characterized by high reproduction rates and rapid generational turnover, as described by r/K selection theory in ecology [25]. This allowed populations to persist despite constant existential threats.

This phenomenon aligns with certain ecological and evolutionary principles observed in nature. Darwin's concept of the "economy of nature" [26] posits that evolution is not a linear progression toward complexity but a process driven by immediate survival and reproductive needs. Many biological systems, particularly in harsh environments, favor strategies that maximize short-term fitness. For instance, r-strategists thrive in unpredictable or lethal conditions by producing numerous offspring quickly, often at the cost of individual longevity or complexity. Analogously, our ALife populations exhibit similar behaviors, demonstrating that increased environmental difficulty does not inherently lead to greater complexity. On the contrary, the presence of critical threats often necessitated reductions in neural complexity, reverting neural architectures to simplified, reflex-driven networks.

Neural complexity metrics derived from the Tononi–Sporns–Edelman (TSE) measure [22] indicate that complexity can rise temporarily but often plateaus or declines under lethal environmental pressures. While Yaeger L. and Griffith V. anticipated that increasing environmental complexity would drive higher neural complexity in agents [18], our findings underscore the necessity of avoiding parasitic r-strategy dynamics. The conditions must provide stable incentives for complexity – selective pressures that consistently reward integrated, adaptive cognitive strategies over short-term exploitative tactics.

These insights suggest that complexity emerges most readily in balanced environments where agents are pressured to adapt and explore richer behavioral niches without facing mortality rates so high that reproductive quantity becomes the most viable strategy.

To promote progressive neural network development through adaptive control of artificial life in dynamic environments, it is advisable to use recursive genetic algorithms [27] with several strategies. Layered Challenges that Reward Complexity – Rather than introducing merely lethal elements, design scenarios where certain high-value resources or safe zones can only be accessed through complex behaviors or problem-solving. Complexity should be not only useful but necessary for stable survival. Social and Cooperative Pressures – introducing tasks that require cooperation, communication, or social learning may incentivize more integrated neural networks. Complexity may flourish if communication or coordination among agents confers a clear survival advantage. Gradualism and Stability – providing stable intermediate niches, where complexity pays off before lethal threats dominate, could help complexity accumulate. Environments that ramp up difficulty more gradually might prevent populations from collapsing into simple r-strategies. Penalizing Simplicity – another approach is to modify fitness landscapes so that purely r-strategic solutions have diminishing returns. For example, increasing energy costs or imposing metabolic trade-offs that only complex networks can efficiently navigate might nudge the evolutionary trajectory toward complexity.

6. Conclusion

Diverse biological organisms have evolved distinct survival strategies, with intelligence being one of them, albeit the most successful. The fact that evolution in terrestrial and aquatic environments has led to the emergence of highly cephalized forms highlights the inherent patterns driving noogenesis. Observations of instances where artificial neural networks exhibit excessive complexity during sustained adaptation further support the arguments against traditional objections to Darwin's theory. These objections often claim that the genetic basis for advanced cognitive abilities, such as mathematical skills, could not have evolved prior to the existence of such cognitive needs.

To replicate simplified noogenesis within neural networks through adaptive control of artificial life, future models must incorporate mechanisms capable of identifying and mitigating parasitic strategies—those that enable populations to thrive without contributing to increased complexity. The use of recursive genetic algorithms offers a promising approach, allowing for the dynamic adjustment of environmental complexity to foster a sustained trend of neural network intellectualization. This iterative, feedback-driven process mirrors the dynamics of natural evolution and holds significant potential for achieving stable noogenesis in artificial life systems.

The practical significance of this study lies in the development of alternative methods for developing and training multilayered neural networks, overcoming existing limitations such as overfitting, poor generalization, high computational demands, and improving approaches to training autonomous robots. These findings have direct applications in various fields, including evolutionary design methods and solving complex practical challenges through adaptive, evolution-inspired learning systems.

REFERENCES

1. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. 2012. vol. 25, pp. 84–90. DOI: 10.1145/3065386.
2. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17).* Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6000–6010. https://dl.acm.org/doi/10.5555/3295222.3295349.
3. Silver D., Huang A., Maddison C. J., Guez A., Sifre L., van den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel T., Hassabis D. Mastering the game of Go with deep neural networks and tree search. *Nature*. 2016, vol. 529, no. 7587, pp. 484–489. DOI: 10.1038/nature16961.
4. Ray T. S. Evolution, Ecology, and Optimization of Digital Organisms. Santa Fe, 1992. 47 p. https://faculty.cc.gatech.edu/~turk/bio_sim/articles/tierra_thomas_ray.pdf
5. Adami C., Brown C. T. Evolutionary Learning in the 2D Artificial Life System "Avida". Artificial Life IV. 2020. pp. 373–377. DOI: 10.7551/MITPRESS/1428.003.0049.
6. Lu C., Beukman M., Matthews M., Foerster J. JaxLife: An Open-Ended Agentic Simulator. *Proceedings of the ALIFE 2024: Proceedings of the 2024 Artificial Life Conference*. Online, July 22–26, 2024. p. 47. DOI: 10.1162/isal_a_00770.
7. Sims K. Evolving virtual creatures. *In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. New York, 1994, pp. 15–22. DOI: 10.1145/192161.192167.
8. Adamatzky A. Framsticks. Kybernetes. 2000, vol. 29, no. 9/10. DOI: 10.1108/k.2000.06729iad.001.
9. Yaeger L. Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or PolyWorld: Life in a New Context. *Santa Fe Institute Studies in the Sciences of Complexity - Proceedings Volume*. 1994, vol. 17, p. 263. https://www.researchgate.net/publication/2448680_Computational_Genetics_Physiology_Metabolism_Neural_Systems_Learning_Vision_and_Behavior_or_PolyWorld_Life_in_a_New_Context
10. Gras R., Devaurs D., Wozniak A., Aspinall A. An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. *Artificial Life*. 2009, vol. 15(4), pp. 423–463. https://pubmed.ncbi.nlm.nih.gov/19463060/
11. Cope D. Real-time Evolution of Multicellularity with Artificial Gene Regulation. *The 2023 Conference on Artificial Life*. MIT Press, May 2023, pp. 77–86. DOI: 10.1162/isal_a_00690.
12. Hamon G., Nisioti E., Moulin-Frier C. Eco-evolutionary Dynamics of Non-episodic Neuroevolution in Large Multi-agent Environments. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 2023, pp. 143–146. DOI: 10.1145/3583133.3590703.
13. Wang R., Lehman J., Clune J., Stanley K. O. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. 2019. Available at: https://arxiv.org/abs/1901.01753 (access date: 29.08.2023)
14. Auerbach J. E., Bongard J. C. Environmental Influence on the Evolution of Morphological Complexity in Machines. *PLOS Computational Biology*. 2014, vol. 10(1). DOI: 10.1371/journal.pcbi.1003399.
15. Giannakakis E., Khajehabdollahi S., Levina A. Environmental variability and network structure determine the optimal plasticity mechanisms in embodied agents. *Proceedings of the ALIFE 2023: Ghost in the Machine*. Online, 2023, p. 22. DOI: 10.1162/isal_a_00606.

16. Canino-Koning R., Wiser M. J., Ofria C. The Evolution of Evolvability: Changing Environments Promote Rapid Adaptation in Digital Organisms. Proceedin*gs of the Artificial Life Conference 2016*. 2016, pp. 268–275. DOI: 10.1162/978-0-262-33936-0-CH047.
17. Yaeger L. S., Sporns O. Evolution of Neural Structure and Complexity in a Computational Ecology. *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*. MIT Press, Cambridge, MA, 2006, pp. 330–336. https://www.researchgate.net/publication/228630335_Evolution_of_neural_structure_and_complexity_in_a_computational_ecology
18. Yaeger L., Griffith V., Sporns O. Passive and Driven Trends in the Evolution of Complexity. Artificial Life XI: Proceedings of the Eleventh *International Conference on the Simulation and Synthesis of Living Systems*. 2008, pp. 725–732. https://arxiv.org/abs/1112.4906
19. Yaeger L. S. Identifying Neural Network Topologies That Foster Dynamical Complexity. *Advances in Complex Systems*. 2013. vol. 16, iss. 02n03, p. 1350032. DOI: 10.1142/S021952591350032X.
20. Zachepylo M., Yushchenko O. The Scientific Basis, Some Results, and Perspectives of Modeling Evolutionarily Conditioned Noogenesis of Artificial Creatures in Virtual Biocenoses. *Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies*. 2023, no. 2 (10), pp. 85–94. DOI: 10.20998/2079-0023.2023.02.13.
21. Stanley K. O., Miikkulainen R. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*. 2002, vol. 10(2), pp. 99–127. DOI: 10.1162/106365602320169811.
22. Tononi G., Sporns O., Edelman G. M. A measure for brain complexity: Relating functional segregation and integration in the nervous system. *Proc Natl Acad Sci USA*. 1994, vol. 91, no. 11, pp. 5033–5037. DOI: 10.1073/PNAS.91.11.5033.
23. Shannon C. E. A Mathematical Theory of Communication. *Bell System Technical Journal*. 1948, vol. 27, no. 3, pp. 379–423. DOI: 10.1002/J.1538-7305.1948.TB01338.X.
24. Zachepylo M., Yushchenko O. Assessing Neural Complexity for Noogenesis in ALife Simulations. *XVIII International Scientific and Practical Conference of Master's and Postgraduate Students Theoretical and Practical Research of Young Scientists*. Kharkiv: National Technical University Kharkiv Polytechnic Institute, 2024, p. 10. https://web.kpi.kharkov.ua/masters/wp-content/uploads/sites/135/2024/11/Zbirnyk-tez-TPRYS2024.pdf
25. MacArthur R. H., Wilson E. O. The Theory of Island Biogeography. Princeton, NJ: Princeton University Press, 1967. 224 p. https://www.semanticscholar.org/paper/The-Theory-of-Island-Biogeography-Macarthur-Wilson/25e2b6dbf36bfe4b269e2dd70f6c3d00fb266484
26. Darwin C. On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life. Evolutionary Writings. 2010. DOI: 10.1093/OWC/9780199580149.003.0005.
27. Yushchenko A. G. Recursive Genetic Algorithm for Solving the Traveling Salesman Problem. Anniversary Edition of "Information Systems" Department at NTU KhPI. 2014, pp. 154–162. Available at: https://www.researchgate.net/publication/262686057 (access date: 03.12.2024)

Зачепило Михайло Олександрович
Аспірант кафедри Системи Інформації ім. В. О. Кравця;
Національний технічний університет «Харківський політехнічний інститут», вул. Кирпичева, 2, Харків, *Україна*, 61024
e-mail: Mykhailo.Zachepylo@cit.khpi.edu.ua
https://orcid.org/0000-0001-6410-5934

Ющенко Олександр Георгійович
кандидат фізико-математичних наук, професор;
професор кафедри Системи Інформації ім. В. О. Кравця, Національний технічний університет «Харківський політехнічний інститут», вул. Кирпичева, 2, Харків, *Україна*, 61024
*e-mail:*agyu@kpi.kharkov.ua;
https://orcid.org/0000-0002-0078-3450

Дослідження стратегій виживання штучного життя у динамічному середовищі

Це дослідження спрямоване на розробку еволюційних методів для побудови глибоких нейронних мереж, пропонуючи потенційні вдосконалення методів машинного навчання шляхом моделювання адаптивних архітектур під впливом селективного тиску.
Мета. Метою роботи є дослідження динаміки нейронної складності у агентів штучного життя, які взаємодіють із дедалі складнішими умовами середовища.
Методи дослідження. Ми провели двовимірну симуляцію для моделювання популяцій агентів з нейронними мережами та фізичними формами, що еволюціонують. Середовище змінюється від простих умов до більш складних сценаріїв, включаючи статичні стіни, рухомі перешкоди, небезпечні зони та смертельні отрути. Наш підхід базується на фундаментальних системах штучного життя, таких як Tierra, Avida та PolyWorld. Нейронні архітектури еволюціонують на основі принципів, натхненних NeuroEvolution of Augmenting Topologies. Ми застосовуємо міру складності Тононі–Спорнса–Едельмана для оцінки нейронної інтеграції та спеціалізації, що допомагає зрозуміти, як агенти адаптують свої мережі для досягнення балансу між глобальною когерентністю та локалізованою функціональністю.
Результати. Дослідження показало, що, хоча складні середовища можуть тимчасово підвищувати нейронну складність, жорсткіші умови часто сприяють простішим, але більш продуктивним репродуктивним r-стратегіям. У результаті популяції можуть формувати рефлекторні, стимул-реакційні поведінкові моделі, замість розвитку складних нейронних структур.
Висновки. Ці результати поглиблюють наше розуміння адаптивного інтелекту і допомагають у розробці підходів до створення масштабованих систем машинного навчання у робототехніці та розробці архітектур глибоких нейронних мереж, що сприяє досягненню ширшої мети розуміння еволюції штучного інтелекту. Ми пропонуємо використовувати рекурсивний генетичний алгоритм для оптимізації цих балансів викликів, що сприяє довгостроковій нейронній адаптації до динамічних середовищ.

Ключові слова: *штучне життя, нейронна складність, еволюційна адаптація, динамічні середовища, поступова складність, необмежена еволюція, нейроеволюція розширюваних топологій, r-стратегія*

Надійшла у першій редакції 03.09.2024, в останній - 10.10.2024.
The first version has been received on 03.09.2024, the final version – on 10.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-06

УДК (UDC) 004.891.2

Kachanov Stanislav *PhD student, Department of Theoretical and Applied Computer Science V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*
e-mail: staskachanov2000@gmail.com
https://orcid.org/0009-0002-6938-6717

Chen Guoxin *Master's student, Department of Theoretical and Applied Computer Science V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*
e-mail: guoxin.chen@student.karazin.ua;
https://orcid.org/0009-0004-0502-3735

Morozova Anastasiia *PhD, Associate Professor, Department of Theoretical and Applied Computer Science V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*
e-mail: a.morozova@karazin.ua;
https://orcid.org/0000-0003-2143-7992

Rukkas Kyrylo *DSc, Associate Professor, Full Professor, Department of Theoretical and Applied Computer Science V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*
e-mail: rukkas@karazin.ua;
https://orcid.org/0000-0002-7614-0793

Prediction of the dynamics COVID19 epidemic process of using the Lasso regression model

This study integrates machine learning and deep learning methods to predict the COVID-19 pandemic.
Relevance. The global outbreak of the COVID-19 pandemic has had a profound impact on public health systems and socio-economic structures worldwide, highlighting the urgent need for effective forecasting tools to aid decision-making. The work is devoted to the development of multi-model framework for epidemic forecasting by integrating advanced methods of mathematical modeling and forecasting theory.
Goal. The purpose of the work was to analyze methods and algorithms for cumulative prediction of COVID-19 cases in order to provide scientific support for public health decision-making.
Research methods. The research methods are based on modern theories of mathematical modeling, artificial intelligence, epidemiological diagnostics, and forecasting theory, namely: Lasso regression, Long Short-Term Memory (LSTM) networks, and LSTM-Attention models. The research details the processes of data preprocessing, model training, evaluation, and visualization to maintain generalization and adaptability in the dynamic pandemic scenario.
The results. The application of Lasso regression model Long Short-Term Memory (LSTM) network for cumulative prediction of COVID-19 cases was investigated to provide scientific support for decision-making. The research details the processes of data preprocessing, model training, evaluation, and visualization to maintain generalization and adaptability in the dynamic pandemic scenario. Additionally, a Multi-Scale LSTM-Attention (MSLA) model was proposed to extract multi-period features from input sequences. These features are critical for addressing data non-stationary.
Conclusions. The task of developing a comprehensive multi-model COVID-19 prediction system by integrating machine learning and deep learning techniques was solved. The system combines Lasso regression, Long Short-Term Memory (LSTM) networks, and a novel Multi-Scale Cumulative Infection Prediction model based on an attention mechanism (MSLA), significantly enhancing prediction accuracy and reliability.

Keywords: *prediction, neural network, Lasso Regression Model, LSTM, COVID-19 prediction, Machine Learning, Deep Learning, Multi-scale model, database.*

Як цитувати: Kachanov S., Chen G., Morozova A., Rukkas K. Prediction of the dynamics COVID19 epidemic process of using the Lasso regression model. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.54-65. https://doi.org/10.26565/2304-6201-2024-64-06

How to quote: S. Kachanov, G. Chen, A. Morozova, K. Rukkas, "Prediction of the dynamics COVID19 epidemic process of using the Lasso regression model", *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 54-65, 2024. https://doi.org/10.26565/2304-6201-2024-64-06

1 Introduction

The global outbreak of the COVID-19 pandemic has had a profound impact on public health systems and socio-economic structures worldwide, highlighting the urgent need for effective forecasting tools to aid decision-making [1]. As powerful data analysis tools, machine learning and deep learning have demonstrated significant potential in epidemic forecasting [2]. This study aims to construct a multi-model ensemble framework for epidemic forecasting by integrating methods such as Lasso regression, Long Short-Term Memory (LSTM) networks, and LSTM-Attention models. The framework is designed to capture complex patterns and temporal dependencies in epidemic data, providing more accurate prediction results to support public health decision-making.

Based on epidemic data from China, this research first preprocesses and engineers a large volume of data to ensure data quality and extract meaningful information. Subsequently, it employs Lasso regression as a machine learning method, alongside LSTM and other deep learning models, to predict epidemic trends. By comparing the performance of different models, this study adopts ensemble techniques to improve prediction accuracy and robustness. The paper details the processes of model training, evaluation, and result visualization and explores optimizing predictive performance by adjusting model parameters and architectures. Through this study, we aim to provide a robust forecasting tool for the public health sector, enabling a scientific and precise response to future public health challenges.

2 Current Research on Prediction Methods

2.1. Traditional Prediction Methods

Traditional forecasting techniques primarily include the moving average method, exponential smoothing, and the Autoregressive Integrated Moving Average (ARIMA) model. The moving average method [3] predicts future infection counts by calculating the arithmetic mean of historical data. While straightforward, it has limitations in capturing dynamic epidemic trends. Exponential smoothing assigns higher weights to more recent data points, making it suitable for handling datasets with trends and seasonal variations. The ARIMA model [4] integrates autoregressive, differencing, and moving average techniques, making it particularly effective for analyzing non-stationary time series data. These methods are generally employed for short-term forecasting and are often used in combination to enhance prediction accuracy.

Despite advancements in technology and the increasing popularity of modern methods such as neural networks and machine learning, traditional methods remain valuable. These approaches continue to provide insights for predicting COVID-19 infection counts, especially in scenarios with limited data or low complexity requirements.

2.2. Machine Learning-Based Prediction Methods

Machine learning-based methods for predicting COVID-19 infection counts have gained increasing attention in recent years due to advancements in data science technologies. Machine learning methods exhibit significant advantages in handling complex and nonlinear data compared to traditional statistical methods. With their strong fitting capabilities [5], these methods can capture intricate patterns in the spread of epidemics. They allow for the integration of various input variables, such as meteorological data, holidays, and population mobility, to improve prediction accuracy. Furthermore, machine learning models can automatically adjust parameters based on new data, adapting to changes in infection trends, and generally provide higher prediction accuracy compared to traditional approaches.

Although machine learning algorithms can handle large datasets and automatically learn patterns, they are typically restricted to panel data, limiting their ability to extract trends, causal relationships, and long-term dependencies from time series data. Model accuracy requires further improvement.

2.3. Deep Learning-Based Prediction Methods

Deep learning methods have shown significant potential in predicting COVID-19 infection and mortality rates [6], particularly in handling large datasets and capturing complex patterns. As epidemic

data often take the form of time series, various time series neural network models have been widely explored.

Long Short-Term Memory (LSTM) networks, a special type of Recurrent Neural Network (RNN), effectively address the long-term dependency challenges in time series data. In COVID-19 infection prediction, LSTMs capture trends and periodic features in time series, achieving accurate forecasts [7–8].

Convolutional Neural Networks (CNNs), primarily used for image recognition, can also extract local features from time series data in infection prediction. Combining CNNs with LSTMs further enhances predictive capabilities [9].

These methods can be summarized as multivariate time series prediction problems with different feature extraction techniques. However, their performance on data with varying periodicities remains suboptimal.

Therefore, the task of prediction of the dynamics covid19 epidemic process of using the Lasso regression model is a relevant problem.

3 Relevant Technical Principles

This paper focuses on the prediction of COVID-19 data, discussing the research background and current state of the field. Different machine learning and deep learning prediction models are utilized, and various ablation experiments are conducted based on real-world production scenarios.

3.1. LSTM Principle

Long Short-Term Memory (LSTM) networks [10] are a special type of Recurrent Neural Network (RNN) designed to address the gradient vanishing or explosion problems encountered by traditional RNNs when processing long sequences of data. LSTMs introduce three key gating mechanisms—the Forget Gate, Input Gate, and Output Gate—whose calculation formulas are given by (3.1), (3.2), (3.3), and (3.4).

$$f_t = \sigma\left(W_f h_{t-1} + U_f x_t + b_f\right) \tag{3.1}$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{3.2}$$

$$\tilde{C}_t = tanh(W_a h_{t-1} + U_a x_t + b_a) \tag{3.3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{3.4}$$

These three gates effectively maintain and update the internal states and outputs of the network, allowing LSTMs to capture long-term dependencies in time series data. The detailed structure of the LSTM is shown in Figure 1.

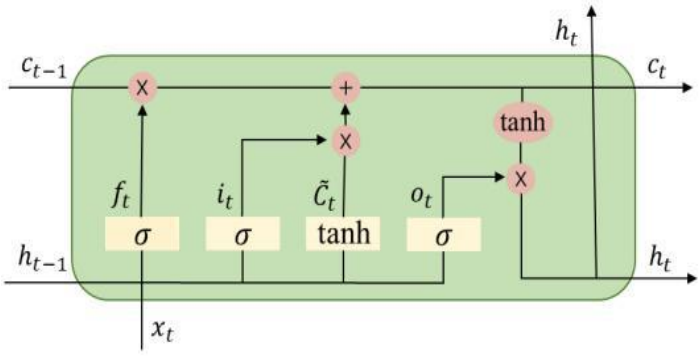


Figure 1. LSTM Structure Diagram

The Input Gate is responsible for deciding how much new information should be added to the cell state at the current time step. The Forget Gate determines what information should be discarded from the cell state at the current time step. The introduction of these two gates allows LSTM to selectively retain or forget information at different time points in the sequence.

3.2. Attention Mechanism

Figure 2 shows the architecture of the encoder part of a Transformer model [11]. The core of this architecture lies in its Multi-Head Attention mechanism, which allows the model to focus on different pieces of information when processing sequence data, thereby capturing richer contextual relationships.

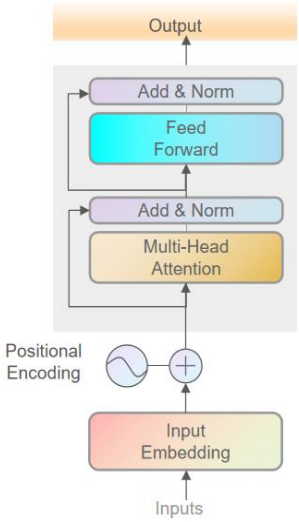


Figure 2. Attention Mechanism Diagram

The input data is first transformed into embedding vectors through the Input Embedding layer. This process maps the features in the sequence to a continuous vector space to capture semantic relationships between different time points. Subsequently, Positional Encoding is added to the embedding vectors to provide positional information for each element in the sequence, as the Transformer model itself lacks a sequence processing mechanism. Next, the data is processed through the Multi-Head Attention mechanism, where multiple attention heads work in parallel. Each head independently calculates attention weights, which are then combined to obtain a richer representation.

Applying the self-attention mechanism to the output of each time step in an LSTM (Long Short-Term Memory) network can further improve the model's performance, especially when handling long sequence data.

3.3. Lasso Regression

Lasso regression is a widely used regression method in machine learning [12]. It introduces an L1 regularization term to promote sparsity in the model coefficients, enabling automatic feature selection. The basic principle of Lasso regression is to add a regularization term to the least squares objective function, as shown in the minimization expression (3.5).

$$||Y - X\beta||^2 + \lambda||\beta||_1 \tag{3.5}$$

In this expression, Y is the vector of observed values, X is the feature matrix, β is the regression coefficient vector, and λ is the hyperparameter that controls the strength of regularization. The introduction of the L1 regularization term forces the coefficients of some features to shrink to zero, thereby achieving feature selection.

When performing Lasso regression, it is necessary to choose an appropriate regularization parameter λ. This can be done using cross-validation, for example, by using LassoCV or GridSearchCV for hyperparameter tuning. During the feature selection process, as λ increases, the coefficients of less important features gradually become zero, while the coefficients of important features remain non-zero. By observing the changes in coefficients for different α (equivalent to λ), it is possible to determine which features are important.

The advantage of Lasso regression in feature selection lies in its interpretability. Since it can set the coefficients of irrelevant or redundant features to zero, the resulting model is more compact. Additionally, Lasso regression is well-suited for high-dimensional datasets, as it can alleviate the curse of dimensionality through feature selection.

4 Data Preprocessing and Visualization

4.1. Data Preprocessing

Table 1 shows a portion of the data used in this study, which records the daily reported new and cumulative confirmed COVID-19 cases, as well as new deaths in China (country code: CN) within the World Health Organization's Western Pacific Region (WPR) from April 13, 2020, to April 21, 2020. Each record includes the report date (**Date reported**), country code, country name, WHO region, the number of new confirmed cases on that day (**New cases**), the total number of cumulative confirmed cases (**Cumulative cases**), and the number of new deaths on that day (**New deaths**), reflecting the development trend and statistical changes during this period.

Table 1. A Portion of the Raw Data

Date reported	Country code	Country	WHO region	New cases	Cumula-tive cases	New deaths	Cumula-tive deaths
2020/4/13	CN	China	WPR	115	83597	2	3351
2020/4/14	CN	China	WPR	99	83696	0	3351
2020/4/15	CN	China	WPR	49	83745	1	3352
2020/4/16	CN	China	WPR	52	83797	0	3352
2020/4/17	CN	China	WPR	27	83824	0	3352
2020/4/18	CN	China	WPR	356	84180	1290	4642
2020/4/19	CN	China	WPR	21	84201	0	4642
2020/4/20	CN	China	WPR	36	84237	0	4642
2020/4/21	CN	China	WPR	13	84250	0	4642
2020/4/22	CN	China	WPR	37	84287	0	4642
2020/4/23	CN	China	WPR	15	84302	0	4642

The dataset preprocessing mainly consists of the following steps:

Data Cleaning: Handling missing values is a crucial task during the data preprocessing phase, as it directly affects the accuracy of subsequent data analysis and modeling. Therefore, this study adopts interpolation methods, which are more suitable for estimating missing values. Linear interpolation is used to fill in the missing data, which is particularly effective for time-series data or data with obvious trends.

Outlier Detection: To ensure the quality of the data, outlier detection is performed to identify and handle potential extreme values. A common method used is the standard deviation-based detection approach. This method assumes that the data follows a normal distribution, and any data point that is more than a certain number of standard deviations away from the mean is considered an outlier. After the initial processing of the dataset, the mean and standard deviation of each column can be calculated to identify all data points that deviate more than a specified number of standard deviations from the mean. Once outliers are identified, they can be either deleted or replaced with other values (such as the median, mean, or boundary values) depending on the specific situation.

4.2. Data Presentation

Subsequently, this study performs descriptive statistics on the collected data, as shown in **Table 2**. The data covers 1,724 days of epidemic statistics, including daily new confirmed cases, cumulative cases, daily new deaths, and cumulative deaths.

The data shows that, on average, approximately 57,645 new confirmed cases were reported daily, with an average of about 37,676,665 cumulative cases. The average daily new deaths were around 70, while the average cumulative deaths were about 49,697.

In addition, the standard deviations for new cases and cumulative cases were 457,776 and 47,068,774, respectively, indicating significant daily fluctuations in case numbers. Similar trends were observed for new deaths and cumulative deaths, but the magnitude of the fluctuations was smaller.

Table 2. Descriptive Statistics of COVID-19 Data in China

	New cases	Cumulative cases	New deaths	Cumulative deaths
count	1724	1724	1724	1724
mean	57645	37676665	70	49697
std	457776	47068774	1095	53633
min	0	1	0	0
25%	23	102167	0	4848
50%	104	1716445	0	15630
75%	1464	99296718	19	121536
max	6966046	99380363	44047	122358

These statistics suggest that during the early stages of the pandemic, new cases and deaths were relatively low, but as the pandemic progressed, these numbers surged, particularly during the peak of the outbreak.

Moreover, the **maximum number of new cases** reached 6,966,046, and the **maximum cumulative cases** totaled 99,380,363, reflecting extreme circumstances on certain days during the pandemic's peak. The **maximum cumulative deaths** reached 122,358, highlighting the severe impact the pandemic had on global health.

5 Prediction Models and Results Analysis

5.1. Problem Definition

Time series forecasting refers to the prediction of the unknown system states over time. This paper primarily focuses on predicting the cumulative COVID-19 infection cases, which essentially involves building an appropriate model based on collected daily case data to predict the number of new cases in the future. Below is the definition of the cumulative infection case prediction problem.

Forecasting Problem Definition: For the multi-step prediction problem of COVID-19 cumulative cases, assume that we have historical time steps of data observations for the new COVID-19 cases $x = (X_1, X_2, \dots, X_p) \in R^{Q \times N}$.Where p represents the dimension of the features. Our goal is to predict the target values for the next Q time steps $y = (Y_{p+1}, Y_{p+2}, \dots, Y_{p+Q}) \in R^{Q \times N}$. The process of predicting the cumulative number of infections can be represented as equation (5.1)

$$Y_{p+1}, Y_{p+2}, \dots, Y_{p+Q} = f(G;\ X_1, X_2, \dots, X_p) \tag{5.1}$$

5.2. Lasso Regression-based Cumulative Infection Prediction

From Table 3, we can summarize the Mean Absolute Error (MAE) and Relative Error (RE) of the Lasso model for different prediction horizons in 2020. The data indicates that as the prediction horizon increases, both MAE and RE also increase, suggesting that the prediction error of the Lasso model grows as the forecast period extends. Specifically, the 3-day forecast has the smallest MAE and RE, while the 30-day forecast has the largest MAE and RE.

Table 3. Summary of Lasso From 2020

Horizon	Mean Absolute Error (MAE)	Relative Error (RE)
3 days	8575221.91	8.63%

5 days	8613817.86	8.67%
7 days	8651950.19	8.76%
10 days	8708268.21	8.84%
14 days	8781684.2	8.96%
21 days	8905424.0	8.96%
30 days	9055299.820	9.11%

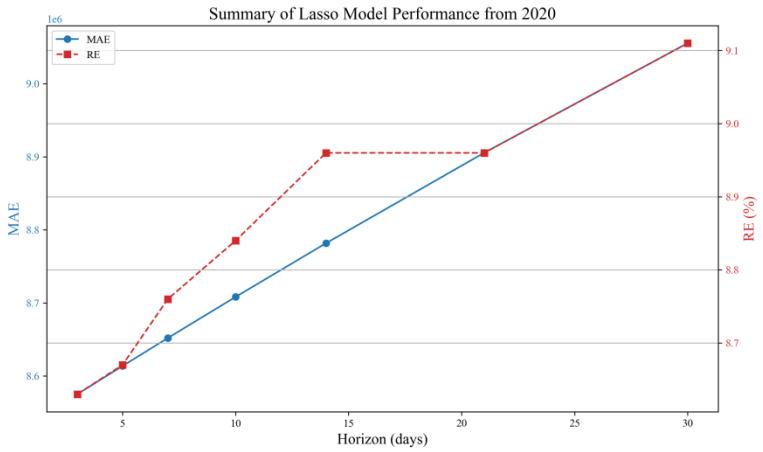


Figure 3. MAE and RE as a Function of Prediction Horizon.

This trend is expected because longer-term forecasts typically involve more uncertainty and variability, making accurate predictions more difficult. The increase in MAE indicates that the average absolute difference between the predicted and actual values is growing, while the rise in RE shows that the error percentage relative to the actual values is also increasing.

5.3. Prediction of Cumulative Infections Based on LSTM

Although traditional Lasso regression can also predict the future cumulative number of infections, the error gradually increases as the prediction horizon expands, leading to a decrease in performance. Meanwhile, both the MAE and RE of the Lasso model are relatively large, making it unsuitable for accurate applications.

Therefore, this study uses the deep learning LSTM model for predicting cumulative infections. Compared to the Lasso model, the LSTM model is capable of extracting trend features from the time series and automatically quantifying these features, which can then be used to predict the number of infections for a given future time horizon.

Table 4. LSTM Network Architecture Diagram

Network Layer Name	Network Layer Shape	Meaning
InputLayer	(None,LookBack,1)	Input as historical LookBack time steps
Dense	(None,LookBack,64)	Non-linear Mapping
LSTM	(None,32)	LSTM Features
Dense	(None,Horizon)	Prediction Layer

In Table 4, the input layer receives historical data, where "None" represents any batch size, LookBack indicates the number of historical time steps considered, and "1" means each time step has

one feature, i.e., the number of infections. Next is a fully connected layer that maps the input data into a 3D tensor with LookBack time steps and 64 neurons. This is followed by an LSTM layer, a long short-term memory network layer specifically designed to capture long-term dependencies in time series data, with an output feature size of 32. Finally, there is a Dense prediction layer that maps the output of the LSTM layer to the predicted results, where Horizon represents the number of future time points the model needs to forecast. The overall design of the model enables it to effectively process and predict time series data.

Table 5. Summary of LSTM From 2020

Horizon	Mean Absolute Error (MAE)	Relative Error (RE)
3 days	3307105.52	1.2%
5 days	3260368.28	3.2%
7 days	3140938.28	3.27%
10 days	2619542.4	2.7%
14 days	984467.71	0.9%
21 days	993624.71	0.11%
30 days	1586796.86	1.6%

In Table 5, the MAE gradually decreases, indicating that the LSTM model achieves higher prediction accuracy in the short term, with the error decreasing over time.

Regarding long-term predictive capabilities, the results for the 21-day and 30-day predictions show relatively low MAE and RE values, which may indicate that the LSTM model has a certain advantage in long-term forecasting, especially in the 30-day forecast where the RE is only 0.11%, demonstrating the model's high accuracy in long-term predictions.

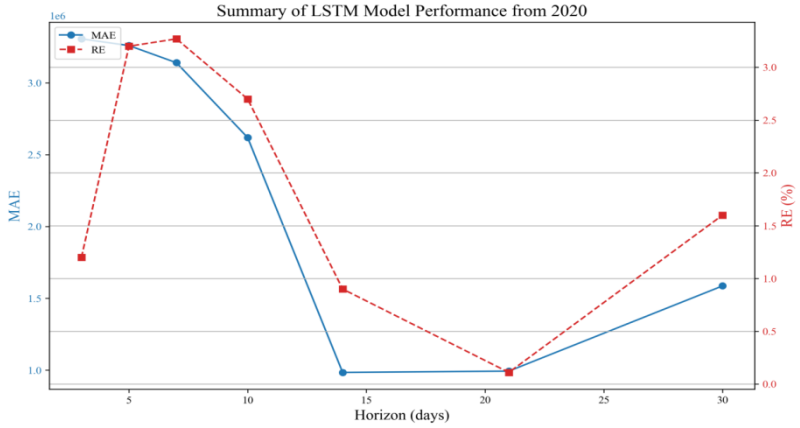


Figure 4. LSTM Model MAE and RE Trend with Varying Prediction Lengths.

Figure 4 shows the trend of MAE and RE for different forecast horizons. It can be observed that the values fluctuate significantly, which may be closely related to seasonality or periodic patterns.

6 Prediction of Cumulative Infections Based on MultiScale-LSTM-Attention

From the previous sections, it can be observed that LSTM performs better than machine learning methods. However, there are still fluctuations in LSTM predictions, which may be due to non-stationary factors at different periods. Existing models often only consider fixed-length historical inputs and extract overall time series trend features, ignoring the impact of different cycles within the data. These cyclical factors are crucial for addressing non-stationarity. Traditional LSTMs only consider the overall

30-day sequence, and some CNN-LSTM models can only consider a fixed number of days, failing to account for changing days.

Therefore, building on the previous work in the sections, this section introduces a multi-scale LSTM model with an attention mechanism to enhance the robustness of the model. Figure 5 shows the specific structure of the Multi-Scale Long Short-Term Memory (LSTM) network. It is designed to capture features at different time scales. The analysis of the figure is as follows:

Input Sequence: The input sequence at the top, denoted as $X_1 - X_t$, is processed by individual LSTM units.

Multi-Scale Processing: The figure shows three LSTM layers at different time scales, labeled as $T = 3$, $T = 5$, $T = 7$. These layers process input sequences with varying lengths of time windows. The $T = 3$ layer processes three consecutive inputs, $T = 5$ processes five consecutive inputs, and $T = 7$ processes seven consecutive inputs.

Feature Extraction: Each LSTM layer at each scale extracts features from its corresponding time window, such as patterns, trends, or other important information in the sequence.

Output: After the lowest layer LSTM, there is an output layer that generates the final prediction or classification result.

In summary, the advantage of the multi-scale LSTM model lies in its ability to simultaneously capture both short-term and long-term temporal dependencies. By combining features from these different scales, the model's performance is improved.

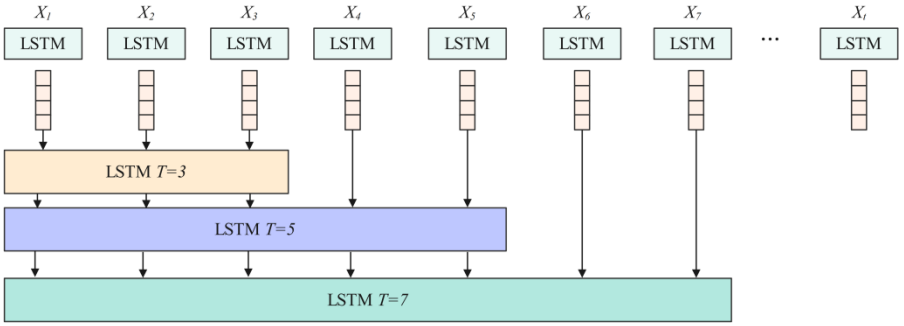


Figure 5. MultiScale-LSTM-Attention Model

7 Comparison of Results between LSTM and MSLA

LSTM demonstrated good performance in previous prediction results, and MLSA further improved upon LSTM by adding multi-scale and attention mechanisms. To validate the effectiveness of MLSA and the added modules, an ablation comparison was conducted.

Table 6. MAE Comparison between LSTM and MLSA

Horizon	LSTM	MLSA
30 days	927.66	786.4

In Table 6, for the 30-day forecast period, the prediction result from LSTM is 927.66, while the prediction result from MLSA is 786.4, indicating that MLSA performs significantly better than LSTM.

Table 7. The results of the ablation experiments for the MLSA model.

Module	MAE
+Attention	803.4
+MultiScale-3	850.32
+MultiScale-5	845.33

+MultiScale-7	820.34
+MultiScale-10	812.56
+MultiScale-14	810.73
+MultiScale-21	800.65

When evaluating the prediction performance of different modules, the Mean Absolute Error (MAE) is used as the metric. MAE represents the average absolute difference between the predicted values and the actual values; the lower the MAE, the higher the prediction accuracy of the model.

Figures 6 and Table 7 show significant differences in performance for each module over a 30-day prediction horizon. First, the "+Attention" module achieved the lowest MAE value of 803.4, indicating that the attention mechanism significantly improves prediction performance. Next, the MultiScale series of modules demonstrated a trend of decreasing MAE as the scale parameter increased. Specifically, the MAE decreased from 850.32 for MultiScale-3 to 800.65 for MultiScale-21, showing a consistent downward trend. This suggests that larger scale parameters help improve the model's prediction accuracy. However, even the optimal MultiScale-21 configuration did not outperform the +Attention and MLSA models in terms of MAE. This result indicates that, although the MultiScale model can be gradually optimized by adjusting the scale parameter, it does not provide higher prediction accuracy than the +Attention/MLSA configuration in the current setup.

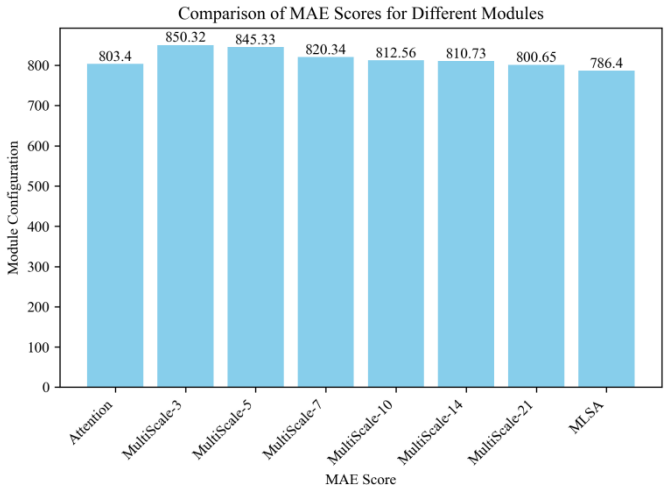


Figure 6. The ablation results of different modules

Based on the above analysis, if the goal is to select the most accurate prediction model, the +Attention or MLSA models should be prioritized based on the MAE metric. Both of these models not only offer the lowest prediction errors, but their performance is also entirely consistent, providing a reliable choice for practical applications. Additionally, incorporating features from different scales significantly improves prediction performance, especially when the scales align with non-stationary cycles. For example, when the non-stationary cycle is 7, using a scale of 7 significantly enhances the prediction performance.

8 Conclusions

This study successfully developed a comprehensive multi-model COVID-19 prediction system by integrating machine learning and deep learning techniques. The system combines Lasso regression, Long Short-Term Memory (LSTM) networks, and a novel Multi-Scale Cumulative Infection Prediction model based on an attention mechanism (MSLA), significantly enhancing prediction accuracy and reliability. Relying on COVID-19 data from China, the study carefully performed data preprocessing and feature engineering to ensure data accuracy and adequate information extraction. The training, evaluation, and visualization of the models were thoroughly documented, providing clear guidance for future model optimization.

The integration of the MSLA model allows this research to simultaneously account for multiple periodic factors, effectively addressing the non-stationary nature of epidemic data. Experimental results reveal that the MSLA model outperforms the traditional LSTM model in 30-day long-term predictions, reducing the Mean Absolute Error (MAE) by 15.22%, offering more accurate scientific support for public health decision-making.

Overall, this study not only provides an innovative and effective tool for COVID-19 prediction but also offers valuable experience and methodology for addressing similar pandemics in the public health field. As the pandemic continues to evolve, the methodology and framework presented in this study will provide strong support for global epidemic monitoring and response, helping to mitigate the impact of the epidemic on social economies and public health systems.

REFERENCES

1. Ciotti, M., Ciccozzi, M., Terrinoni, A., Jiang, W. C., Wang, C. B., & Bernardini, S. (2020). The COVID-19 pandemic. *Critical Reviews in Clinical Laboratory Sciences*, *57*(6), 365–388. https://doi.org/10.1080/10408363.2020.1783198
2. Santosh, K.C. COVID-19 Prediction Models and Unexploited Data. *J Med Syst* 44, 170 (2020). https://doi.org/10.1007/s10916-020-01645-z
3. Singh, R. K., Rani, M., Bhagavathula, A. S., Sah, R., Rodriguez-Morales, A. J., Kalita, H., ... & Kumar, P. (2020). Prediction of the COVID-19 pandemic for the top 15 affected countries: Advanced autoregressive integrated moving average (ARIMA) model. JMIR public health and surveillance, 6(2), e19115. https://doi.org/10.2196/19115
4. Alabdulrazzaq, H., Alenezi, M. N., Rawajfih, Y., Alghannam, B. A., Al-Hassan, A. A., & Al-Anzi, F. S. (2021). On the accuracy of ARIMA based prediction of COVID-19 spread. Results in Physics, 27, 104509. https://doi.org/10.1016/j.rinp.2021.104509
5. Heidari, A., Jafari Navimipour, N., Unal, M. *et al.* Machine learning applications for COVID-19 outbreak management. *Neural Comput & Applic* 34, 15313–15348 (2022). https://doi.org/10.1007/s00521-022-07424-w
6. Alakus, T. B., & Turkoglu, I. (2020). Comparison of deep learning approaches to predict COVID-19 infection. Chaos, Solitons & Fractals, 140, 110120. https://doi.org/10.1016/j.chaos.2020.110120
7. Shahid, F., Zameer, A., & Muneeb, M. (2020). Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM. Chaos, Solitons & Fractals, 140, 110212. https://doi.org/10.1016/j.chaos.2020.110212
8. Islam, M. Z., Islam, M. M., & Asraf, A. (2020). A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images. Informatics in medicine unlocked, 20, 100412. https://doi.org/10.1016/j.imu.2020.100412
9. Shah, P. M., Ullah, F., Shah, D., Gani, A., Maple, C., Wang, Y., & Islam, S. U. (2021). Deep GRU-CNN model for COVID-19 detection from chest X-rays data. Ieee Access, 10, 35094-35105. https://doi.org/10.1109/ACCESS.2021.3077592
10. Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), 1235-1270. https://doi.org/10.1162/neco_a_01199
11. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., & Wang, Y. (2021). Transformer in transformer. Advances in neural information processing systems, 34, 15908-15919.
12. Ranstam, J., & Cook, J. A. (2018). LASSO regression. Journal of British Surgery, 105(10), 1348-1348. https://doi.org/10.1002/bjs.10895
13. Van Tinh, N. (2020). Forecasting of COVID-19 confirmed cases in Vietnam using fuzzy time series model combined with particle swarm optimization. Comput Res Prog Appl Sci Eng, 6(2), 114-120. https://crpase.com/archive/CRPASE-Vol-06-issue-02-20802699.pdf
14. Song, J., Xie, H., Gao, B., Zhong, Y., Gu, C., & Choi, K. S. (2021). Maximum likelihood-based extended Kalman filter for COVID-19 prediction. Chaos, Solitons & Fractals, 146, 110922. https://doi.org/10.1016/j.chaos.2021.110922
15. Chen Guoxin (2024) Prediction of the dynamics covid19 epidemic process of using the Lasso regression model (master diploma) V. N. Karazin Kharkiv National University.

Качанов Станіслав Андрійович
здобувач третього (освітньо-наукового) рівня вищої освіти за спеціальністю 122 Комп'ютерні науки кафедри теоретичної та прикладної інформатики
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: staskachanov2000@gmail.com
https://orcid.org/0009-0002-6938-6717

Чень Госінь
здобувач другого (магістерського) рівня вищої освіти за спеціальністю 122 Комп'ютерні науки, освітньо-професійної програми "Інформатика" кафедри теоретичної та прикладної інформатики
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: guoxin.chen@student.karazin.ua;
https://orcid.org/0009-0004-0502-3735

Морозова Анастасія Геннадіївна
к.т.н., доцент закладу вищої освіти кафедри теоретичної та прикладної інформатики
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: a.morozova@karazin.ua
https://orcid.org/0000-0003-2143-7992

Руккас Кирило Маркович
д.т.н, доцент, професор закладу вищої освіти кафедри теоретичної та прикладної інформатики
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: rukkas@karazin.ua
https://orcid.org/0000-0002-7614-0793

Прогнозування динаміки епідемічного процесу COVID19 з використання моделі Ласо регресії

У роботі запропоновано об’єднати методи машинного та глибокого навчання для прогнозування пандемії COVID-19.
Актуальність. Глобальний спалах пандемії COVID-19 мав великий вплив на системи здоров’я та соціально-економічні структури в усьому світі, підкреслюючи нагальну потребу в ефективних інструментах прогнозування для допомоги в прийнятті рішень. Робота присвячена побудові мультимодельної структури для прогнозування епідемії шляхом інтеграції передових методів математичного моделювання та теорії прогнозування.
Мета. Метою роботи було проаналізувати методи та алгоритми для кумулятивного прогнозування випадків COVID-19з метою надання наукової підтримки для прийняття рішень у сфері охорони здоров’я.
Методи дослідження. Методи дослідження базуються на сучасних теоріях математичного моделювання, штучного інтелекту, епідеміологічної діагностики, теорії прогнозування, а саме: регресія Лассо, мережі довгострокової короткочасної пам’яті (LSTM) і моделі LSTM-Attention.
Результати. Було досліджено застосування моделі Лассо регресії та мережі довгострокової короткочасної пам’яті (LSTM) для кумулятивного прогнозування випадків COVID-19з метою надання наукової підтримки для прийняття рішень. В роботі детально описано процеси попередньої обробки даних, навчання моделі, оцінювання та візуалізації для підтримки узагальнення та адаптивності в динамічному сценарії пандемії. Крім того, була запропонована модель Multi-Scale LSTM-Attention (MSLA) для вилучення багатоперіодних ознак із вхідних послідовностей. Ці функції є критично важливими для вирішення проблеми нестаціонарності даних.
Висновки. Вирішено задачу розробки системи прогнозування COVID-19 шляхом інтеграції методів машинного та глибокого навчання. Розроблена система поєднує в собі регресію Лассо, мережі довгострокової короткочасної пам’яті (LSTM) і нову багатомасштабну модель кумулятивного прогнозування зараження на основі механізму уваги (MSLA), що значно підвищує точність і надійність прогнозування.

Ключові слова: *прогнозування, нейронні мережі, Lasso Regression Model, LSTM, COVID-19, Машинне навчання, Глибоке навчання, Multi-scale model, бази даних.*

Надійшла у першій редакції 12.08.2024, в останній - 29.09.2024.
The first version has been received on 12.08.2024, the final version - on 29.09.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-07

УДК (UDC) 519.7, 004.8

Novikov Artem *PhD student, Department of Artificial Intelligence Systems*
Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine, 61022
email: artem.slick@gmail.com;
https://orcid.org/0009-0004-5914-7098

Smyrnov Vadym *Master's student, Department of Artificial Intelligence Systems*
Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine, 61022
e-mail: vadym.smyrnov@student.karazin.ua;
https://orcid.org/0009-0000-1743-8541

Yanovsky Volodymyr *Doctor of physical and mathematical sciences; Professor*
"Institute for Single Crystals" of National Academy of Sciences, Nauky ave. 60, Kharkiv, Ukraine, 61001
e-mail: yanovsky@isc.kharkov.ua; **https://orcid.org/0000-0003-0461-749X**

Fractal properties of neural networks

The work is devoted to the research of neural networks' properties, which have been extremely intensively used in various applied directions recently. The study of their general and fundamental properties is becoming more and more **actual** due to their wide application.

The key goal of the work is to investigate the reaction field of an artificial neural network in the space of all possible input signals of a certain length. Based on the example of a simple perceptron, zones where the reaction field of the neural network has a structurally complex nature are studied.

Research methods: To research an output signal field, software was developed, which allowed modeling and visualization of the output signal field over the space of all input signals. The software also allowed changing of activation functions, weights, and thresholds of each neuron, which made it possible to research the influence of all these factors on the structural complexity of the output signal field.

As a result, the study established that, in general, within the space of input signals, there are shadow zones where the response field of the neural network has a self-similar fractal structure. Conditions for the appearance of symmetry of such structures were determined, the influence of activation functions, weights and thresholds of network neurons on the properties of fractal structures was investigated. It was revealed that the input layer of neurons predominantly influences these properties. Dependences of the fractal dimension of the structures on the neuron weights were obtained. Changes occurring with the increase in the dimensionality of the input signal space were discussed.

The presence of shadow zones with a fractal output signal field is important for understanding the functioning of artificial neural networks. Such shadow zones define regions within the input signal space where the neural network's response is extremely sensitive even to minute changes in input signals. This sensitivity leads to a fundamental change in output signals with a slight change in input signals.

Keywords: *artificial neural network, space of input signals, field of output signals, perceptron, artificial neuron, fractal structures, fractal dimension.*

Як цитувати: Novikov A. O., Smyrnov V. M., Yanovsky V. V. Fractal properties of neural networks. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.66-79. https://doi.org/10.26565/2304-6201-2024-64-07

How to quote: A. O. Novikov, V. M. Smyrnov and V. V. Yanovsky, "Fractal properties of neural networks", *Bulletin of V.N. Karazin Kharkiv National University, series "Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 66-79, 2024. https://doi.org/10.26565/2304-6201-2024-64-07

1 Introduction

The emergence of artificial neural networks begins with the work of Warren McCulloch and Walter Pitts [1], in which they proposed the concept of an artificial neuron and artificial neural networks based on electronic circuits. The artificial neuron itself was a simplified abstraction of the understanding of the natural neuron's functioning that existed at the time. Further development of these ideas led to the creation of a perceptron in the 1960s by Frank Rosenblatt [2, 3], which is a classical neural network that solves the problem of dividing signals into two classes. Initially, the perceptron sparked high hopes for artificial intelligence, bolstered by Rosenblatt's statements at a conference [4] in 1958. However, it quickly became clear that the perceptron struggled to discern certain patterns. A particularly significant decline in interest in the perceptron occurred after the publication of M. Minsky and S. Papert's book [5], which criticized the single-layer perceptron.

For an extended period, neural networks remained outside of active scientific research. Interest in this direction was reignited with the realization that neurons of forward-propagation networks with two or more layers have extraordinary power. The situation changed in the 1980s due to the contributions of John Hopfield [6, 7, 8], who proposed a new perspective on the architecture of neural networks and introduced the ideas of feedback and self-learning algorithms for the first time.

Research on algorithms for linear classification continues even now. For example, the paper [9] proposed a classification algorithm that combines the perceptron algorithm with the Helmbold and Varmuth "leave one output" method.

An overview of existing error limits and new limits for the perceptron algorithm or the perceptron core can be found in [11]. Discussion of the obtained limits go beyond standard margin-loss type bounds, allow for any convex and Lipschitz loss function. The development of information technologies in the 2000s showed a lot of applications of neural networks, such as voice and image recognition [12, 13, 14]. Nowadays, research in the field of neural networks has gained immense popularity [15]. However, the focus on practical applications of neural networks diverted attention from their intrinsic properties, which are not studied enough.

This work delves into the structural properties of neural networks' reactions using a simple two-layer perceptron as an example. The primary focus of this article lies in exploring the properties of the output words' field within the space of all input words of a certain length. It is shown that within the space of input words, under general conditions, there are shadow zones where the field of output words has a fractal structure. The range of scales covered by self-similar fractal patterns is determined by the input words' lengths. The study also uncovers the influence of neuron activation functions and their wights on the fractal dimension of these structures, pinpointing a more significant impact from neurons in the first layer. Typical dependences between changes in the fractal dimension and neuron weights are established.

The obtained results regarding the study on artificial neural network behavior's fractal properties hold theoretical and practical significance for enhancing the stability of neural networks and mitigating the impact of minor input data changes that might otherwise obscure the correct output. Research in this direction can open new prospects for neural networks development and provide deeper understanding and improvement of their functionality.

2 Formulation of the problem

Let's consider the properties of input word processing by a simple two-layer perceptron with two inputs and one output. The input words that this perceptron will process are formulated in some finite alphabet $A = (a_1, \dots, a_n)$. Thus, words in the alphabet A will be the inputs of the perceptron. One of the main characteristics of input words, for example $a_1a_3a_2a_1$, is their length $|a_1a_3a_2a_1| = 4$ or the number of letters in a word. After input sequences processing, we will receive a sequence in some alphabet as the output word of the perceptron. For the sake of simplicity, let's assume that the alphabet of the output signals coincides with the alphabet A. Thus, after processing of two input words with a length of n we will receive an answer or a word of length n in the alphabet A as an output of the neural network. Next, we will explore how the neural network's responses to all possible input words look like.

If we consider words of length n in the alphabet of k letters, then k^n words can be created. Then if we take a unit segment and divide it into k^n segments of length $\frac{1}{k^n}$, one word can be assigned to each of the segments. That is, we can enter specific coordinates – placing all words (sequences) in lexicographic order and assigning them to segments of length $\frac{1}{k^n}$ starting from the first left one (see Fig. 2.1). From now on, we will use such coordinates to identify input words.

Then all input words which are applied to the input of the perceptron correspond to a unit square divided into $k^n \times k^n$ squares with a side length of $\frac{1}{k^n}$. Each pair of input words supplied to the neural network's inputs has corresponding coordinates along the axis x and the axis y. The input word given to the input y, defines the coordinate along the axis y and the word given to the input x, defines the coordinate of the pair along the axis x. Thus, each pair of input words corresponds to one specific square with a side length of $\frac{1}{k^n}$. All output words can be assigned to such coordinates in a completely similar way. This unit square can be considered as an input word space of the perceptron with two input channels. All output words can be visualized by assigning a specific "color" to each small square based on a received response to the corresponding input words. Thus, the output word space of the perceptron

will be represented as a painted unit square. If input words have infinite length, then each dot of the square defines a particular input word, and its color represents a corresponding output word. The details of such divisions are given in Appendix A.

Next, we will use $A = (0,1,2\ldots,9)$ decimal symbols as the alphabet. All neurons have certain weights, which will change in different experiments. All activation functions will be step functions with values defined in the alphabet A. It is vital to receive a network response in the same alphabet. Thus, we will further consider perceptrons with different activation functions. We will be interested in the structure of all output words over all input words of a certain length, as well as the rearrangement of output words as the length of input words increases. In another words, we will explore the field of output words in the space of input words.

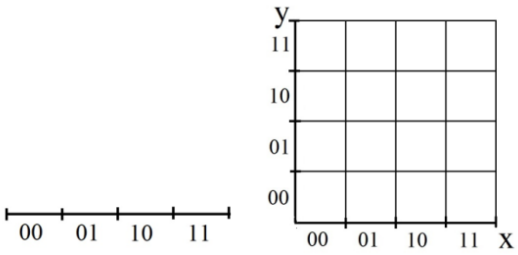


Fig. 2.1. On the left is a division of the unit segment that corresponds to all words of length 2, for simplicity, in the binary alphabet $(0,1)$. *Words are arranged in lexicographic order from left to right. On the right is a unit square with the corresponding coordinates. Each small square corresponds to a certain value of input sequences* (x, y) *of a certain length. Similarly, the unit segment is divided into a larger number of parts that correspond, for example, to words of length 4 and the corresponding coordinates.*

3 Perceptron model

To analyze the properties, a simple model of a two-layer perceptron was built (see Fig. 3.1). It receives input words (x, y) and forms output words z which length matches with the length of the input words. Each neuron of the perceptron weights ω_x, ω_y and a threshold or bias b that can be changed. As discussed before, the values of the activation function correspond to the values of the alphabet $A = (0,1,2\ldots,9)$. The domain of the activation function is all real number. Thus, the general activation function that satisfies the necessary conditions is

$$f(x) = \begin{cases} a_{i_1} & if \quad x < x_1 \\ a_{i_2} & if \quad x_1 \le x < x_2 \\ a_{i_3} & if \quad x_2 \le x < x_3 \\ a_{i_4} & if \quad x_3 \le x < x_4 \\ a_{i_5} & if \quad x_4 \le x < x_5 \\ a_{i_6} & if \quad x_5 \le x < x_6, \\ a_{i_7} & if \quad x_6 \le x < x_7 \\ a_{i_8} & if \quad x_7 \le x < x_8 \\ a_{i_9} & if \quad x_8 \le x < x_9 \\ a_{i_{10}} & if \quad x_9 \le x \end{cases} \qquad (1)$$

where $a_{i_j} \in A$ are the letters of the alphabet A,

x_i satisfy the conditions $x_1 < x_2 < x_3 < x_4 < x_5 < x_6 < x_7 < x_8 < x_9$ and form ten subintervals.

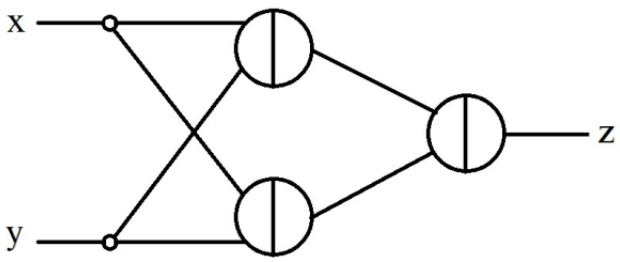


Fig 3.1. The model of the perceptron, which is researched in the work.

Each activation function is defined by a value $x_1, \dots, x_{10}$ and a word $a_{i_1} a_{i_2} \dots a_{i_{10}}$ in the alphabet A. That is, with the given values $x_1, \dots, x_{10}$, we can create 10^{10} different activation functions. It is clear that only a small number of activation functions can be considered in the work. It should be noted that activation functions of different neurons of the perceptron may be different. Additionally, for each neuron, the activation function can divide its domain into intervals of different lengths depending on the choice of weights and threshold of the neuron. Since the argument of the activation function can be values from the interval $[\sigma; (\omega_x + \omega_y)9 + \sigma]$, it is necessary to choose such values of weights, threshold, and type of activation function that the intervals formed by the activation function are close to the interval $[\sigma; (\omega_x + \omega_y)9 + \sigma]$.

Using the developed application, it is possible to choose activation functions, weights, thresholds of neurons, and the length of input words to study the corresponding perceptron response fields. A detailed description of the software can be found in Appendix B.

4 Results of modeling

Fig. 4.1 shows the results of modeling with certain activation functions and neuron weights. On the left is the response field for all input words of length 3, which demonstrates the complex geometric structure of the response field. The part of the answers, which is marked with a blue square, is shown on the right in 10 times enlarged view. Comparing the pictures, it is easy to see the complex, self-similar structure of the answers, which persists as the length of the input words increases. Such structures belong to fractal structures, which have been intensively studied in mathematics and physics (see, for example, [16]). Fractal structures are characterized by a fractal dimension. Having already used this self-similarity, it is possible to calculate the fractal dimension, for example, of black areas [17]. From Fig. 4.1, we can see that the number of black squares is $N = 556688$ with a word length of 3.

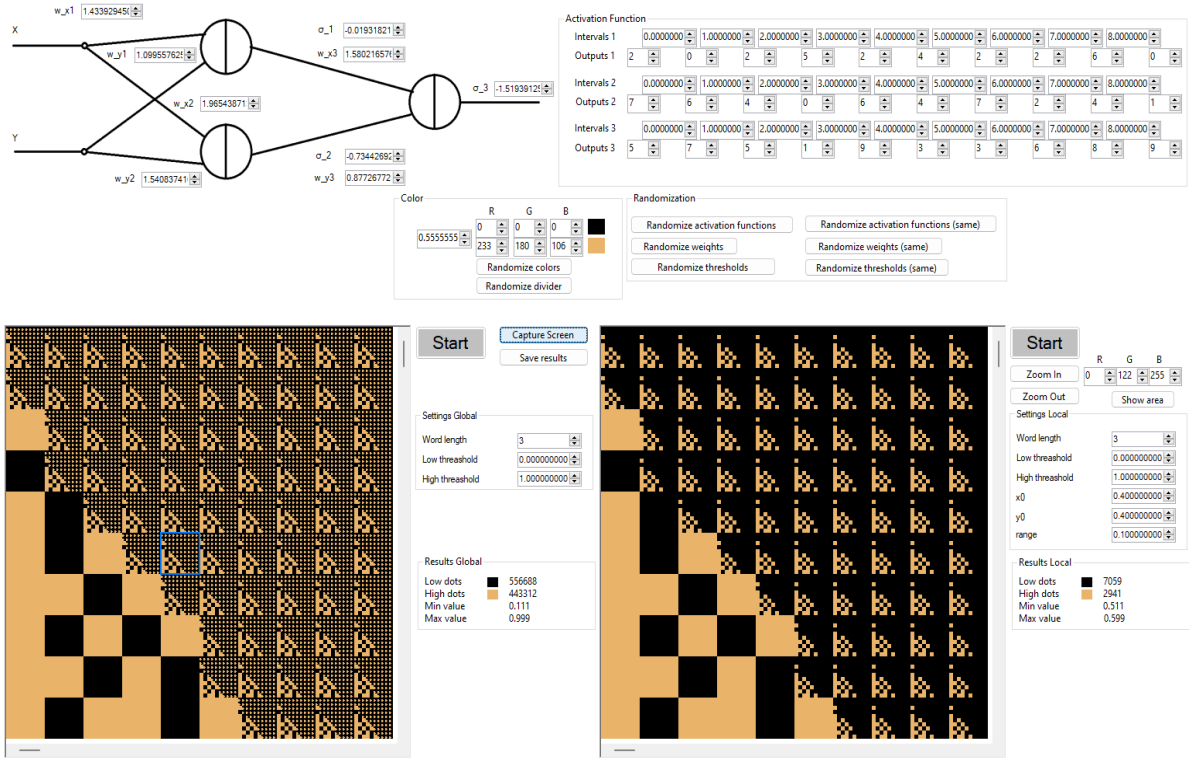


Fig 4.1. An example of a graphic display of output signals with set weights of synapses, for all questions of length 3. On the right, in the upper corner, the activation functions of neurons are shown. Each neuron has its own activation function. The area bounded by the blue square is 10 times enlarged and shown on the left. The comparison of these images shows a self-similar structure on smaller scales. Such self-similarity extends to scales of 10^{-n}, the display of which is impossible for large n due to the limitation of the pixel size.

Thus, by counting the number of black squares N at different input word lengths L, we obtain a linear dependence of $\ln(N)$ on $\ln(\frac{1}{L})$, as shown in Fig. 4.2. Such a linear dependence indicates the fractal structure of the perceptron response field, the fractal dimension of which can be calculated as follows

$$D_F = \frac{\ln(N_2) - \ln(N_1)}{\ln(\frac{1}{L_2}) - \ln(\frac{1}{L_1})},$$

where N_1 and N_2 are the number of black squares with sizes L_1 and L_2, respectively.

That is, in this case, $D_F = \frac{\ln(21563200)-\ln(83)}{\ln(\frac{1}{0,0001})-\ln(\frac{1}{0,1})} \approx 1.805$ is the fractal dimension of the set of a certain answer, marked in black and shown in Fig. 4.1. Such self-similarity is observed for the scales, which are determined by the length of the input words and covers the range of scales from 1 to $\frac{1}{10^n}$, where n is the length of the questions. That is, it occupies a gigantic range of scales. As the length of the words increases, this range increases. Thus, the existence of fractal regions of responses in a simple perceptron is shown.

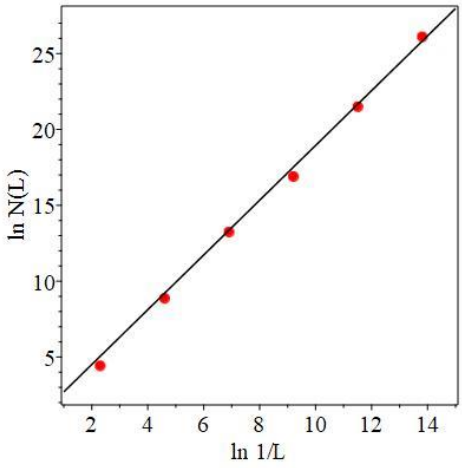


Fig. 4: Graph of the logarithm of the number of black squares N versus the logarithm of the inverse of the side length of one square L. As you can see, a linear relationship is formed, which indicates the presence of a fractal structure the fractal dimension of which is equal to the slope of this line.

It should also be noted that a fractal structure can occur when you select thresholds for the output signals to be displayed that have several decimal places other than 0 that is not less than the length of the output words. For example, if the output words have a length of 3, then a threshold of 0.5555 can be used to create a fractal structure.

Otherwise, the fractality will disappear because when comparing the output signals with the threshold, it is enough to compare only a part of the characters of the output word. For example, if the threshold for displaying output signals is 0.5 and the length of the output words is 3, it is enough to know only the first character of the output word to determine whether the signal exceeds the threshold, which eliminates the need to determine the subsequent response characters. This leads to the elimination of the sensitivity of the structure of the field of output signals to the length of words, which leads to the elimination of fractality.

The presence of such fractal structures in the input word space is important, so changing even the last letter of the input word drastically changes the answer. In other words, there are shadow zones of input words in which the answer fundamentally changes when one last letter in the question is changed. So, for example, if we use the binary alphabet of answers $A = (\text{Yes}, \text{No})$, then when changing the last letter in questions of length 10, instead of the answer “Yes” we can get “No”. Such sensitivity will always occur in shadow zones. By shadow zones we will understand the areas of input words, in which the field of output words has a fractal structure. Such a sensitivity can be used for various purposes. Similar shadow zones and fractal patterns of responses were observed for all tested activation functions.

Let us consider the question of what types of structures can be observed with different activation functions and weights of the neurons. It is clear that it is impossible to model perceptrons with all possible activation functions due to their gigantic number. Therefore, it is necessary to consider some of them, based on theoretical considerations and observations for a number of selected activation functions that were used in the simulation.

4.1 Effect of activation functions

The question then arises, what response patterns can be expected for different activation functions? Let's start with the symmetry of the structures. It is easy to understand that if you use neurons that have

the property of giving the same response when x is replaced by y, then the field of responses will be symmetrical about the unit square diagonal. An example of a symmetrical structure is shown in Fig. 4.2. In general, such symmetry is absent, as shown in Fig. 4.1. The condition when such symmetry is present limits the properties of activation functions of the first layer neurons only. The activation functions of the f_1 of the first and f_2 of the second neuron of the first layer must satisfy the condition

$$\begin{cases} f_1(\omega_1 x + \omega_2 y + b) = f_1(\omega_1 y + \omega_2 x + b) \\ f_2(\tilde{\omega}_1 x + \tilde{\omega}_2 y + b) = f_2(\tilde{\omega}_1 y + \tilde{\omega}_2 x + b) \end{cases} \quad (2)$$

which is satisfied by the equality of weights $\omega_1 = \omega_2$ and $\tilde{\omega}_1 = \tilde{\omega}_2$ regardless of the form of the activation functions. But values of the weights may be different in the first and second activation functions.

Thus, the symmetry of the structures about the diagonal is fulfilled when the weights of the first layer neurons of both synapses are symmetrical. But there may be other solutions to this functional equation, for special activation functions. An example of symmetrical fractal structures is shown in Fig. 4.3. This figure shows the fulfillment of criteria (2) and, accordingly, the lack of influence of the output layer neuron (that is, its weights and activation functions) on the symmetry of fractal structures.

Now let's discuss the types of fractal structures that arise with various activation functions. During the simulation, it was found that various fractal structures appear for all considered activation functions. This means that the presence of shadow zones and the appearance of fractal structures corresponds to the general case. It is clear that it is impossible to consider all cases of activation functions due to their large number. Various fractal structures were obtained by changing activation functions. An example of one of them is shown in Fig. 4.3. As a result of the simulation, it was found that the appearance and fractal dimension of the structures depend on the choice of the activation function and may change.

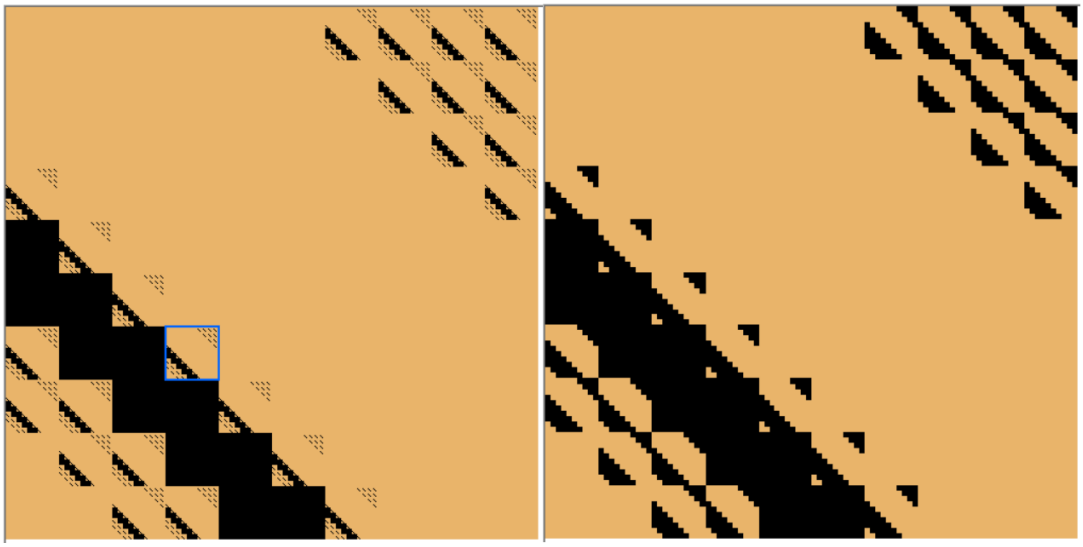


Fig 4.3. An example of a diagonally symmetric self-similar fractal structure. The activation functions of the neurons are different. The first is 6008234346, the second is 9883057840 and the activation function of the output neuron is 6252371258 (see ratio (2)). The first neuron weights are $\omega_x = \omega_y = 0.233$, the weights of the second neuron are $\omega_x = \omega_y = 1.348$ and the third neuron weights are $\omega_x = 1.692$, $\omega_y = 1.076$. The threshold of the first neuron is $\sigma = -0.463$, the second $\sigma = -1.339$ and the third $\sigma = -1.535$. Condition (2) is fulfilled. This structure has a fractal dimension $D_F \approx 1.88$. Shown on the right is a 10 times magnified portion of the fractal structure drawn in the blue box on the left to demonstrate self-similarity of the output field.

All described properties are preserved even when choosing the same activation functions for all neurons of the perceptron. The appearance of the fractal structures changes a little, but no qualitative changes are observed.

Thus, the appearance and fractal dimension of the structures change when activation functions are changed.

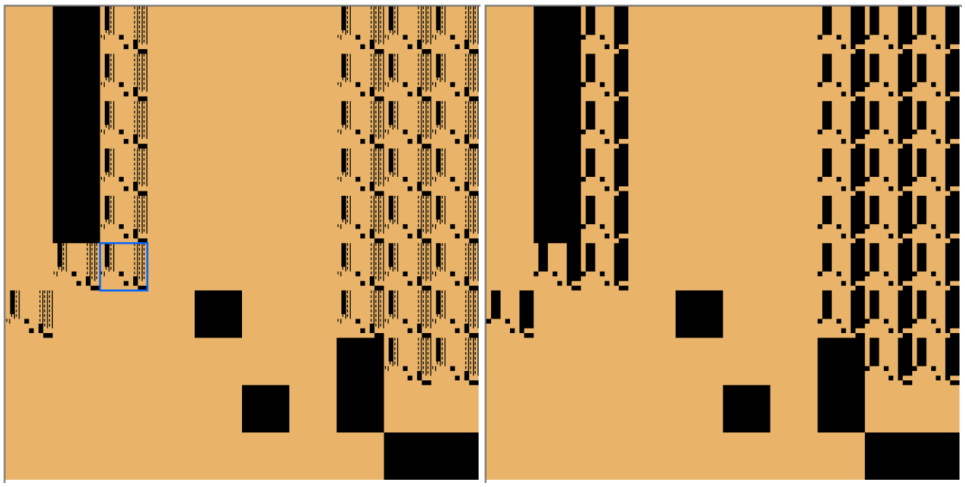


Fig 4.3. Self-similar structure using other activation functions. Activation functions of the neurons are different and defined by the words 2865111974, 2786803702 and 3460777549. Weights of the first neuron are $\omega_x = 1.274$ $\omega_y = 0.088$, *the second neuron weights are* $\omega_x = 0.588$, $\omega_y = 1.775$ *and weights of third neuron are* $\omega_x = 0.823$, $\omega_y = 1.668$. *The threshold of the first neuron is* $\sigma = -0.686$, *the second* $\sigma = -0.195$ *and the third* $\sigma = -0.564$. *Fractal dimension* $D_F \approx 1.85$. *On the right is a 10 times magnified portion of the fractal structure drawn in the blue square on the left to demonstrate self-similarity.*

4.2 Effect of weights of the neurons

Let's consider how alterations of weight values affect fractal structures. It is natural to start with the impact of weights on the symmetry of fractal structures. For instance, changing the weights of neurons, such as those for the symmetric fractal structures mentioned earlier, can break the symmetry about the diagonal. An example demonstrating the loss of symmetry due to alterations of the weight values ω_x of only the first neuron is shown in Fig. 4.5. In the general case of weights and activation functions, such symmetry of fractal structures is absent. Examples of the absence of symmetry under general conditions are shown in Fig. 4.1 and Fig. 4.4.

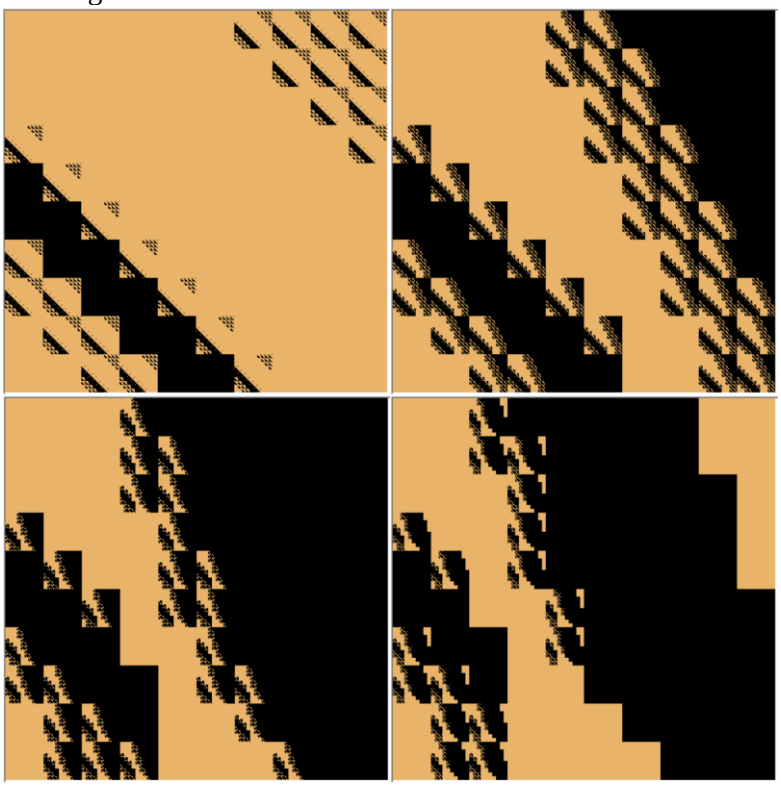


Fig 4.4. In the first figure, a symmetrical structure is generated by a perceptron with activation functions of the neurons defined by the words 6008234346, 9883057840 and 6252371258 and weights of the first neuron $\omega_x = \omega_y = 0.233$, *the second* $\omega_x = \omega_y = 1.348$, *and the third* $\omega_x = 1.692, \omega_y = 1.076$, *and thresholds of the first neuron* $\sigma = -0.463$, *the second* $\sigma = -1.339$, *and the third* $\sigma = -1.535$. *Fractal dimension of the symmetrical*

structure is $D_F \approx 1.871$ Further, the drawings are obtained by changing only values of the weight ω_x of the first neuron. In the following pictures the values of the weight ω_x are 0.253, 0.273 and 0.293 respectively. Fractal dimensions of the structures D_F are 1.966, 1.984 and 1.979 respectively. They show violation of the symmetry when values of the weight ω_x of the first neuron change.

Moreover, changes in values of the weights of the neurons affect the fractal dimension of the structures as well. Examples of such changes are shown in Fig. 4.4 and Fig. 4.5. The fractal dimensions were found for the structures arising from different values of the weights, showing the dependence between fractal dimensions and the weights of the respective neurons. In each case, only one weight of a certain neuron changed. Figure 4.6 shows the corresponding dependencies. Considering the case of different activation functions of the neurons (see Fig. 4.6, top row), the dependences on the input layer neuron weights are symmetrical. In other words, altering the weights of either neuron results in identical dependences. However, the effects of these neurons show different dependencies. The dependence on the weights of output neuron of the perceptron is more intricate, causing a loss of symmetry in the dependencies (see Fig. 4.6). If every neuron of the perceptron has the same activation function, the dependence of the fractal dimension on the weights is simplified. In the lower row of Fig. 4.6 the left side shows the dependence on the weight of any neuron in the first layer, while the right side shows no changes in the fractal dimension with changes in weights of the output neuron. Consequently, the fractal dimension of the structures within the shadow zones is notably more affected by the weights of neurons in the first layer.

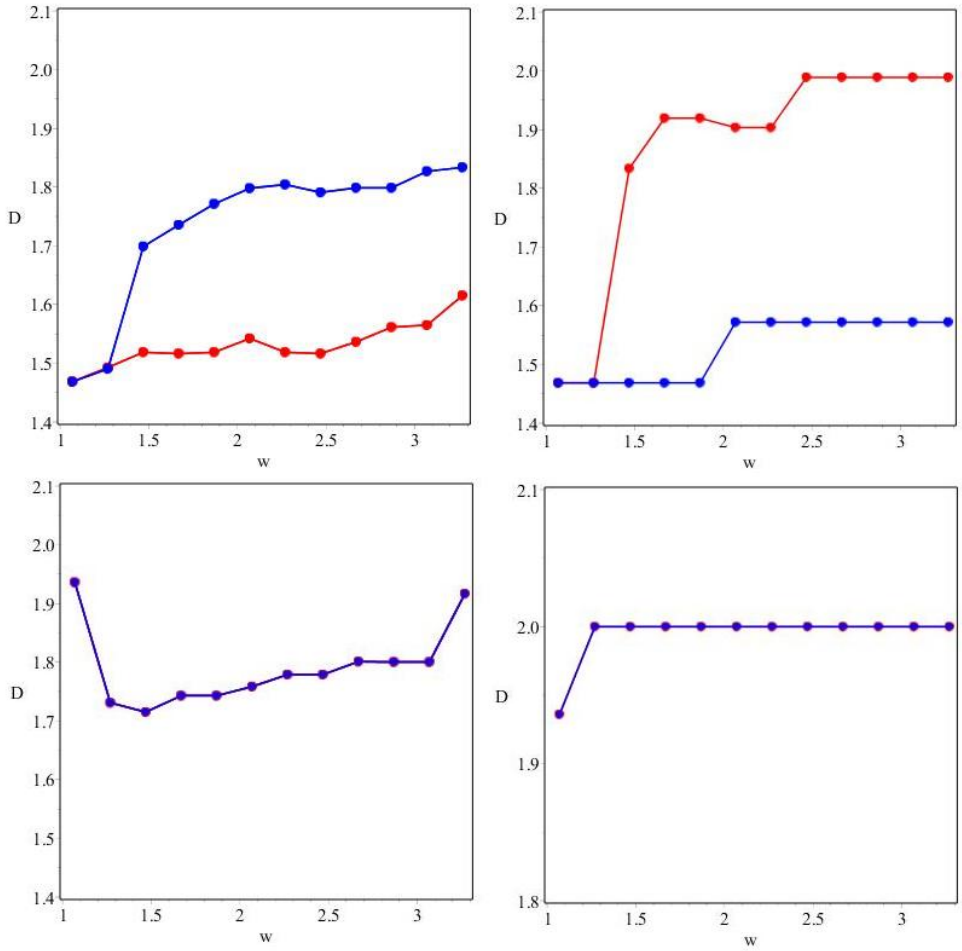


Fig 4.5. The dependence of the fractal dimension on the weights of the neurons of the perceptron. The upper row corresponds to the situation when all activation functions of the neurons are different. The left section stands for the dependence on the weights of the first and second neurons in red and blue, respectively, within the first layer. The right section shows the dependency on the weights of the output neuron, where dependencies on weights ω_x and ω_y are red and blue, respectively. The lower row corresponds to the case of identical activation functions for all neurons. On the left side is the dependency on the weights of any neuron within the first layer, while the right side shows the dependency of the fractal dimension on the weights of the output neuron, which demonstrates its weak influence on the fractal dimension of the structures.

5 Conclusions

Thus, in the general case, there are shadow zones within the space of input words in which the response field of the perceptron has a fractal structure. Under certain conditions explored in the work, fractal structures can be symmetrical about the diagonal of the unit square. Changing the weights of the neurons of the perceptron within the first layer in a certain way can disrupt this symmetry while the activation function and the weights of the output layer have no impact on the symmetry of fractal structures. In the general case, such structures lack symmetry. The fractal dimension of the structures in the shadow zones within the input word space depends both on activation functions of the neurons and their weights. Changes in the weights of the neurons within the first layer affect this characteristic more significantly, while, in the case of identical activation functions, the weights of the output neuron have a weak influence on its value.

It is clear that similar fractality will occur in more complex neural networks with a larger number of inputs. The main difference is in the large dimension of the space of questions and, accordingly, in the impossibility of demonstrating fractality visually. The dimensionality of the space of input words aligns with the number of input channels, impeding direct visual confirmation of fractality. In this case, verifying fractal characteristics may require specialized data processing algorithms, enabling determination of the fractal dimension without visualization. Such algorithms find application in the study of complex systems such as strange attractors in nonlinear dissipative systems with high dimensionality of the phase space [18]. The alternative approach of their determination involves studying certain intersections between the space of input words and the response field, focusing on these intersections of smaller dimensions.

6 Appendix A

Let's discuss some details of the location of the input words in the unit square and their coloring, according to the received output words. For the sake of simplicity, we will use the binary alphabet $A = (0,1)$. In Chapter 2, we discussed the way to assign coordinates to every input of the perceptron. Now we will discuss the changes that occur as the length of the questions increases. Figure 6.1 shows a comparison of the location of two divisions with different sequence lengths.

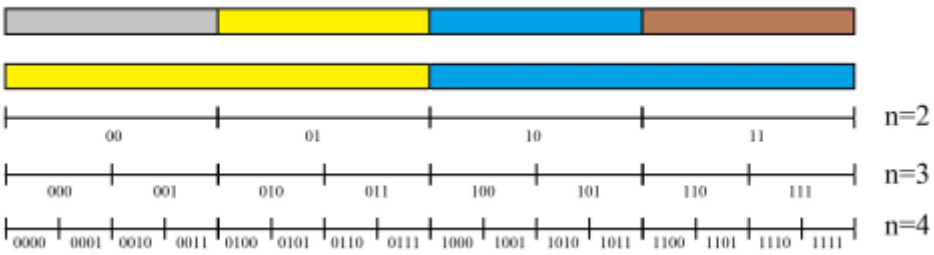


Fig 6.1. Examples of dividing a unit segment for different sentence lengths. The coordinates of the corresponding segments are defined. Below, the length of sentences is 4. It is easy to compare the consistency of a smaller-scale division with a larger-scale one. There are two examples of color palettes shown above that match the corresponding source sentences below, regardless of sentence length.

Such coordinates ensure consistency between the smaller-scale division and the larger-scale one. For instance, if we simplify the corresponding coordinates by truncating with $n = 4$, that is, discarding the last letter, both segments and their coordinates would coincide with the truncation with $n = 3$.

Now let's discuss the visualization of responses to all possible input words. As suggested in Chapter 2, it can be achieved by using some color palette. Any of different palettes can be used for this task. Thus, Fig. 6.1 shows two simple ways of coloring with two colors and 4 colors respectively and the relationship of the palette with words. The color palette stays the same for any division of segments. The color of a word is defined by the color above its position. Of course, it is possible to use more detailed palettes with more colors, including a palette where each word has its own color. Also, in some cases, when a specific sentence must be visualized, a specific color can be assigned to it.

7 Appendix B

To conduct experiments with the perceptron model, we developed a software that allows a user to perform the following actions:

1. setting parameters for the model of the perceptron;
2. calculating output values for different combinations of input signals of the perceptron;
3. setting parameters for graphical representation of results;
4. controlling the calculation process;
5. viewing the result of calculations in the form of a graphic representation of the output signals;
6. viewing the number of squares of a certain color and size to determine the fractal dimension of the constructed image;
7. uploading the results to a file.

The general view of the user interface is shown in Fig. 4.1. Before performing the calculations, all the characteristics of the neural network must be set.

Thus, to set the values of the weights of each connection, the corresponding fields “w_x1”, “w_y1”, “w_x2”, “w_y2”, “w_x3”, “w_y3” must be filled in (Fig. 7.1). To set the threshold values of each neuron, the corresponding fields “σ_1”, “σ_2”, “σ_3” must be filled in (Fig. 7.1).

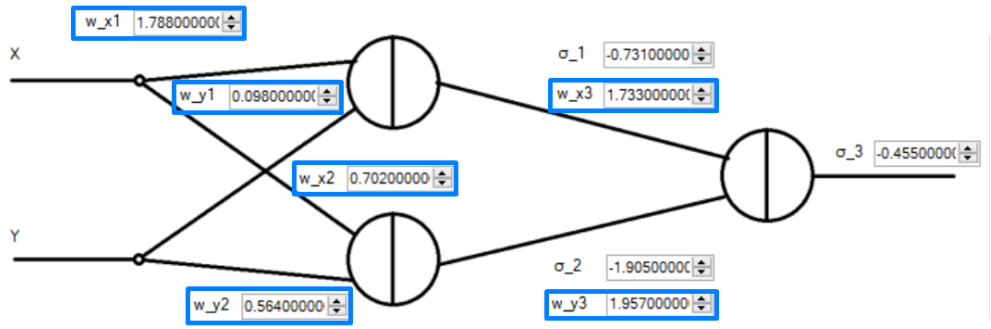


Fig 7.1. Fields for setting the weights of each connection of the perceptron model (highlighted in blue), fields for setting neuron thresholds are not highlighted.

The user can also define activation functions for each neuron. To do this, you need to set the values of the ends of the intervals, in which the corresponding output characters will be set, and the values of these output characters for each interval must be set as well. All these fields are filled in the upper right corner of the interface (see Fig. 4.1).

The software allows the user to construct two images to compare the structure of the global signal field and its parts. To do this, two separate modules were developed. Each of them has a picture box for visual representation of the output field, a “Start” button for starting calculations, “Word length” field to set the length of words, “Low threshold” and “High threshold” fields for setting the minimum and maximum intensity thresholds of the output signals for display, fields “Low dots” and “High dots” for displaying the number of squares painted with the corresponding color, fields “Min value” and “Max value” for displaying the minimum and maximum values of the output signals (Fig. 4.1).

In addition, in the right module for displaying the results there are buttons "Zoom In" and "Zoom Out" to zoom in and out the image, respectively, fields “R”, “G”, “B” to set the color to highlight the part of the global field that is considered on the right, the “Show area” button to select the part of the global field of output signals on the left that is shown on the right, and the “x0”, “y0” and “range” fields to set the initial coordinates and the length of the side of the square within which the output signals of the perceptron are calculated and displayed on the screen (Fig. 4.1).

To set the threshold and color for displaying signals that exceed and do not exceed the set threshold, fill in the appropriate fields in the “Color” section (Fig. 7.2).

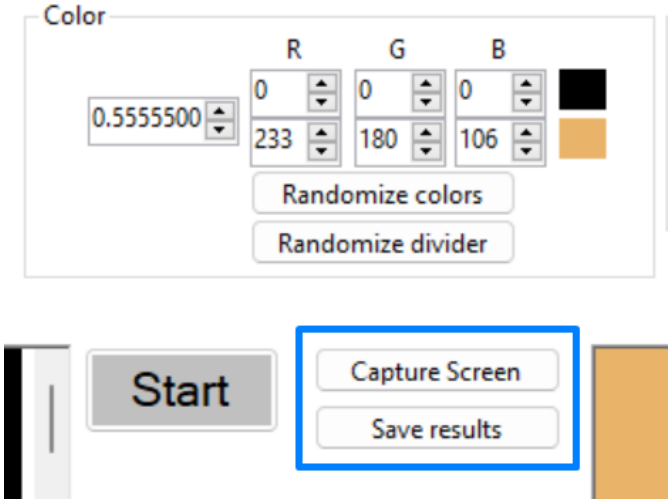


Fig 7.2. Fields for setting colors for displaying low and high signals and a button for saving parameter values and experiment results (highlighted in blue).

After setting all parameters and pressing the "Start" button, the program calculates the values of output signals for all possible combinations of input signals with the specified length in a specified area within the input signal space of the perceptron. The values of the output signals are calculated according to the algorithm described in Chapter 3.

Intensities of the input signals are represented as numbers $0, a_1a_2a_3a_4$, where a_1, a_2, a_3, a_4 are some digits from 0 to 9. In this way, it is possible to analyze all possible values that are transmitted to the inputs of the neural network. Since the neural network must work within a specific alphabet, for simplicity of data processing we will choose the alphabet $A = 0,1,2,3,4,5,6,7,8,9$.

The neural network sequentially processes each symbol of the input. This means that the neuron, which receives the signals $X = 0, x_1x_2x_3x_4$ and $Y = 0, y_1y_2y_3y_4$, processes the symbols x_1 and y_1 on the first iteration, then the symbols x_2 and y_2 are processed etc. Each connection of the perceptron has its own weight. Let the signal X pass through the connection with the weight w_1, and the signal Y – through the connection with the weight w_2, then the total intensity of the signals will be in the interval $D = [b, 9(w_1 + w_2) + b]$.

Since it is necessary to process the signal in such a way as to obtain a symbol from the alphabet A at the output, it is necessary to select the proper activation function. This activation function is the theta-function described in Chapter 3.

After processing all the symbols of the input signals in this way, a sequence of symbols of the output signal will be received. That is, after processing the corresponding symbols of the signals $X = 0, x_1x_2x_3x_4$ and $Y = 0, y_1y_2y_3y_4$, we will get the output signal $Z = 0, z_1z_2z_3z_4$. So, using this algorithm, for each of the possible values of the input signals, the intensities of the output signals are calculated and a matrix of the intensities of the resulting signals is created, where the columns correspond to the values of the first input signal X, and the rows correspond to the values of the second input signal Y. In the intersection of the corresponding column X_n and row Y_k, the value of the output signal $Z_{k,n}$ for the inputs X_n and Y_k is set:

$$\begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{k1} & Z_{k2} & \cdots & Z_{kn} \end{pmatrix} \quad (3)$$

Matrix 3 is square, that is $k = n$:

$$\begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{pmatrix} \quad (4)$$

After calculations are completed, the matrix (4) of the output values of the perceptron is shown in the form of a square image, in which each point stands for the intensity of the output signal for the corresponding inputs. The specified accuracy of calculations (number of decimal places) is used to construct the image. Each square of the given size is painted with one of the specified colors (Fig. 7.2), depending on whether the signal is greater or less than the set threshold. For example, if you set the word length to 3 and the threshold to 0.555, the output signals from 0.0 to 0.554 will have a color for low signals, and from 0.555 to 0.999 will have a color for high signals. The greater the specified accuracy of the calculations, the smaller the cell size, and therefore the greater their number in the figure.

After calculating the output signals, the software also calculates the number of squares of the corresponding color displayed on the image of the output field of the perceptron to determine the fractal dimension of this field.

In addition, in the results section, the program displays the values of the maximum and minimum signal in the field of output signals represented in the image, as well as the number of colored squares with high signals to verify the calculation of the fractal dimension.

REFERENCES

1. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943. https://doi.org/10.1007/BF02478259
2. F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. https://doi.org/10.1037/h0042519
3. F. Rosenblatt, *Principles of Neurodynamics*. Washington, DC: Spartan Books, 1962. https://doi.org/10.2307/1419730
4. Olazaran, Mikel, “A Sociological Study of the Official History of the Perceptrons Controversy,” *Social Studies of Science*, vol. 26, no. 3, p.611-659, 1996. https://doi.org/10.1177/030631296026003005
5. M. L. Minsky and S. A. Papert, Perceptrons. Cambridge, MA: MIT Press, 1969. https://doi.org/10.1016/s0361-9230(99)00182-3
6. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, April 1982. https://doi.org/10.1073%2Fpnas.79.8.2554
7. J. J. Hopfield, "Neural with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088-3092, May 1984. https://doi.org/10.1073/pnas.81.10.3088
8. J. J. Hopfield, "Learning algorithms and probability distributions in feed-forward and feed-back networks," *Proceedings of the National Academy of Sciences*, vol. 84, no. 23, pp. 8429-8433, December 1, 1987. https://doi.org/10.1073/pnas.84.23.8429
9. Freund, Y.; Schapire, R. E., "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277-296, 1999. https://doi.org/10.1023/A:1007662407062
10. Mehryar Mohri1, Afshin Rostamizadeh, “Stability Bound for Stationary Phi-mixing and Beta-mixing Processes,” *Journal of Machine Learning Research (JMLR)*, vol. 11, pp. 798-814, 2010. https://doi.org/10.48550/arXiv.0811.1629
11. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N. et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine*, IEEE, vol. 29, no. 6, pp. 82-97, 2012. http://dx.doi.org/10.1109/MSP.2012.2205597
12. Jurafsky D., Martin J.H., *Speech and language processing, 2nd edition*. NJ: Prentice Hall, 2008.
13. Shinde, B. S., Dani, A. R., "The origins of digital image processing and application areas in digital image processing medical images," *IOSR Journal of Engineering*, vol. 1, no. 1, pp. 066-071, 2012. http://dx.doi.org/10.9790/3021-0116671
14. He, K., Zhang, X., Ren, S., Sun, J., "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016. http://dx.doi.org/10.1109/CVPR.2016.90

15. Schmidhuber, J., "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, no. 3, pp. 85-117, 2015. http://dx.doi.org/10.1016/j.neunet.2014.09.003
16. Benoit B. Mandelbrot, *The Fractal Geometry of Nature*. San Francisco: W.H.Freeman and Co, 1982.
17. Richard M. Crownover, *Introduction to Fractals and Chaos*. University of Missouri-Columbia: Jones and Bartlett Publishers, 1995.
18. Takens, F., “Detecting strange attractors in turbulence,” *Dynamical Systems and Turbulence*, vol. 898, pp. 366-381, 1981. https://doi.org/10.1007/BFb0091924

Фрактальні властивості нейронних мереж

Новіков Артем Олександрович *аспірант кафедри штучного інтелекту та програмного забезпечення Харківський національний університет імені В.Н. Каразіна; майдан Свободи, 6, м. Харків, Україна, 61022*

Смирнов Вадим Максимович *студент магістратури кафедри штучного інтелекту та програмного забезпечення Харківський національний університет імені В.Н. Каразіна; майдан Свободи, 6, м. Харків, Україна, 61022*

Яновський Володимир Володимирович *доктор фізико-математичних наук; професор, Інститут монокристалів, Національна Академія Наук Україн; пр. Науки 60, м. Харків, Україна, 61001*

Роботу присвячено дослідженню властивостей нейронних мереж, які надзвичайно інтенсивно використовуються останнім часом у різноманітних прикладних напрямках. Вивчення їх загальних та фундаментальних властивостей набуває все більшої **актуальності** у зв’язку з широким застосуванням.
Метою роботи є вивчення поля реакції штучної нейронної мережі у просторі всіх можливих вхідних сигналів певної довжини. На прикладі простого персептрона досліджуються зони в яких поле реакцій нейронної мережі має структурно складний тип.
Методи дослідженн: Для дослідження поля вихідних сигналів було розроблено програмне забезпечення, яке дозволило моделювати та візуалізовати поле вихідних сигналів над простором всіх вхідних сигналів. Також програмне забезпечення дозволяло змінювати функцію активації, ваги та пороги кожного нейрону, що дозволило вивчити вплив усіх цих факторів на структурну складність поля вихідних сигналів.
В результаті було доведено, що, у випадку загального положення, у просторі вхідних сигналів існують зони тіні в яких поле реакції нейронної мережі має самоподібну фрактальну структуру. Визначено умови появи симетрії таких структур, досліджено вплив функцій активації, ваг та порогів нейронів мережі на властивості фрактальних структур. Виявлено, що вхідний шар нейронів на ці властивості впливає домінуючим чином. Отримані залежності фрактарної розмірності структур від ваг нейронів. Обговорено зміни, які відбуваються при зростанні розмірності простору вхідних сигналів.
Наявність зон тіні з фрактальним полем вихідних сигналів має важливе значення для розуміння функціонування штучних нейронних мереж. Такі зони тіні визначають області вхідного простору сигналів у яких реакцію нейронної мережі надзвичайно чутлива навіть до незначних змін вхідних сигналів. Це приводить до принципової зміни вихідних сигналів при незначній зміні вхідних сигналів.

Ключові слова: *штучна нейронна мережа, простір вхідних сигналів, поле вихідних сигналів, персептрон, штучний нейрон, фрактальні структури, фрактальна розмірність.*

Надійшла у першій редакції 12.08.2024, в останній - 19.09.2024.
The first version has been received on 12.08.2024, the final version - on 19.09.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-08

УДК (UDC) 004.7(075.8)

Руккас Кирило Маркович *д.т.н, доцент, професор закладу вищої освіти кафедри теоретичної та прикладної інформатики*
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: rukkas@karazin.ua
https://orcid.org/0000-0002-7614-0793

Шкловський Валерій Борисович *здобувач другого (магістерського) рівня вищої освіти за спеціальністю 122 Комп'ютерні науки, освітньо-професійної програми "Інформатика" кафедри теоретичної та прикладної інформатики*
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail:xa11800826@student.karazin.ua
https://orcid.org/0009-0006-3223-8680

Морозова Анастасія Геннадіївна *к.т.н., доцент закладу вищої освіти кафедри теоретичної та прикладної інформатики*
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: a.morozova@karazin.ua
https://orcid.org/0000-0003-2143-7992

Кузнєцова Вікторія Олександрівна *к.ф-м.н., доцент закладу вищої освіти кафедри вищої математики та інформатики*
Харківський національний університет імені В .Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022
e-mail: vkuznietcova@karazin.ua
https://orcid.org/0000-0003-3882-1333

Керування трафіком реального часу в комп'ютерних мережах

У роботі розглянуто задачу моделювання критичних за часом мереж з метою дослідження ефективності різних методів управління трафіком реального часу.
Актуальність. Моделювання мереж реального часу є активним напрямком досліджень в галузі комп'ютерних наук та інформаційної технології. Дослідження в цій галузі спрямовані на розробку методів та алгоритмів, які дозволяють забезпечувати ефективну передачу даних у реальному часі. Робота присвячена моделюванню чутливих за часом мереж, використовуючи різні методи керування трафіком реального часу.
Мета. Метою роботи було проаналізувати існуючи стандарти мереж реального часу, розробити імітаційну модель мережі реального часу та дослідити ефективність різних методів управління трафіком реального часу.
Методи дослідження. Методи дослідження базуються на сучасних теоріях математичного моделювання, імітаційного моделювання, телекомунікаційних стандартах, а саме Time-Sensitive Networking (TSN).
Результати. Було ретельно проаналізовано існуючі стандарти мереж реального часу. Детальну увагу було приділено технології TSN (Time Sensitive Network), досліджена проблема досягнення синхронізації в таких мережах, досліджені різні методи формування трафіку в мережі і перевірені отримані аналітичні результати на відповідність специфікації TSN мережі. Для моделювання було використано програмне забезпечення OMNeT++. Для кожного методу формування трафіку були побудовані графіки наскрізної затримки в різних умовах, що специфічно для мереж реального часу.
Висновки.В роботі було розглянуто мережі стандарту TSN, які дозволяє забезпечувати надійний зв'язок в режимі реального часу та підвищують універсальність в промислових мережах. Для TSN була досліджена проблема досягнення синхронізації пристроїв в мережі, розглянуті різні методи планування трафіку, встановлена важливість використання технології наскрізної комутації, розроблена імітаційна модель мережі TSN, реалізована функціональність такої мережі для планування трафіку.

Ключові слова:*.комп'ютерні мережі, трафік, мережі реального часу, TSN, наскрізна затримка, мережевий протокол, Ethernet, наскрізна комутація, вузол мережі, gPTP, UDP, комутатор, синхронізація часу.*

Як цитувати: Руккас К. М., Шкловський В. Б., Морозова А. Г., Кузнєцова В. О. Керування трафіком реального часу в комп'ютерних мережах. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології.*

Автоматизовані системи управління. 2024. вип. 64. С.79-89. https://doi.org/10.26565/2304-6201-2024-64-08

How to quote: K. Rukkas, V. Shklovskyi, A. Morozova, V. Kuznietcova, "Real-time traffic management in computer networks", *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 79-89, 2024. https://doi.org/10.26565/2304-6201-2024-64-08[in Ukrainian]

1 Вступ

Сьогодні в часи глобальної індустріалізації і технологічного прориву важко уявити життя без розумного будинку, автомобіля, який самостійно безпечно може їхати і супутника на орбіті, який аналізує стан атмосфери в реальному часі. Для якісної роботи всіх вузлів недостатньо великої швидкості передачі даних, велику роль відіграє надійність мережі і каналів зв'язку. Стандартне мережеве обладнання [1] сучасних інформаційних технологій не може забезпечити автоматичну синхронізацію елементів, але крок за кроком інтеграція нових технологій робить індустрію інформаційних мереж більш високо динамічною, гнучкою і передбачуваною. Однією з таких технологій є TSN (Time Sensitive Network) [2], яка забезпечує надійний зв'язок в реальному часі та підвищує універсальність в промислових мережах. Такі мережі жорстко налаштовані під виконання специфічних задач, тому для перевірки нових функцій в реальному часі неможливе і постає задача створення симуляції у віртуальному середовищі для демонстрації своїх переваг і недоліків. Це дозволяє провести дослідження в контрольованих умовах без небезпеки втручання в реальне середовище.

2 Постановка задачі. Огляд відомих результатів в області дослідження.

Моделювання мереж реального часу є активним напрямком досліджень в галузі комп'ютерних наук та інформаційної технології. Дослідження в цій галузі спрямовані на розробку методів та алгоритмів, які дозволяють забезпечувати передачу даних у реальному часі, а також використання удосконалених методів збору інформації для покращення продуктивності мережевих систем [3]. Одним з відомих результатів моделювання мереж реального часу є мережевий стек протоколів Ethernet AVB (Audio Video Bridging) [4], який було розроблено з метою забезпечення передачі аудіо- та відеоданих у реальному часі з низькими затримками та високою якістю обслуговування. Іншим відомим результатом є модель передачі даних TTEthernet (Time-Triggered Ethernet), яка базується на принципах строгого часового планування та дозволяє передавати дані з високою гарантією надійності та точності у реальному часі. Однак обидві технології є обмеженими за сферою використання і не мають інструментів для передачі універсальних даних.

Задачею цієї роботи полягала в моделюванні критичних за часом мереж з метою дослідження ефективності різних методів управління трафіком реального часу. Виходячи з існуючих стандартів TSN було визначено, що наступні мережі повинні відповідати певним критеріям, а саме 1) доставка даних з низькою затримкою та мінімальним відхиленням від заданого графіка; 2) розділення трафіку на групи з різними пріоритетами; 3) синхронізація часу в мережі; 4) захист від мережевих колізій та переповнень.

Дана задача пов'язана з наступними науковими та технічними завданями:

1. Дослідження проблеми досягнення глобального часу в мережі.
2. Огляд різних методів планування трафіку, а саме формування з урахуванням часу і формування на основі кредитів.
3. Визначення необхідності призупинення передачі кадрів в TSN мережі.
4. Встановлення важливості технології наскрізної комутації.
5. Розробка імітаційної моделі TSN мережі реального часу методом модульного проектування і реалізація функціональності TSN для планування трафіку.
6. Аналіз отриманих аналітичних результатів на відповідність специфікації TSN мережі.

3 Виклад основного матеріалу з обґрунтуванням отриманих наукових та практичних результатів

3.1. Мережі реального часу

Комп'ютерна мережа, яка може гарантувати відповідь протягом визначеного проміжку часу, так званого «дедлайну», можна класифікувати як мережу реального часу [5]. Існує два важливих

фактора для того, щоб система відповідала своїм термінам доставки даних: детермінізм і обмежена затримка для зв'язку. Детермінізм означає, що система в будь-якому випадку буде формувати однаковий результат для однакових вхідних даних. Обмежена затримка означає, що існує верхня межа, яку гарантовано не буде перевищено. Проте стандартний Ethernet протокол не може гарантувати ні детермінізм, ні обмежену за часом затримку. У випадку, коли декілька звичайних пристроїв Ethernet надсилають серію пакетів на один і той самий порт приймача, може виникнути непередбачена затримка, оскільки пакети стануть в чергу, або навіть втрату даних, якщо буфер черги не достатньо великий. З метою уникнення даних проблем були розроблені спеціальні стандарти для забезпечення жорстких гарантій для мереж реального часу, які використовують Ethernet у формі обмеженої затримки або гарантії пропускної здатності. Кожен з них має свої особливості і використовується в різних галузях. Основні стандарти мереж реального часу відображено нижче (табл. 1) [6].

Таблиця 1. Основні стандарти мереж реального часу

Стандарт	Сфера використання
Ethernet/IP, OPC UA, PROFINET	Промислові системи автоматизації
Profibus	Автоматизовані системи
Modbus, HART	Збір даних з пристроїв, що забезпечують контроль та управління процесами, включаючи вимірювальні прилади та сенсори
CAN	Передача даних між мікроконтролерами в автомобільних системах
EtherCAT	Передача даних в режимі реального часу між промисловими пристроями, що використовує технологію «Master-Slave»
TSN	Забезпечення точності часу та передачі даних з обмеженою затримкою та гарантованою доставкою

Серед вище зазначених стандартів для подальшого дослідження був обраний стандарт TSN. Даний стандарт використовується як в промислових мережах (автоматизовані виробничі лінії, машини з ЧПУ, медичне обладнання), так і в телекомунікаційних мережах (передача відео і аудіо), де вкрай важливо передавати дані з високою точністю і гарантованою доставкою.

3.2. Time Sensitive Networking (TSN)

Time Sensitive Networking (TSN) [7] – це визначена стандартом IEEE 802.1Q технологія для гарантування детермінованого обміну пакетами в Ethernet. Технологія TSN передбачає центральне управління, забезпечує гарантії доставки даних та мінімізовану затримку за допомогою планування часу. TSN є технологією другого рівня мережевої моделі OSI, таким чином дана технологія – це стандарт Ethernet, а не стандарт інтернет протоколу. Рішення про відправку пакетів приймають мости TSN і використовують вміст заголовка Ethernet, а не IP-адресу. Корисне навантаження кадрів Ethernet може бути будь-яким і ніяким чином не обмежується інтернет протоколом. Завдяки цьому технологія TSN може використовуватися в будь-якому середовищі для переноса будь-якого корисного навантаження промислового застосування.

Основним поняттям технології TSN є формування трафіку (рис. 1). Формувальник трафіку – це механізм, який спрямований на підвищення якості обслуговування для одного або декількох типів трафіків у мережі. На рис. 1 продемонстровано використання комутації «зберегти і переслати» проти наскрізної комутації для надсилання одного пакету даних від вузла A до вузла B через два мости Ethernet.

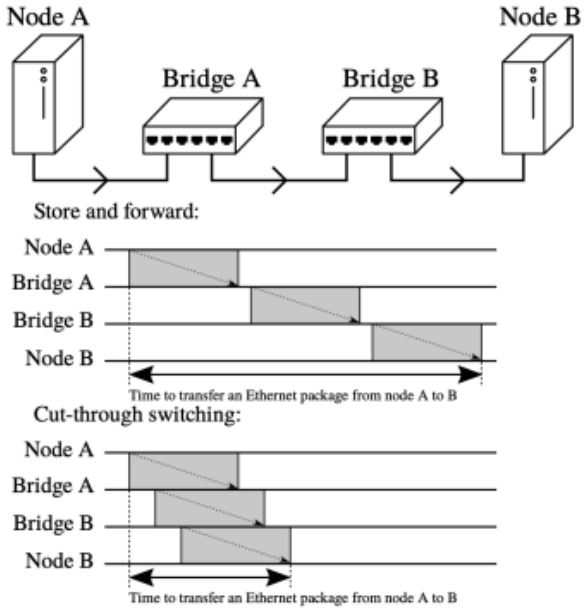


Рис. 1. Формування трафіку в TSN мережах.

Технологія TSN складається з набору стандартів, розроблених для підтримки вимог реального часу та гарантованої якості обслуговування [8].

3.3. Синхронізація часу

Станом на сьогодні реальний світ не має глобально узгодженого часу і кожен вузол в мережі використовує свій власний фізичний годинник для вимірювання проміжків часу. Через те, що кожний годинник має свою похибку, час на годинниках на різних вузлах мережі з часом може відрізнятися. Для вирішення цієї проблеми необхідно використовувати методи синхронізації і періодично коригувати час на всіх вузлах мережі для коректної і злагодженої роботи. Синхронізація часу надважлива в TSN мережах, тому що точне відстеження має вирішальне значення в цих мережах. Для TSN мереж необхідна реалізація протоколу gPTP, який є розширенням стандарту IEEE 802.1 AS [9], який визначає протокол PTP або також відомий як IEEE 1588, всіма вузлами для досягнення спільної концепції часу. Протокол gPTP (рис. 3) використовує ієрархічну архітектуру, в якій є один або декілька головних вузлів і багато залежних. Коригування часу відбувається шляхом надсилання повідомлень синхронізації від головного вузла до всіх підлеглих.

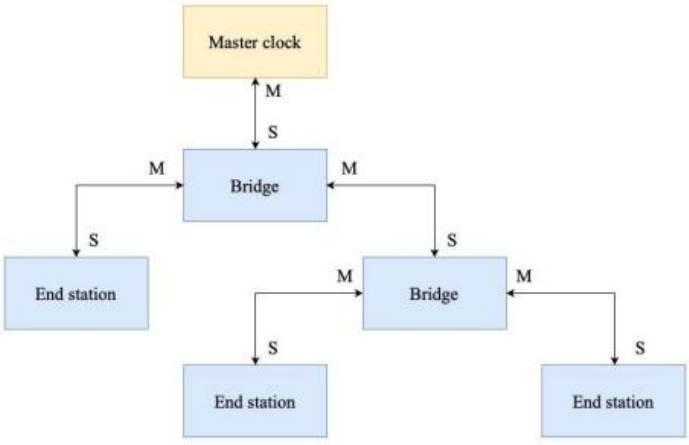


Рис. 2. Приклад мережі, побудованої згідно протоколу gPTP, де M – Ethernet порт має роль головного, а S – підлеглого.

Для демонстрації необхідності синхронізації годинників була змодельована наступна комп'ютерна мережа (рис. 4), яка складається з двох хостів-джерел, які надсилають UDP дані до двох хостів-приймачів через два поєднаних між собою комутатора.

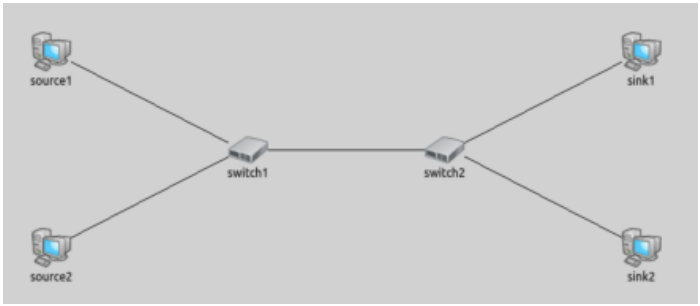


Рис. 3. Топологія мережі для виявлення проблеми глобального часу

В даній мережі всі вузли мають випадкову швидкість дрейфу годинників, яка періодично змінюється і всі годинники синхронізуються з часом модуля switch 1 (рис. 4).

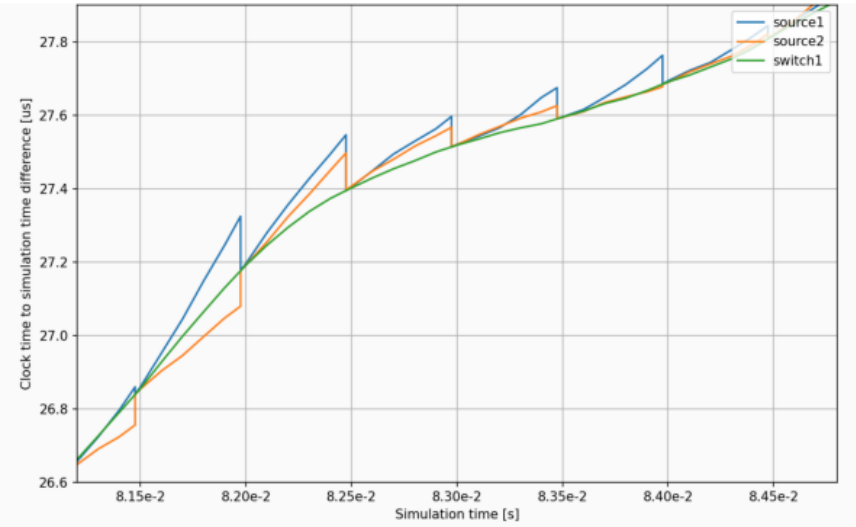


Рис. 4. Дрейф годинників з періодичною синхронізацією

Також була виміряна наскрізна затримка пакетів на шляху від хоста-джерела до відповідного хоста-приймача у різних випадках.

3.4. Планування та формування трафіку

Планування і формування трафіку грають важливу роль в TSN мережах, оскільки саме так гарантується, що критичний трафік буде мати вищий пріоритет над іншим трафіком і загальна продуктивність мережі буде оптимізована відповідно до вимог якості обслуговування різних потоків трафіку. Було розглянуто два важливих метода формування трафіку:

1. Формування з урахуванням часу (Time-Aware Shaping).
2. Формування на основі кредитів (Credit-Based Shaping)ю

Формування з урахуванням часу

Стандарт IEEE 802.1Qbv [10] визначає спосіб передачи TSN кадрів за розкладом, дозволяючи іншим Ethernet кадрам передаватися в інтервалах, відповідно пріоритетності даних. Це забезпечує можливість гарантувати обмежену затримку передачі критичних за часом кадрів, що важливо для TSN мереж.

Для демонстрації необхідності формування трафіку з урахуванням часу в TSN мережах була змодельована наступна мережа (рис. 5). Вона складається з наступних модулів: клієнт і сервер є

модулями TSNDevice, TrafficGenerator і TrafficReceiver – StandardHost, а комутатор – TSNSwitch. Клієнт створює високопріоритетні дані, а генератор трафіку – фонові дані з низьким пріоритетом. Пакети надсилаються клієнтом і генератором трафіку у випадкові проміжки часу, але комутатором у мережі вони пересилаються тільки згідно встановленого графіку відкриття дверей.

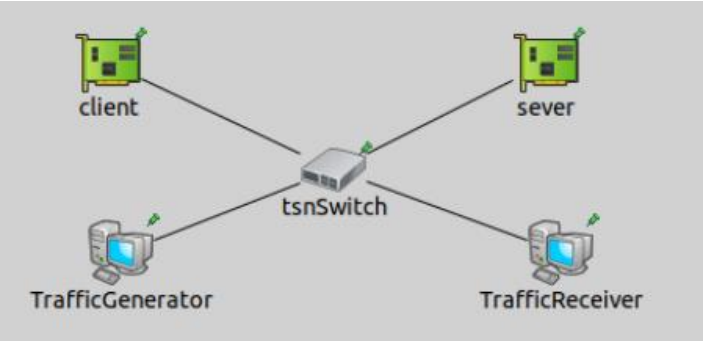


Рис.5. Мережа для демонстрації формування трафіку з урахуванням часу

На діаграмі нижче (рис. 6) продемонстровано результат роботи формувача трафіку з урахуванням часу – обмеження швидкості передачі кадрів зверху до заданих значень, а отже і обмеження передачі трафіку. Пунктирною лінією зображена швидкість до роботи формувача, суцільною – після.

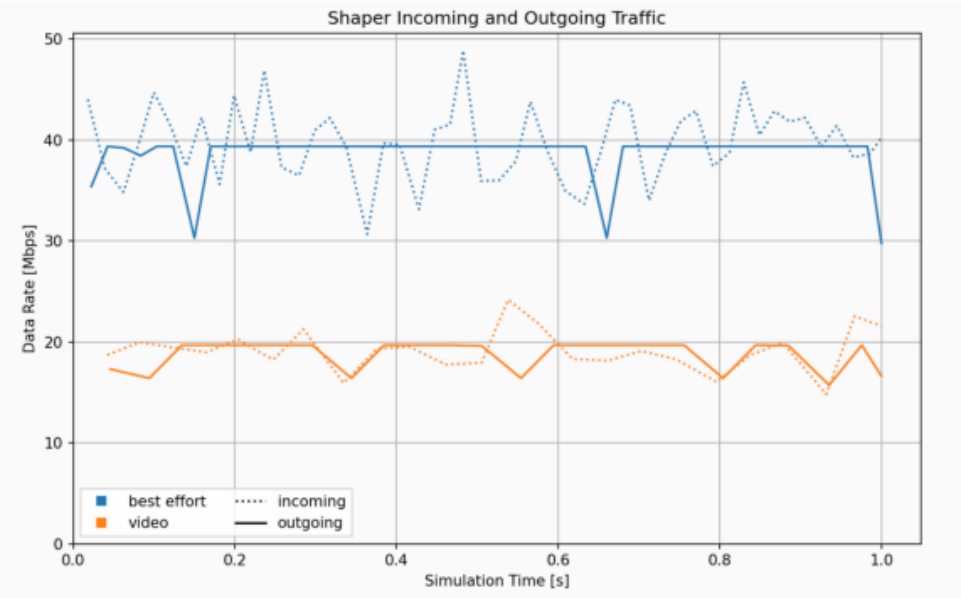


Рис.6. Формування трафіку з урахуванням часу

Формування на основі кредитів

Формування трафіку на основі кредитів досягається за допомогою алгоритма регулювання потоку трафіку в мережі. Ідея алгоритму полягає в тому, що кожному потоку трафіку призначається певна кількість кредитів, за допомогою яких далі визначається скільки даних можна передати за певний проміжок часу.

Для демонстрації роботи формувальника трафіку була використана таж сама мережа, що і для формування трафіку з урахуванням часу (рис. 5). На діаграмі нижче (рис. 7) продемонстровано результат роботи формувача трафіку з урахуванням часу – обмеження швидкості передачі кадрів зверху до заданих значень, а отже і обмеження передачі трафіку. Пунктирною лінією зображена швидкість до роботи формувача, суцільною – після.

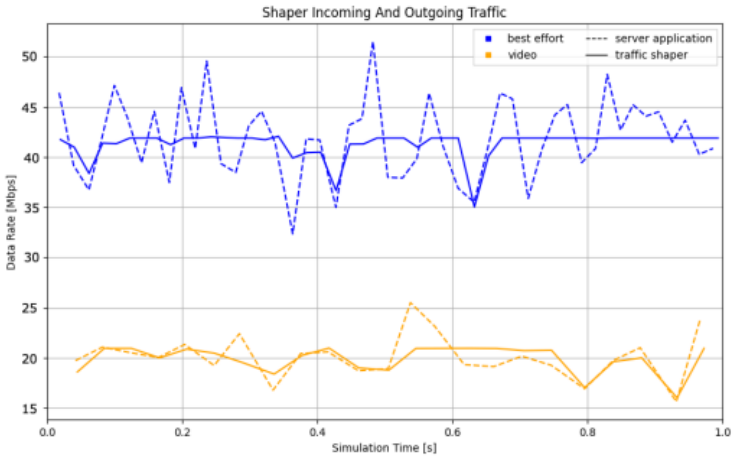


Рис.7. Формування трафіку на основі кредитів

3.5. Призупинення передачі кадрів

Технологія призупинення передачі кадрів (Frame Preemption) [11], яка визначена в стандарті 802.1Qbu [6], дає можливість мережі визначити деякі класи як пріоритетні, а весь інший трафік, як той, передачу якого можна призупинити. Дана технологія разом з фрагментацією дає можливість мосту призупинити передачу Ethernet кадру в тому випадку, якщо необхідно відправити пакет вищого пріоритету. Дана технологія дуже важлива для TSN мереж, тому що кадри, які мають вищий пріоритет, можуть містити критичні до часу дані, які необхідно доставити з мінімальною затримкою.

Для демонстрації функції призупинення передачі кадрів змодельована мережа, яка складається з двох модулів StandardHost в трьох конфігураціях:

1) черга FIFO – базова конфігурація, яка не використовує пріоритетну чергу або функцію призупинення передачі кадрів;

2) пріоритетна черга – конфігурація, яка використовує чергу пріоритетів для зменшення затримки високо пріоритетних кадрів;

3) функція призупинення передачі кадрів – використовує призупинення для високо пріоритетних кадрів для низької затримки із забезпеченням верхньої межі.

Для кожної з конфігурацій було визначено наскрізна затримка для пакетів однакової довжини, яка дорівнює сумі часу передачі кадру, затримці в черзі та міжкадровому проміжку.

У випадку реального трафіку на затримку фонових пакетів не впливає ні використання черги пріоритетів, ні функція призупинення передачі кадрів. Затримка високопріоритетних кадрів значно зменшується за рахунок того, що трафік сильно відрізняється за довжиною пакетів.

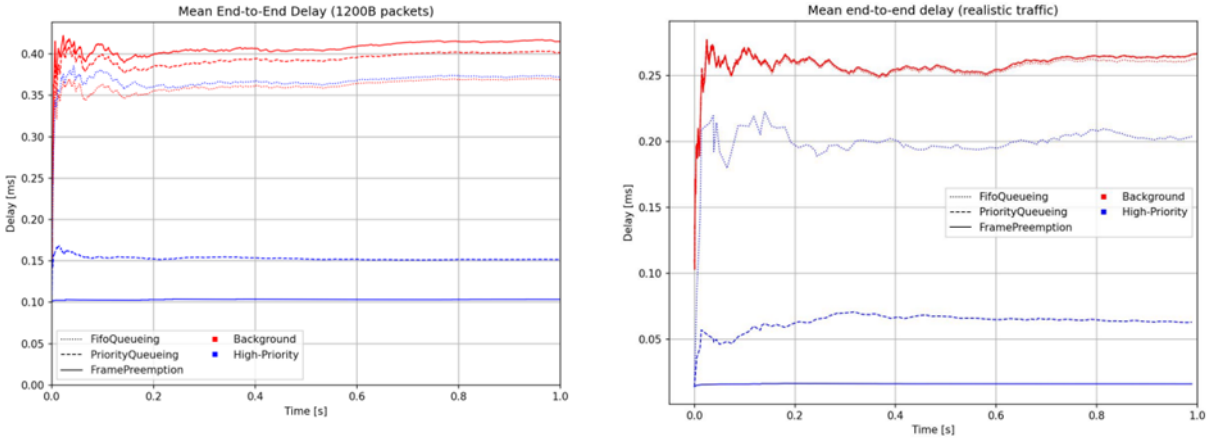


Рис.8. Графік наскрізної затримки на відрізку [0, 1]
ліворуч – для пакетів довжиною 1200Б;праворуч – для трафіку реального часу

3.6. Наскрізна комутація

Наскрізна комутація – це технологія, яка використовується в системах комутації пакетів для пересилання кадрів в мережі [12]. Вона, на відміну від комутації зі зберіганням і пересиланням кадру тільки після завершення отримання, передбачає початок процесу пересилання до того, як буде отримано весь кадр. До переваг цієї технології можно віднести зменшення наскрізної затримки пересилання кадрів, оскільки комутатор починає відправку до повного отримання. Недоліком є вища частота помилок, якщо порівнювати з пересиланням кадрів після завершення отримання його, тому що дана технологія не передбачає перевірку кадру на наявність помилок перед початком пересилання.

Для аналізу даної технологія була змодельована мережа, яка складається з чотирьох модулів: клієнт і сервер є модулями TSNDevice, а switch1 і switch2 – TSNSwitch. Для моделювання в мережі був визначений наступний трафік: джерело – клієнт; пункт призначення – сервер; тип трафіку – video; інтервал передачі – 100 мс; розмір корисного навантаження – 1200.

Детально проаналізовані і винесені в діаграму (рис. 9) результати моделювання для двох випадків – пересилання кадру після завершення отримання його і технологія наскрізною комутації.

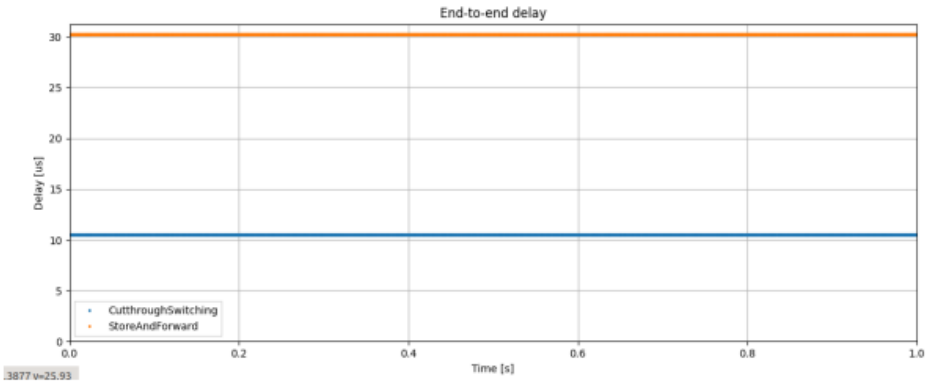


Рис.9. Графік наскрізної затримки на відрізку [0, 1]

3.7. Комбіновані функції в мережі реального часу

Для демонстрації комбінованих функцій TSN мережі змодельована наступна топологія (рис. 10). Вона складається з двох комутаторів (TSN Switch), які з'єднані між собою, чотирьох хостів-джерел (TSN Talker), які надсилають запланований трафік чотирьом хостам приймачам (TSN Listener), одного генератора трафіку (Traffic Generator), одного приймача трафіку (Traffic Receiver) й головного годинника мережі (MasterClock). Товстими лініями позначені канали з пропускною спроможністю в 1 Гбіт/с, а тонкими – 100 Мбіт/с.

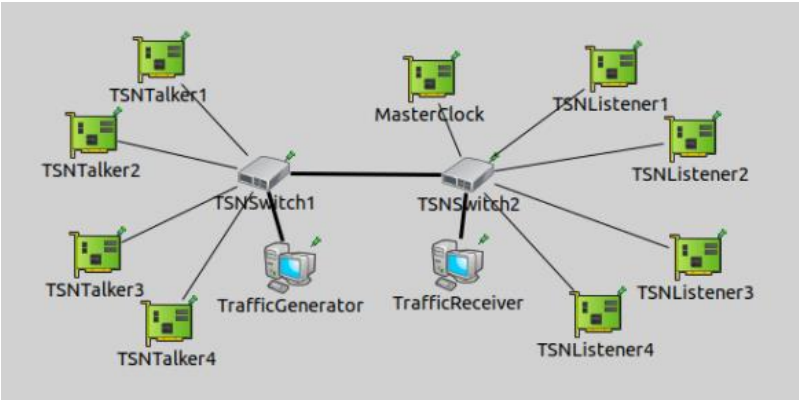


Рис. 10. Топологія моделювання TSN мережі

За допомогою програмного пакету CNC Cisco визначений розклад керування інтервалами для передачі запланованого трафіку. З метою ізолювання запланованого трафіку від іншого і гарантування достатньої пропускносної здатності стандартом IEEE802.1Qbv [9] вводиться захисна смуга (GB). Всі інші часові інтервали, за винятком інтервалів для запланованого трафіку

і трафіку захисної смуги, призначені для передачі фонового трафіку. Для кожного потоку трафіку між хостом-джерелом і хостом-слухачем була розрахована наскрізна затримка за формулою (3.1)

$$\text{затримка} = T_{\text{destination}} - T_{\text{source}} \quad (3.1)$$

де $T_{\text{destination}}$ і T_{source} – це часові мітки, отримані при виході з джерела і вході в слухача відповідно. Вимірювання затримки було проведено на 1000 пакетах запланованого трафіку в найгіршому випадку, тобто інтервал незапланованого трафіку дорівнював 1 мкс, результат був усереднений і наведений у таблиці 2.

Таблиця 2. Середнє значення затримки для запланованого трафіку

	Trafic 1	Trafic 2	Trafic 3	Trafic 4
Середнє значення затримки	569	469	338	319

Окрім ситуації з найгіршим випадком, був проведений ряд випробувань, в яких змінювалась кількість незапланованого трафіку. Аналогічно попередньому моделюванню значення наскрізної затримки було проаналізовано відносно 1000 пакетів запланованого трафіку. На основі отриманих даних побудований лінійний графік (рис. 11), де вісь X представляє інтервал передачі незапланованого трафіку, вісь Y відображає затримку трафіку.

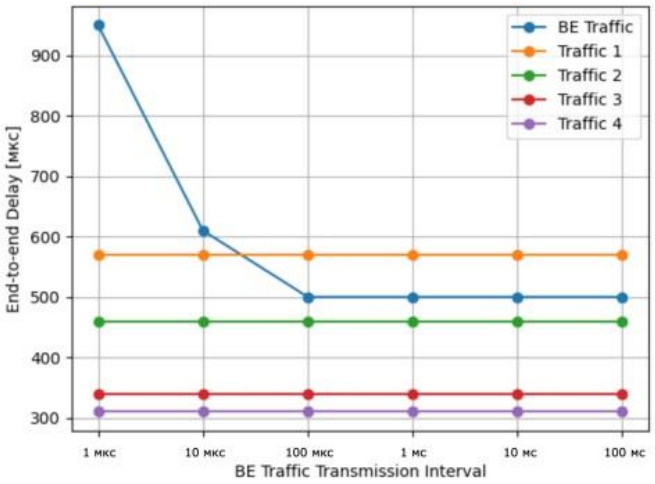


Рис. 11. Наскрізна затримка для запланованого і фонового трафіку

У випадку, коли інтервал надсилання незапланованого трафіку перевищує відмітку в 100 мкс, затримка його має постійне значення, тому що канал для передачі даних здебільшого простоює. Якщо збільшити обсяг фонового трафіку, то бачимо, що наскрізна затримка зростає майже з експоненціальною щвидкістю. Однак затримка запланованого трафіку залишається постійною в будь-якому випадку. Навіть в найгіршому випадку, коли канал повністю завантажений пакетами незапланованого трафіку, розроблена функціональність здатна належним чином планувати трафік і задовольняти вимоги детермінності мережі.

4 Conclusions

В роботі було детально вивчено існуючі стандарти мереж реального часу. Для подальшого аналізу був обраний стандарт TSN, який дозволяє забезпечувати надійний зв’язок в режимі реального часу та підвищує універсальність в промислових мережах. Окрім цього було досягнуто наступних результатів: 1) досліджена проблема досягнення глобального часу в мережі; 2) розглянуті різні методи планування трафіку, а саме формування з урахуванням часу і

формування на основі кредитів; 3) визначена необхідність призупинення передачі кадрів в TSN мережі; 4) встановлена важливість використання технології наскрізної комутації; 5) розроблена імітаційна модель TSN мережі реального часу методом модульного проектування і реалізована функціональність TSN для планування трафіку; 6) проаналізовані отримані аналітичні результати на відповідність специфікації TSN мережі. Розроблена імітаційна модель відповідає основним властивостям TSN мережі, а саме: детермінованість передачі даних; гарантована пропускна здатність; резервування пропускної здатності; забезпечення низької затримки передачі даних; функціонування в реальному часі. В процесі моделювання було визначено розподіл часових інтервалів для всього трафіку на основі розкладу управління воротами пересилання трафіку в мережі. У результаті симуляції даної мережі, було отримано досить точну і детальну характеристику її функціональності, а саме проаналізована наскрізна затримка для різних типів трафіку, що властиво для мережі реального часу. Представлені імітаційні модулі точно відповідають специфікаціям TSN для детермінованого зв'язку, а розроблена модель є ефективним інструментом верифікації для подальшого дослідження фізичної моделі мережі.

REFERENCES

1. "IEEE Standard for Ethernet - Corrigendum 1: Multi-lane Timestamping," in *Corrigendum to IEEE Std 802.3-2015 as amended by IEEE Std 802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, IEEE Std 802.3bp-2016, IEEE Std 802.3br-2016, IEEE Std 802.3bn-2016, IEEE Std 802.3bz-2016, IEEE Std 802.3bu-2016, IEEE Std 802.3bv-2017* , vol., no., pp.1-16, 21 April 2017, https://doi.org/10.1109/IEEESTD.2017.7907155.
2. ISO/IEC/IEEE International Standard - Telecommunications and information exchange between information technology systems - Requirements for local and metropolitan area networks - Part 1CM: Time-sensitive networking for fronthaul," in *ISO/IEC/IEEE 8802-1CM:2019(E)* , vol., no., pp.1-66, 23 Aug. 2019, https://doi.org/10.1109/IEEESTD.2019.8811785.
3. Rukkas, K., Morozova, A., Uzlov, D., Kuznietcova, V., & Chumachenko, D. (2024). Optimizing information support technology for network control: a probabilistic-time graph approach. *Radioelectronic and Computer Systems, 2024*(2), 85-97. https://doi.org/10.32620/reks.2024.2.08
4. 802.1AS - Timing and Synchronization. [Електронний ресурс]. Доступно: https://www.ieee802.org/1/pages/802.1as.html
5. H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, 2nd ed., ser. Real-Time Systems Series, 2011.
6. Lee, Kyung-Chang & Lee, Suk. Performance evaluation of switched Ethernet for real-time industrial communications. Computer Standards & Interfaces. 24. 411-423. https://doi.org/10.1016/S0920-5489(02)00070-3.
7. Finn, Norman. (2018). Introduction to Time-Sensitive Networking. IEEE Communications Standards Magazine. 2. 22-28. https://doi.org/10.1109/MCOMSTD.2018.1700076.
8. Pruski, A., Ojewale, M. A., Gavrilut, V., Yomsi, P. M., Berger, M. S., & Almeida, L. (2021). Implementation Cost Comparison of TSN Traffic Control Mechanisms. In *Proceedings of 26th IEEE International Conference on Emerging Technologies and Factory Automation* IEEE. https://doi.org/10.1109/ETFA45728.2021.9613463
9. P802.1AS-Rev – Timing and Synchronization for Time Sensitive Applications," IEEE 802.1. [Електронний ресурс]. Доступно: https://1.ieee802.org/tsn/802-1as-rev/
10. "IEEE Standard for Local and Metropolitan Area Network--Bridges and Bridged Networks," in *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)* , vol., no., pp.1-1993, 6 July 2018, https://doi.org/10.1109/IEEESTD.2018.8403927.
11. TSN Technology: Ethernet Frame Preemption, Part 1. Part 2. Industrial ethernet book. Technology. [Електронний ресурс]. Доступно: https://iebmedia.com/technology/tsn/tsn-technology-ethernet-frame-preemption/
12. J. Jiang, Y. Li, S. H. Hong, M. Yu, A. Xu and M. Wei, "A Simulation Model for Time-sensitive Networking (TSN) with Experimental Validation," *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019, pp. 153-160, https://doi.org/10.1109/ETFA.2019.8869206

13. V. Gavriluţ, A. Pruski, and M. S. Berger, “Constructive or optimized: An overview of strategies to design networks for time-critical applications,” ACM Computing Surveys, vol. 55, no. 3, Feb. 2022. https://doi.org/10.1145/3501294
14. OMNeT++ Discrete Event Simulator, 2022. [Електронний ресурс]. Доступно: https://omnetpp.org/.

Rukkas Kyrylo *DSc, Associate Professor, Full Professor, Department of Theoretical and Applied Computer Science*
V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022
e-mail: rukkas@karazin.ua;
https://orcid.org/0000-0002-7614-0793

Shklovskyi Valerii *Master’s student, Department of Theoretical and Applied Computer Science*
V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022
e-mail: xa11800826@student.karazin.ua
https://orcid.org/0009-0006-3223-8680

Morozova Anastasiia *PhD, Associate Professor, Department of Theoretical and Applied Computer Science*
V.N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022
e-mail: a.morozova@karazin.ua;
https://orcid.org/0000-0003-2143-7992

Kuznietcova Victoriya *Doctor of Philosophy, Associate professor, Associate professor of higher mathematics and computer sciences department, V. N. Karazin Kharkiv National University, Svobody Sq., 4, Kharkiv, Ukraine, 61022*
e-mail: vkuznietcova@karazin.ua
https://orcid.org/0000-0003-3882-1333

Real-time traffic management in computer networks

The paper considers the problem of modeling time-critical networks in order to study the effectiveness of various real-time traffic management methods.
Relevance. Real-time network modeling is an active area of research in computer science and information technology. Research in this area is aimed at developing methods and algorithms for efficient real-time data transmission. The work is devoted to modeling time-sensitive networks using various real-time traffic management methods.
Goal. The purpose of the work was to analyze existing real-time network standards, develop a simulation model of a real-time network, and investigate the effectiveness of various real-time traffic management methods.
Research methods. The research methods are based on modern theories of mathematical modeling, simulation modeling, and telecommunications standards, namely Time-Sensitive Networking (TSN).
The results. The standards of real-time networks were thoroughly investigated. Detailed attention was paid to TSN (Time Sensitive Network) technology, the problem of achieving synchronization in such networks and various methods of traffic generation were investigated; the obtained analytical results were checked for compliance with the TSN network specification. OMNeT++ software was used for modeling real-time networks. End-to-end delay graphs were constructed for each traffic generation method under different to conditions of real-time networks.
Conclusions. The paper considered TSN standard networks for reliable real-time communication and increase versatility in industrial networks. The problem of achieving synchronization of devices in the TSN network was investigated, various methods of traffic planning were considered, the importance of using end-to-end switching technology was established, a simulation model of the TSN network was developed.

Keywords: *computer networks, traffic, real-time networks, TSN, end-to-end delay, network protocol, Ethernet, end-to-end switching, network node, gPTP, UDP, switch, time synchronization.*

Надійшла у першій редакції 21.09.2024, в останній - 29.10.2024.
The first version has been received on 21.09.2024, the final version - on 29.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-09

УДК (UDC) 539.3+519.6

Rusanov Roman *PhD, senior researcher, Department of Thermogasdynamics of Power Machines, Anatolii Pidhornyi institute of power machines and systems of the National academy of sciences of Ukraine, Komunalnykiv st., 2/10, Kharkiv, Ukraine, 61046 e-mail:roman_rusanov@ipmach.kharkov.ua;* **https://orcid.org/0000-0003-2930-2574**

Rusanov Andrii *Academician NASU, director of Anatolii Pidhornyi institute of power machines and systems of the National academy of sciences of Ukraine, Komunalnykiv st., 2/10, Kharkiv, Ukraine, 61046 e-mail:rusanov@ipmach.kharkov.ua;* **https://orcid.org/0000-0002-9957-8974**

Kriutchenko Denys *PhD, researcher, Department of Thermogasdynamics of Power Machines, Anatolii Pidhornyi institute of power machines and systems of the National academy of sciences of Ukraine, Komunalnykiv st., 2/10, Kharkiv, Ukraine, 61046 e-mail: wollydenis@gmail.com;* **https://orcid.org/0000-0002-6804-6991**

Bykov Yurii *CTS, senior researcher, Department of Thermogasdynamics of Power Machines, Anatolii Pidhornyi institute of power machines and systems of the National academy of sciences of Ukraine, Komunalnykiv st., 2/10, Kharkiv, Ukraine, 61046 e-mail: yubykoff@gmail.com;* **https://orcid.org/0000-0001-7089-8993**

Degtyarev Kirill *CTS, senior researcher, Department of Thermogasdynamics of Power Machines, Anatolii Pidhornyi institute of power machines and systems of the National academy of sciences of Ukraine, Komunalnykiv st., 2/10, Kharkiv, Ukraine, 61046 e-mail: kdegt89@gmail.com;* **https://orcid.org/0000-0002-4486-2468**

Computer modeling temperature and strength characteristics of an ultra-supercritical loop-type steam turbine rotor

Relevance. A global trends have emerged involving the continuous increase in demands for efficiency and reliability of power equipment, particularly ultra-supercritical steam turbines. Such turbines operate under challenging conditions of high temperatures and pressures, which can lead to significant thermomechanical stresses in rotor. Computer modeling is ideally suited to solving such problems. Therefore, research dedicated to strength calculations and determining the temperature characteristics of blades in high- and medium-pressure cylinders is highly relevant. These studies enable engineers to ensure the strength and durability of components in steam turbines.

Objective. To perform strength and temperature distribution calculations for the blades of high- and medium-pressure cylinders in a loop-type steam turbine with ultra-supercritical initial steam parameters.

Results. Data on the temperature field distribution in rotor blades of high- and medium-pressure cylinders were obtained. Using the results of the temperature calculations, the strength of the first rotor blades in the high-pressure cylinder was assessed under the influence of uneven temperature distribution and rotor rotation. Cooling of the first-stage turbine blades is achieved through convective heat exchange from the flow of cold steam from the last stage to the internal channels of the blades.

Conclusions. The results demonstrate the efficiency of the selected blade cooling method and the level of maximum stresses within the blades. One of the notable features of the operating conditions for loop-type turbine blades is the uneven heating that occurs both during transient and steady-state operating modes. Uneven heating leads to the formation of thermal stresses in the blades, which negatively affects their lifespan. Moreover, the high temperature of steam with ultra-supercritical parameters can significantly reduce the material's strength properties.

Keywords: mathematical modeling, finite element method, steam turbine, blade strength, blade cooling, computer simulation

Як цитувати: Rusanov R. A., Rusanov A. V., Kriutchenko D. V., Bykov Yu. A., Degtyarev K. G. Computer modeling temperature and strength characteristics of an ultra-supercritical loop-type steam turbine rotor. *Вісник Харківського національного університету серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління.* 2024. вип. 64. С. 2024. С. 90-103.

https://doi.org/10.26565/2304-6201-2024-64-09

https://doi.org/10.26565/2304-6201-2024-64-09

Introduction

Turbine blades are the most expensive and complex parts of a steam turbine. Operating under extreme conditions, these blades are subjected to significant forces and impulse effects, which cause blade vibrations and the possibility of resonance. Moreover, high temperatures in the working fluid negatively affect the material strength, creating thermally induced stresses [1]. Blade loads are divided into static and dynamic. Static loads remain constant during operation in steady-state mode and change slowly during transient states, allowing inertial effects to be neglected. These include gas-dynamic forces on the blade profile surface, centrifugal forces on rotor blades, and thermal fields [2]. Centrifugal forces cause stresses and deformations in the blades. They can also lead to blade bending and twisting. Gas-dynamic forces also cause deformations and stresses in the blade's profile section, affecting the object's position [3]. During the design of turbine blades, both static and dynamic strength calculations, as well as thermal loads, are essential. Static loads include gas-dynamic forces and non-uniform temperature fields, the latter leading to blade deformations and cracking [4]. Long-term exposure to static and dynamic loads leads to microscopic damage accumulation, resulting in the development and growth of defects, which may cause cracks and failure [5]. The physical mechanisms that determine damage are not fully understood, but numerous models exist to evaluate blade efficiency and durability [6]. Static failure and high-cycle fatigue differ in their damage accumulation mechanisms. Thus, blade design must include calculations and experiments to verify not only static strength but also cyclic durability under high-cycle fatigue [7]. Environmental program requirements for reducing harmful emissions and rising fuel costs demand significant improvements in the efficiency of steam turbines. This can be achieved by increasing the initial temperature and pressure of steam and using intermediate reheating.

Currently, achievable steam parameters are:

-Temperature: above 700°C;

-Pressure: more than 30 MPa.

The application of such ultra-supercritical parameters allows turbines to achieve an efficiency of up to 50% [8]. However, long-term operation of blade rows at these temperatures requires the use of expensive high-temperature alloys or blade cooling methods. Using working steam as a blade coolant reduces the overall efficiency of the turbine unit, but, according to estimates, the efficiency loss will be significantly smaller than the efficiency gain achieved by using steam with ultra-supercritical initial parameters [9]. Cooling systems are a standard feature in gas turbines and are becoming necessary for steam turbines as well. The most accessible method, which avoids mixing the working fluid and coolant, is internal or convective cooling [10]. This method is used to cool both working and nozzle blades [11]. Other methods, which involve injecting cold gas into the working area, significantly reduce the overall turbine efficiency and do not allow optimization of the cooling zone [12]. The convective method of blade cooling is quite flexible and effective, offering many degrees of freedom. By selecting the shape and number of channels and the parameters of the cooling gas, satisfactory temperature characteristics of the blade material can be achieved [13]. However, the local nature of cooling can cause significant temperature gradients, leading to premature wear of turbine components. Therefore, for new and improved cooling system designs, a thorough study of the thermal state using modern numerical methods is necessary [14].

1. Description of the object of research

Figures 1.1-1.2 show longitudinal sections of new variants of high and medium pressure cylinders and their additional elements. The flow parts of the high and medium pressure cylinders of the loop type are limited by the upper contour lines 1 and the lower contour lines 2.

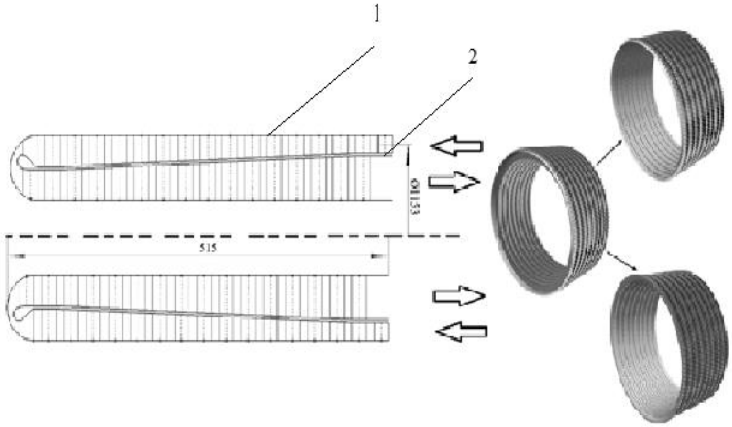


Fig. 1.1. High-pressure cylinder of a loop-type steam turbine

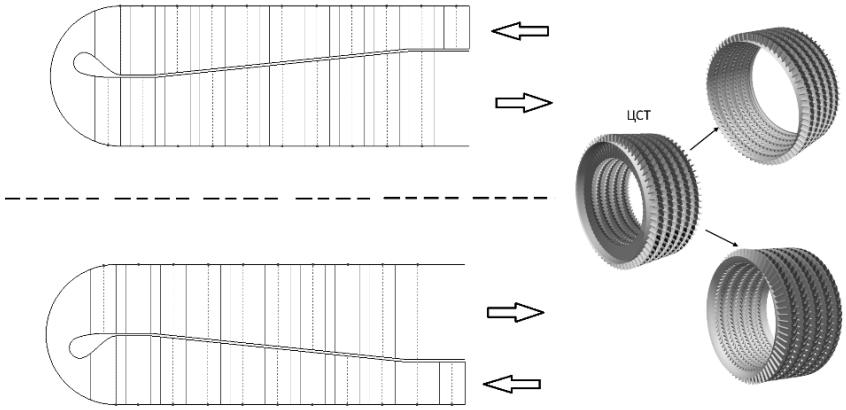


Fig. 1.2. Medium pressure cylinder of loop-type steam turbine

While the profile of the vanes is constant in all segments, the vanes of the upper and lower tiers are located asymmetrically according to the flow direction. There is no edge fastening mechanism in the blades; instead, a hard locking method is used. The geometric model of the rotor blade of the first and last stage is presented in Figure 1.3.

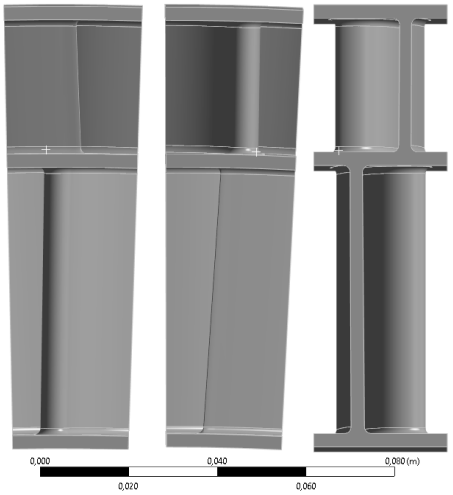


Fig. 1.3. Three-dimensional geometric model of the rotor blade of the first and last stage

2. Solution methods

2.1. Basic equations

To determine the stress-strain state of an elastic body, the system of second-order elliptic partial differential equations with respect to the displacement vector U is used

$$\mu\Delta U_j + (\lambda + \mu)\frac{\partial \vartheta}{\partial x_j} = 0, \Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2}, \vartheta = \mathrm{div}\mathbf{U}, \quad (1)$$

where λ and μ are the Lame coefficients.

Let us introduce the differential surface tension operator from the classical theory of elasticity as follows [22]

$$\mathbf{T^{n(x)}U} = 2\mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}} + \lambda\mathbf{n}\mathrm{div}\mathbf{U} + \mu(\mathbf{n} \times \mathrm{rot}\mathbf{U}),$$

where **n** is outward unit normal to the surface.

The following boundary value problem is formulated for an unbounded three-dimensional body weakened by a system of cuts $S = \bigcup_{k=1}^{n} S_k$, to determine U:

$$\Delta \mathrm{U_j} + (\lambda + \mu)\frac{\partial \vartheta}{\partial x_j} = 0, j=1,2,3, \mathbf{T^{n(x)}U(x)} = \mathbf{X(x)}, \mathbf{x} \in S. \quad (2)$$

Here **X**(**x**) is the given forces.

For the numerical analysis of the shell structure using ANSYS Workbench, homogeneous volumetric twenty-nodal finite elements of the SOLID186 type [15] were selected.

The SOLID186 element is a universal three-dimensional element (Fig. 2.1); quadratic approximation of displacements is used here. The element is determined by twenty nodes having three degrees of freedom: movement in the direction of the OX, OU, and OZ axes.

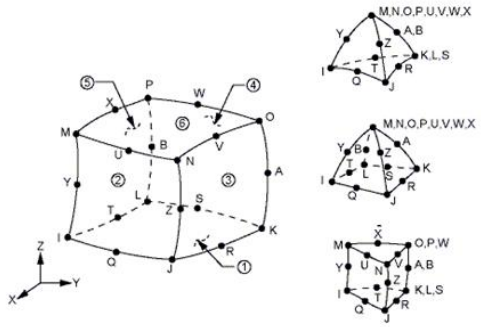


Fig. 2.1. Three-dimensional finite element SOLID186

Equations for the finite element are the equations of the theory of elasticity.

The advantages of the specified element are due to the fact that it can have an arbitrary spatial orientation, be specified in the form of a tetrahedron, a pyramid and a prism. This element allows you to model quite general geometric features of this structure, for example, sharp changes in thickness, the presence of corner points, etc., take into account plasticity, creep, hyperelasticity, strengthening, geometric nonlinearity [18]. Use of the selected element also allows you to take into account the anisotropy of the material, residual stresses and temperature loads. A finite element mathematical model of the rotor blade of the first and last stage was created, containing more than 1 million finite elements with thickenings in places of expected local stress maxima and zones of sharp changes in geometric parameters. Part of this model is shown in Fig. 2.2.

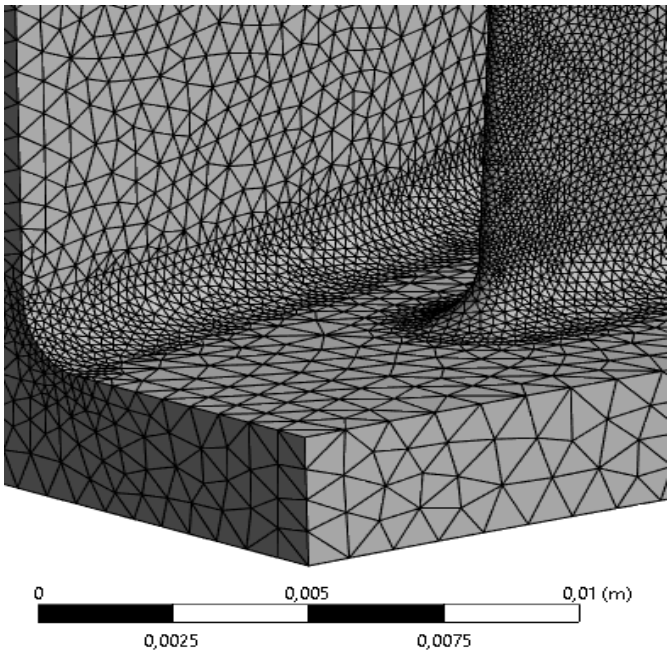


Fig. 2.2. *Finite element model of the scapula in the fillet region*

In general, the surfaces of the blades were thickened, especially around the leading edge, where the smallest size of the finite element was 0.075 of the base size, as well as the surface of the fillet rounding, where the size of the finite element was 0.05 of the base size.

In addition, as a result of additional test calculations, the quality of the finite element model was checked. The calculations showed that when the number of finite elements changes by 50 thousand, the change in the measurement result does not exceed 0.1%.

Thus, the choice was made precisely on the model described above, which allows for high-precision calculations and at the same time limits the requirements for the computing power of the equipment.

Modeling of the temperature state of the rotor blades of the first and last stages of the loop turbine was performed using the ANSYS Fluent Academic Edition R19.3 software package. The computational domain includes one rotor channel of the first and last stages of the high and medium pressure turbine and consists of four subdomains: the first stage rotor channel, the last stage rotor channel, the first stage blade cooling channel and the blade body. Numerical flow simulation was performed using a pressure-based algorithm that solves the Navier-Stokes equation [15], supplemented by the two-parameter Menter SST turbulence model [16], and the equation of state for water vapor according to the Peng-Robinson method [17]. The temperature distribution on the blades was simulated by solving the coupled problem of heat conduction and aerodynamics [15].

3. Numerical simulation of the temperature state of the blades

3.1 Simulation of the temperature state of the blades of the first rotor of the high-pressure cylinder

Tables 3.1 and 3.2 show the boundary conditions for the computational domains. The blade surfaces adjacent to the flow were assumed to be temperature and heat flux equal, while the others were assumed to be adiabatic. The mass flow rate for the cooling channels was determined to be 2% of the mass flow rate of the main flow in the last stage cylinder.

Table 3.1. Boundary conditions for blade channels

	1st stage rotor	16th stage rotor
Total pressure at the inlet, MPa	34,9	11,1
Total temperature at the inlet, K	975,0	760,3
Outlet pressure, MPa	32,9	9,7

Table 3.2. Boundary conditions for the cooling channel

Mass flow, kg/s	4,4
Total temperature at the inlet, K	760,3
Outlet pressure, MPa	9,7

On Figure 3.1, the temperature distribution on the blade surface is shown, while Figure 3.2 depicts the temperature isolines in the mid-cross-section of the first-stage blade, including the cooling channels. Figure 3.3 presents the isolines of the relative velocity of the main flow and the flow within the cooling channels in the mid-cross-section of the first-stage blade. Figure 3.4 illustrates the temperature distribution on the surface of the first-stage blade along the axial direction.

According to the distribution, the average surface temperature of the blade is 912.8 K, which is 62.2 K lower than the overall steam flow temperature.

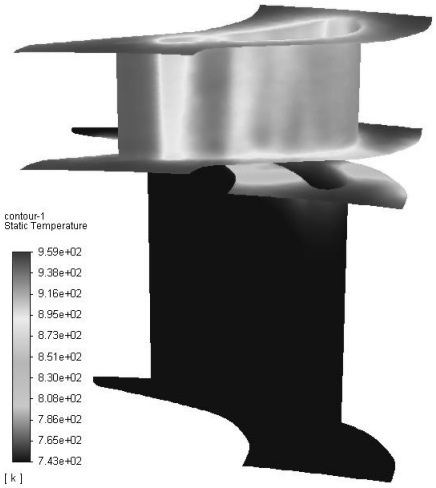


Fig. 3.1. *Temperature distribution on the surface of the blades*

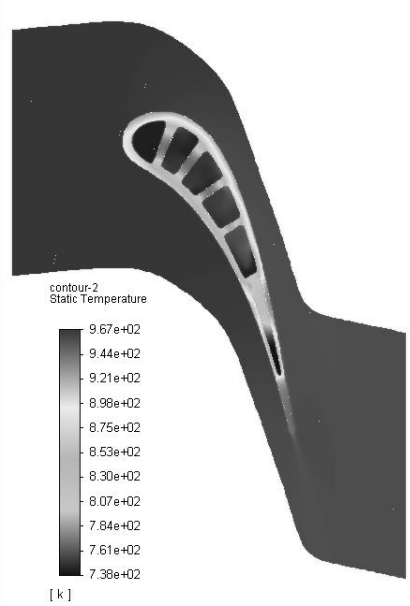


Fig. 3.2. *Temperature isolines in the middle cross-section of the blade of the first stage*

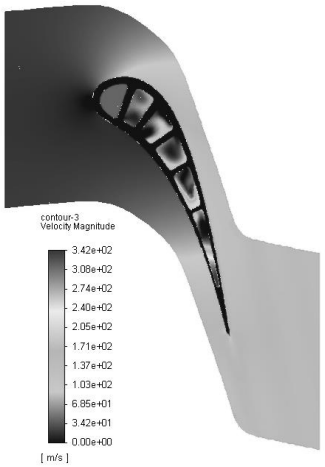


Fig. 3.3. *Relative velocity isolines in the middle cross-section of the blade of the first stage*

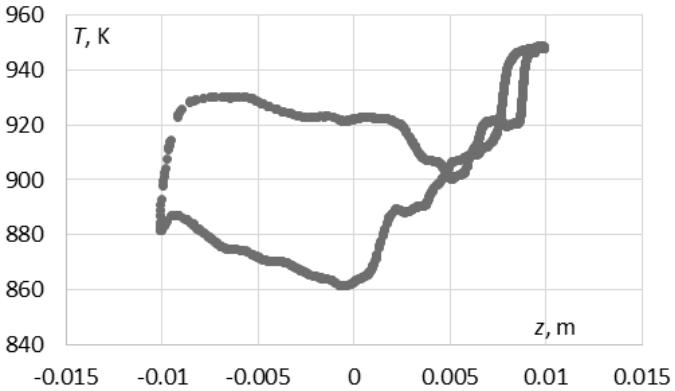


Fig. 3.4. *Temperature distribution on the blade surface of the first stage in the axial direction*

3.2 Modeling temperature state of blades of the first rotor of medium pressure cylinder

The thermal state modeling of the rotor blades in the first and last stages of the medium-pressure cylinder of the reheat turbine was performed similarly to the high-pressure cylinder [17].

Tables 3.3 and 3.4 present the boundary conditions for the calculation domains. On the surfaces of the blades in contact with the flow, conditions of temperature and heat flux equality were applied, while adiabatic conditions were set for other surfaces. The mass flow rate for the cooling channels was determined as 2% of the mass flow rate of the main flow in the last stage of the cylinder.

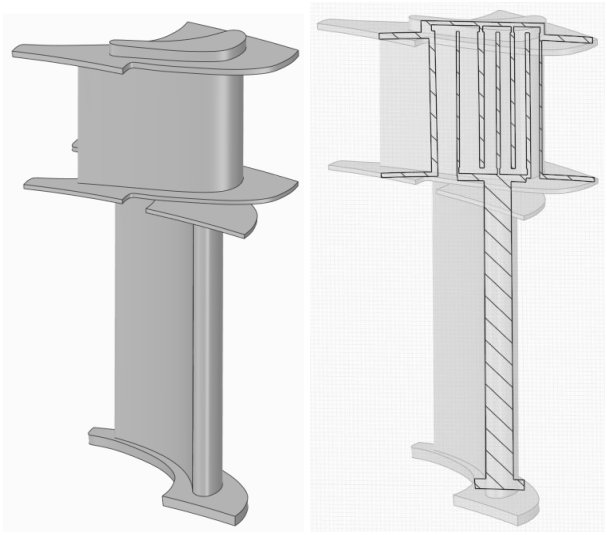
Table 3.3. Boundary conditions for blade channels

	1st stage rotor	10th stage rotor
Total pressure at the inlet, MPa	8,54	3,96
Total temperature at the inlet, K	973,2	834,0
Outlet pressure, MPa	7,88	3,53

Table 3.4. Boundary conditions for the cooling channel

Mass flow, kg/s	4,4
Total temperature at the inlet, K	834,0
Outlet pressure, MPa	3,53

Figure 3.5 shows the external appearance of the modeled blades and the configuration of the cooling channels within the first-stage blade.



a) b)

a – appearance of the blades; b – configuration of cooling channels

Fig. 3.5. *Appearance of blades, configuration of cooling channels*

Figure 3.6 shows the temperature distribution on the blade surface, while Figure 3.7 displays the temperature isolines in the midsection of the first-stage blade, which also includes cooling channels. Figure 3.8 presents the isolines of the relative velocity of the main flow and the flow within the cooling channels in the midsection of the first-stage blade. Figure 3.9 illustrates the temperature distribution along the axial direction of the first-stage blade.

According to the distribution, the average surface temperature of the blade is 932.8 K, which is 40.4 K lower than the total temperature of the steam flow.

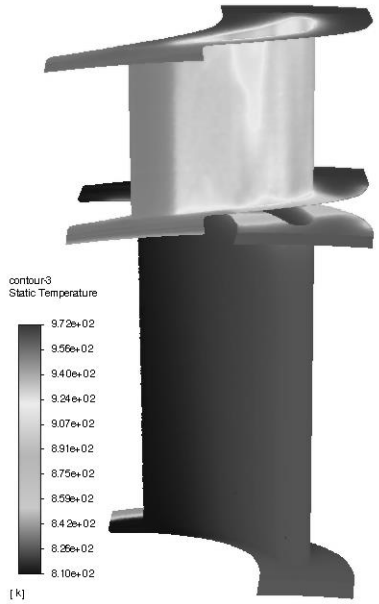


Fig. 3.6. *Temperature distribution on the surface of the blades*

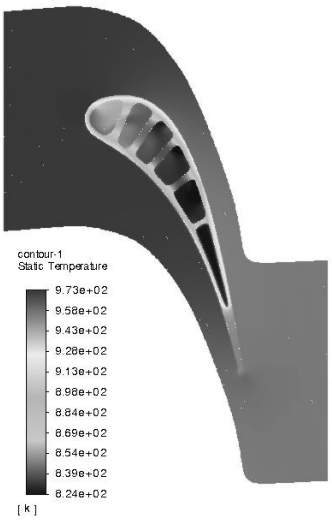


Fig. 3.7. Isolines of temperature in the middle cross-section of the blade of the first stage

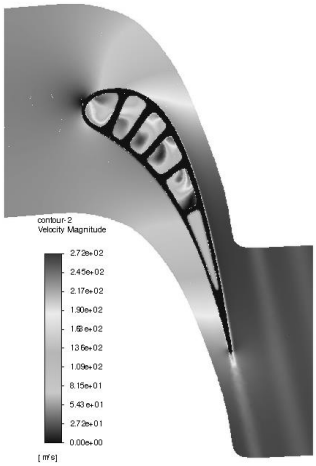


Fig. 3.8. Isolines of relative velocity in the middle section of the blade of the first stage

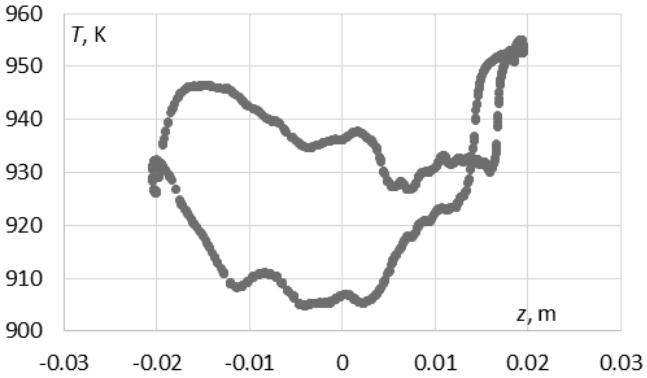


Fig. 3.9. Temperature distribution on the blade surface of the first stage in the axial direction

4. Strength Calculation of the First Rotor Blades and Analysis of Results

The dominant stresses in the rotor arise under the influence of centrifugal forces at a rotational speed of 3000 rpm. These stresses also depend on the density of the material used, with higher density causing higher stresses. As a result of preliminary calculations, a material with a density of 8160 kg/m³ was selected. The stress-strain state of the blades was determined based on calculations and is shown in Figure 4.1 as a distribution of equivalent stresses according to the von Mises criterion.

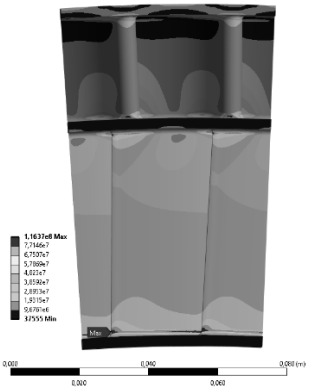


Fig. 4.1 – Distribution of equivalent stresses in the blades according to the Mises?

Global maximum stress (Global) is 116.4 MPa, it is located almost near the attachment point at the transition point of the fillet rounding into the main part of the blade at the exit from the crown, Fig. 4.2, it is the most highly loaded part of the rotor.

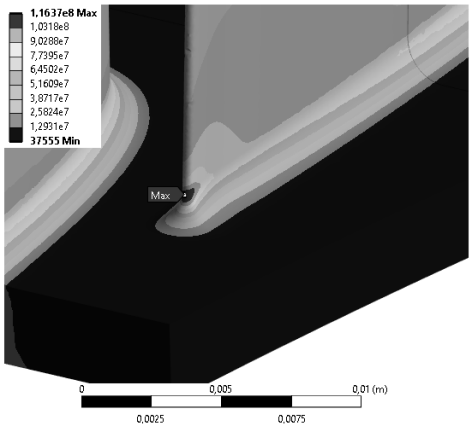


Fig. 4.2 – Localization of the global stress maximum

A similar location is observed for the next, localized stress maximum (L1), which is 98.8 MPa and is also situated near the base of the blade, but only at the crown inlet, as shown in Fig. 4.3. The Global and L1 maxima are interconnected and depend on the overall balance of the crown.

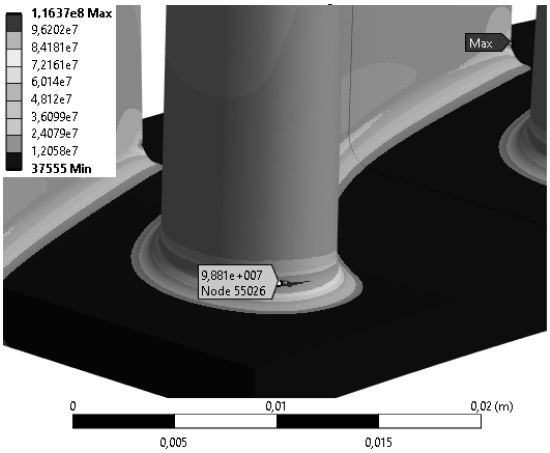


Fig. 4.3 – Localization of the L1 stress maximum

This distribution includes the main maximum stresses in the rotor as well as its local maxima. Additionally, due to the significant similarity of all the cylinder rotors, they will also contain these maxima with minor numerical differences. The global maximum stress (Global) is 116.4 MPa, located near the attachment point at the transition from the curvature of the blade fillet to the main part of the blade at the rotor outlet.

Based on the obtained data, a summary Table 4.1 was created, containing the calculation results for the identified stress maxima.

Table 4.1. Maximum stresses of the first rotor

	Global	L1	L2	L3	L4	L5	L6
Values, MPa	116,4	98,8	102,6	90,4	72,3	69,1	66,3
Temperature, ^{o}C	464	464	464	464	695	695	695

Conclusion

Promising design of a new high-pressure cylinder (HPC) and additional stages of the intermediate-pressure cylinder (IPC) for the K-300 series loop-type turbine with two-tier configuration has been presented [19]. The new flow path is designed to fit within the dimensions of the existing turbine. Number of innovative solutions, some of which are unprecedented in the global steam turbine industry, were employed in developing the new flow path, including a special meridional contour shape. The design was executed using a comprehensive methodology implemented in the IPMFlow and Ansys Workbench software packages. This methodology incorporates gas-dynamic calculations of varying complexity levels, as well as methods for constructing the spatial shape of blade paths based on a limited number of parameterized values [21].

Numerical study was conducted on the thermal and stress-strain state of the rotor blades in the high-pressure and intermediate-pressure cylinders of a state-of-the-art loop-type steam turbine with ultra-supercritical steam parameters. Strength calculations were performed to determine the stress distribution in the rotor blades, with the results indicating a sufficiently low level of maximum stresses. Numerical modeling of the thermal state of the first rotor blades in the high-pressure and intermediate-pressure cylinders of the loop turbine was also carried out. Cooling of the first-stage rotor blades was achieved by extracting steam from the last-stage channel. It was determined that using 2% of the steam mass flow for cooling reduces the average surface temperature of the first-stage blade by 62.2 K [20] for the high-pressure cylinder and by 40.4 K for the intermediate-pressure cylinder.

Awarded for support of a grant from the National Academy of Sciences of Ukraine to research laboratories/groups of young scientists of the National Academy of Sciences of Ukraine for research of priority areas in the development of science and technology 2022-2023.

REFERENCES

1. Nicholas T.E.W., Scobie J.A., Lock G.D., Tang H. A Model of Mass and Heat Transfer for Disc Temperature Prediction in Open Compressor Cavities (2024) Journal of Turbomachinery, 146 (4), art. no. 041001, DOI: 10.1115/1.4064082
2. Babazadeh M.A., Babaelahi M., Saadatfar M. New optimum configuration for the FGM rotating disc in gas turbines using combined thermal-mechanical analysis: *An analytical approach (2024) Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 238 (7), pp. 3038 - 3051, DOI: 10.1177/09544062231195447
3. Sierikova O., Strelnikova E., Degtyariov K. Numerical Simulation of Strength and Aerodynamic Characteristics of Small Wind Turbine Blades (2023) Lecture Notes in Networks and Systems, 657 LNNS, pp. 357 - 370. https://link.springer.com/chapter/10.1007/978-3-031-36201-9_31
4. Ghoreishi S.M.N., Mehri Khansari N. Mode (I, II, III) Stress Intensity Factors of Composite-Coated Gas Turbine Blade Using Semi-Elliptical Crack (2023) Iranian Journal of Science and Technology - Transactions of Mechanical Engineering, 47 (4), pp. 1841-1857, DOI: 10.1007/s40997-023-00592-7
5. Hu P., Meng Q., Fan W., Gu W., Wan J., Li Q. Vibration characteristics and life prediction of last stage blade in steam turbine Based on wet steam model (2024) Engineering Failure Analysis, 159, art. no. 108127, DOI: 10.1016/j.engfailanal.2024.108127
6. Strelnikova O., Gnitko V., Degtyariov K., Tonkonozhenko A. Advanced computational models and software on predicting the effective elastic properties for computer-simulated structures of nanocomposite, (2020) *2020 IEEE KhPI Week on Advanced Technology, KhPI Week 2020 -*

Conference Proceedings, art. no. 9250093, pp. 171 - 176. https://ieeexplore.ieee.org/document/9250093
7. Zachary, J. "Steam Turbine Design Considerations for Ultrasupercritical Cycles." Proceedings of the ASME Turbo Expo 2010: Power for Land, Sea, and Air. Volume 7: Turbomachinery, Parts A, B, and C. Glasgow, UK. June 14–18, 2010. pp. 2171-2179. ASME. https://doi.org/10.1115/GT2010-22317
8. Wojciech Kosman. The influence of external cooling system on the performance of supercritical steam turbine cycles. archives of thermodynamics Vol. 31(2010), No. 3, 131–144 DOI: 10.2478/v10173-010-0019-4
9. Bohn, D., & Kusterer, K. (2005). Turbulent and conjugate heat transfer simulation for gas turbine application. Modelling and Simulation of Turbulent Heat Transfer, 16, 247-313. doi:10.2495/978-1-85312-956-8/08
10. Bohn, D, Ren, J, & Kusterer, K. "Cooling Performance of the Steam-Cooled Vane in a Steam Turbine Cascade." Proceedings of the ASME Turbo Expo 2005: Power for Land, Sea, and Air. Volume 3: Turbo Expo 2005, Parts A and B. Reno, Nevada, USA. June 6–9, 2005. pp. 217-226. ASME. https://doi.org/10.1115/GT2005-68148
11. Wojciech Kosman, Thermal analysis of cooled supercritical steam turbine components, Energy, Volume 35, Issue 2, 2010, Pages 1181-1187, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2009.05.033.
12. Grzegorz Nowak, Włodzimierz Wróblewski, Iwona Nowak, Convective cooling optimization of a blade for a supercritical steam turbine, International Journal of eat and Mass Transfer, Volume 55, Issues 17–18, 2012, Pages 4511-4520, ISSN 0017-9310, https://doi.org/10.1016/j.ijheatmasstransfer.2012.03.072
13. Włodzimierz Wróblewski, Numerical evaluation of the blade cooling for the supercritical steam turbine, Applied Thermal Engineering, Volume 51, Issues 1–2, 2013, Pages 953-962, ISSN 1359-4311, https://doi.org/10.1016/j.applthermaleng.2012.10.048.
14. ANSYS Mechanical APDL Fluids Theory Guide. – Canonsburg, PA: ANSYS, Inc., 2019. – 914 p. https://www.academia.edu/33546434/ANSYS_Mechanical_APDL_Fluids_Analysis_Guide
15. ANSYS Fluent Theory Guide. – Canonsburg, PA: ANSYS, Inc., 2019. – 988 p. https://dl.cfdexperts.net/cfd_resources/Ansys_Documentation/Fluent/Ansys_Fluent_Theory_Guide.pdf
16. Menter F. R. Two-equation eddy viscosity turbulence models for engineering applications / F. R. Menter // AIAA J. – 1994. – 32, № 11. – P. 1299–1310.
17. Peng D.-Y. A New Two-Constant Equation of State / D.-Y. Peng, D. B. Robinson // Industrial & Engineering Chemistry Fundamentals. – 1976. – 15 (1). – P. 59-64. DOI: 10.1021/i160057a011
18. Vo D.-T., Mai T.-D., Kim B., Jung J.-S., Ryu J. Numerical investigation of crack initiation in high-pressure gas turbine blade subjected to thermal-fluid-mechanical low-cycle fatigue (2023) International Journal of Heat and Mass Transfer, 202, art. no. 123748, DOI: 10.1016/j.ijheatmasstransfer.2022.123748
19. Rusanov A.V., Rusanov R.A., Dehtyar′ov K.H., Pal′kov S.A., Pal′kov I.A., Kryutchenko D.V., Protochna chastyna parovoyi turbiny petl′ovoho typu dlya roboty na ul′tra-superkrytychnykh pochatkovykh parametrakh pary, № 156077, MPK: F01D25/30, 08.05.2024, byul. № 19/2024.https://sis.nipo.gov.ua/uk/search/detail/1798427/[in Ukrainian]
20. Rusanov A.V., Rusanov R.A., Degtyar'ov K.G., Pal'kov S.A., Pal'kov Í.A., Kryutchenko D.V., Bikov YU.A., Sistema okholodzhennya lopatok rotora turbíni petl'ovogo tipu, № 156085, F01D5/18, 0
21. Solovey, V., Zipunnikov, M., Rusanov, R. Increasing the manoeuvrability of power units of the thermal power plants due to applying the hydrogen-oxygen systems. JPhys Energy. 2023. Vol. 5. 014012, doi 10.1088/2515-7655/aca9ff
22. Strelnikova, E., Kriutchenko, D., Gnitko, V., Degtyarev, K.: Boundary element method in nonlinear sloshing analysis for shells of revolution under longitudinal excitations, Engineering Analysis with Boundary Elements, 111, 2020, 78-87, https://doi.org/10.1016/j.enganabound.2019.10.008.

СПИСОК ЛІТЕРАТУРИ

1. E.W.Nicholas, Scobie J.A., Lock G.D., Tang H. A Model of Mass and Heat Transfer for Disc Temperature Prediction in Open Compressor Cavities (2024) Journal of Turbomachinery, 146 (4), art. no. 041001, DOI: 10.1115/1.4064082

2. M.A. Babazadeh, M. Babaelahi, M. Saadatfar New optimum configuration for the FGM rotating disc in gas turbines using combined thermal-mechanical analysis: *An analytical approach (2024) Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 238 (7), pp. 3038 - 3051, DOI: 10.1177/09544062231195447
3. O. Sierikova, E. Strelnikova, K. Degtyariov. Numerical Simulation of Strength and Aerodynamic Characteristics of Small Wind Turbine Blades (2023) Lecture Notes in Networks and Systems, 657 LNNS, pp. 357 - 370. https://link.springer.com/chapter/10.1007/978-3-031-36201-9_31
4. S.M.N. Ghoreishi, Mehri Khansari. Mode (I, II, III) Stress Intensity Factors of Composite-Coated Gas Turbine Blade Using Semi-Elliptical Crack (2023) Iranian Journal of Science and Technology - Transactions of Mechanical Engineering, 47 (4), pp. 1841-1857, DOI: 10.1007/s40997-023-00592-7
5. P. Hu, Q. Meng, W. Fan, W. Gu, J. Wan, Q. Li. Vibration characteristics and life prediction of last stage blade in steam turbine Based on wet steam model (2024) Engineering Failure Analysis, 159, art. no. 108127, DOI: 10.1016/j.engfailanal.2024.108127
6. O. Strelnikova, V. Gnitko, K. Degtyariov, A. Tonkonozhenko. Advanced computational models and software on predicting the effective elastic properties for computer-simulated structures of nanocomposite, (2020) *2020 IEEE KhPI Week on Advanced Technology, KhPI Week 2020 - Conference Proceedings*, art. no. 9250093, pp. 171 - 176. https://ieeexplore.ieee.org/document/9250093
7. J. Zachary, "Steam Turbine Design Considerations for Ultrasupercritical Cycles." Proceedings of the ASME Turbo Expo 2010: Power for Land, Sea, and Air. Volume 7: Turbomachinery, Parts A, B, and C. Glasgow, UK. June 14–18, 2010. pp. 2171-2179. ASME. https://doi.org/10.1115/GT2010-22317
8. W. Kosman. The influence of external cooling system on the performance of supercritical steam turbine cycles. archives of thermodynamics Vol. 31(2010), No. 3, 131–144 DOI: 10.2478/v10173-010-0019-4
9. D. Bohn, K. Kusterer, (2005). Turbulent and conjugate heat transfer simulation for gas turbine application. Modelling and Simulation of Turbulent Heat Transfer, 16, 247-313. doi:10.2495/978-1-85312-956-8/08
10. D. Bohn, J. Ren, K. Kusterer. "Cooling Performance of the Steam-Cooled Vane in a Steam Turbine Cascade." Proceedings of the ASME Turbo Expo 2005: Power for Land, Sea, and Air. Volume 3: Turbo Expo 2005, Parts A and B. Reno, Nevada, USA. June 6–9, 2005. pp. 217-226. ASME. https://doi.org/10.1115/GT2005-68148
11. W. Kosman, Thermal analysis of cooled supercritical steam turbine components, Energy, Volume 35, Issue 2, 2010, Pages 1181-1187, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2009.05.033.
12. G. Nowak, W. Wróblewski, I. Nowak, Convective cooling optimization of a blade for a supercritical steam turbine, International Journal of eat and Mass Transfer, Volume 55, Issues 17–18, 2012, Pages 4511-4520, ISSN 0017-9310, https://doi.org/10.1016/j.ijheatmasstransfer.2012.03.072
13. W. Wróblewski, Numerical evaluation of the blade cooling for the supercritical steam turbine, Applied Thermal Engineering, Volume 51, Issues 1–2, 2013, Pages 953-962, ISSN 1359-4311, https://doi.org/10.1016/j.applthermaleng.2012.10.048.
14. ANSYS Mechanical APDL Fluids Theory Guide. – Canonsburg, PA: ANSYS, Inc., 2019. – 914 p. https://www.academia.edu/33546434/ANSYS_Mechanical_APDL_Fluids_Analysis_Guide
15. ANSYS Fluent Theory Guide. – Canonsburg, PA: ANSYS, Inc., 2019. – 988 p. https://dl.cfdexperts.net/cfd_resources/Ansys_Documentation/Fluent/Ansys_Fluent_Theory_Guide.pdf
16. Menter F. R. Two-equation eddy viscosity turbulence models for engineering applications / F. R. Menter // AIAA J. – 1994. – 32, № 11. – P. 1299–1310.
17. Peng D.-Y. A New Two-Constant Equation of State / D.-Y. Peng, D. B. Robinson // Industrial & Engineering Chemistry Fundamentals. – 1976. – 15 (1). – P. 59-64. DOI: 10.1021/i160057a011
18. T. D..Vo, T. D. Mai, B. Kim, J.-S. Jung, J . Ryu. Numerical investigation of crack initiation in high-pressure gas turbine blade subjected to thermal-fluid-mechanical low-cycle fatigue (2023) International Journal of Heat and Mass Transfer, 202, art. no. 123748, DOI: 10.1016/j.ijheatmasstransfer.2022.123748
19. А.В. Русанов, Р.А. Русанов, К.Г. Дегтярьов, С.А. Пальков, І.А.Пальков, Д.В. Крютченко, Проточна частина парової турбіни петльового типу для роботи на ультра-суперкритичних початкових параметрах пари, № 156077, МПК: F01D25/30, 08.05.2024, бюл. № 19/2024. https://sis.nipo.gov.ua/uk/search/detail/1798427/

20. А.В. Русанов, Р.А. Русанов, К.Г. Дегтярьов, С.А. Пальков, І.А.Пальков, Д.В. Крютченко, Ю.А. Биков, Система охолодження лопаток ротора турбіни петльового типу, № 156085, F01D5/18, 08.05.2024, бюл. № 19/2024. https://sis.nipo.gov.ua/uk/search/detail/1798401/
21. V. Solovey, M. Zipunnikov, R. Rusanov. Increasing the manoeuvrability of power units of the thermal power plants due to applying the hydrogen-oxygen systems. JPhys Energy. 2023. Vol. 5. 014012, doi 10.1088/2515-7655/aca9ff
22. E. Strelnikova, D. Kriutchenko, V. Gnitko, K. Degtyarev. Boundary element method in nonlinear sloshing analysis for shells of revolution under longitudinal excitations, Engineering Analysis with Boundary Elements, 111, 2020, 78-87, https://doi.org/10.1016/j.enganabound.2019.10.008.

Русанов Роман Андрійович *Д-р. філос., старший науковий співробітник, відділ термогазодинаміки енергетичних машин, інститут енергетичних машин і систем ім. А. М. Підгорного НАН України, вул. Комунальників, 2/10, м. Харків, Україна, 61046*

Русанов Андрій Вікторович *Академік НАНУ, директор інституту енергетичних машин і систем ім. А. М. Підгорного НАН України, вул. Комунальників, 2/10, м. Харків, Україна, 61046*

Крютченко Денис Володимирович *Д-р. філос., науковий співробітник, відділ термогазодинаміки енергетичних машин, Інститут енергетичних машин і систем ім. А. М. Підгорного НАН України, вул. Комунальників, 2/10, м. Харків, Україна, 61046*

Биков Юрій Адольфович *К.т.н., старший науковий співробітник, відділ термогазодинаміки енергетичних машин, інститут енергетичних машин і систем ім. А. М. Підгорного НАН України, вул. Комунальників, 2/10, м. Харків, Україна, 61046*

Дегтярьов Кирило Георгійович *К.т.н., старший науковий співробітник, відділ термогазодинаміки енергетичних машин, інститут енергетичних машин і систем ім. А. М. Підгорного НАН України, вул. Комунальників, 2/10, м. Харків, Україна, 61046*

Комп’ютерне моделювання температурних та міцнісних характеристик ротору ультрасуперкритичної парової турбіни петельного типу

Актуальність. У світі з’явилась тенденція, пов’язана з постійним зростанням вимог до ефективності та надійності енергетичного обладнання, зокрема ультрасуперкритичних парових турбін. Такі турбіни працюють у складних умовах високих температур і тисків, що може викликати значні термомеханічні напруження у роторі. Комп’ютерне моделювання ідеально підходить для розв’язку подібних задач. Тому є актуальним дослідження, присвячено розрахункам на міцність та визначення температурних характеристик лопаток у циліндрах високого та середнього тиску. За допомогою даних задач інженери можуть гарантувати міцність та довговічність вузлів у парових турбінах.
Мета. Провести розрахунки на міцність та розподіл температури лопаток для циліндрів високого та середнього тиску парової турбіни петельного типу із ультра-суперкритичними початковими параметрами пари.
Результати. Були отримані дані по розподіленню температурних полів у лопатках ротора циліндру високого та середнього тиску. Використовуючи результати температурного розрахунку, було отримано оцінку міцності лопаток першого ротору циліндру високого тиску під впливом нерівномірності розподілу температури і обертання ротору. Охолодження лопаток першого ступеня турбіни досягається завдяки конвективному теплообміну від потоку холодної пари з останнього ступеня до внутрішніх каналів лопаток.
исновки. За результатами зроблено висновок, про ефективність обраного способу охолодження лопаток та рівень максимальних напружень всередині лопаток. Однією з особливостей умов роботи лопаток турбіни петельного типу є нерівномірне нагрівання, яке відбувається як у перехідних, так і у стаціонарних режимах роботи лопаток. Нерівномірне нагрівання призводить до виникнення температурних напружень в лопатках, що негативно впливає на їхній ресурс. Крім того, висока температура пари з ультра-суперкритичними параметрами може значно знизити міцнісні властивості матеріалу.

Ключові слова: *математичне моделювання, метод скінченних елементів, парова турбіна, міцність лопаток, охолодження лопаток*

Надійшла у першій редакції 07.09.2024, в останній - 14.10.2024.
The first version has been received on 07.09.2024, the final version - on 14.10.2024.

DOI: https://doi.org/10.26565/2304-6201-2024-64-10

УДК (UDC) 004.415.2

Vladyslav Samoilenko *Student Kharkiv National University named V. N. Karazin 61000*
e-mail: vladyslav.samoilenko@student.karazin.ua;
ORCID: 0009-0001-3217-4473
Dmitry Bulavin *PhD, associate professor Kharkiv National University named V. N. Karazin 61000. e-mail:* d.bulavin@karazin.ua
ORCID: 0000-0002-4840-4763

Self-Configuring Virtual Machine Environments Using Ansible Pull: A Review of Approaches and Challenges

The automation of virtual machine (VM) configuration has become an integral part of modern IT infrastructure, ensuring compliance with the high demands of scalability, reliability, and security. With the increasing adoption of cloud technologies, there is a growing need for standardized approaches to infrastructure management that eliminate the limitations of traditional methods, such as manual configuration or individual scripts. The goal of this research is to design an architecture for automated VM self-configuration based on Ansible Pull in combination with tools like Terraform, Hashicorp Vault, and Git to ensure flexibility, autonomy, and security.
To build the system, an analytical approach was used to evaluate existing configuration tools, model the architecture using Git repositories, Terraform, and Vault, and experimentally implement cloud-init to initialize Ansible Pull at the VM level. The proposed architecture automates the VM configuration process, ensuring scalability and reducing dependence on centralized management servers. The implementation of Ansible Pull with a tagging system provides idempotent task execution, regular updates, and maintenance of the desired system state.
It has been proven that combining Ansible Pull with Terraform, Git, and Vault ensures simplicity of implementation, flexibility in scaling, and secure configuration management. The proposed approach is effective for dynamic environments requiring frequent updates and aligns with modern DevOps practices.
Keywords: *Ansible Pull, automation of the configuration, Terraform, Vault, self-configuring, DevOps, Ansible*

Як цитувати: Samoilenko V. V., Bulavin D. O. «Self-Configuring Virtual Machine Environments Using Ansible Pull: A Review of Approaches and Challenges». *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2024. вип. 64. С.104-111. https://doi.org/10.26565/2304-6201-2024-64-10

How to quote: V. Samoilenko, D. Bulavin, "Self-Configuring Virtual Machine Environments Using Ansible Pull: A Review of Approaches and Challenges" *Bulletin of V.N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 64, pp. 104-111, 2024. https://doi.org/10.26565/2304-6201-2024-64-10

1 Introduction

In the modern world of information technology, process automation is a key factor in enhancing efficiency and reducing operational costs. One of the primary tasks in this context is the automation of virtual machine (VM) configuration, which constitutes a vital part of the infrastructure for most organizations. Traditional methods of manual configuration are not only labor-intensive but are also prone to human error, potentially leading to severe system failures.

Automation tools such as Ansible, Puppet, and Chef provide a means to simplify this process by employing a declarative approach. Among these, Ansible stands out for its simplicity, idempotency (ensuring consistent execution results), support for YAML files, and agentless architecture. These features make Ansible an effective tool for managing large-scale server infrastructures in cloud environments, such as AWS, Google Cloud, and Azure.

This study focuses on exploring the capabilities of Ansible Pull as a method for self-organizing virtual machine environments. The emphasis is placed on configuration management, performance optimization, and cost reduction when employing this approach. In addition to analyzing the advantages of Ansible Pull, the paper examines key aspects of deployment process management, including comparisons with similar solutions. The primary challenges associated with integrating Ansible into scalable environments are identified, and strategies for overcoming these challenges are proposed.

Thus, this work aims to establish a foundation for further research and the implementation of automated solutions for virtual machine configuration in modern IT environments.

2 Analysis of the Problem of Self-Configuring Virtual Environments

2.1 Analysis of Modern Methods and Their Limitations

Traditionally, self-configuration is an automated process through which a system configures itself to operate by following predefined instructions without requiring manual intervention. This process includes:

- Installation and management of software dependencies.
- Creation and configuration of user accounts with appropriate access rights.
- Configuration of network settings and ensuring their security.
- Monitoring and ensuring compliance with predefined standards.

The goal of self-configuration is to minimize the risk of human error, accelerate deployment cycles, and ensure the scalability and repeatability of the infrastructure configuration process. This is particularly important for environments that require frequent updates or have complex distributed configurations.

Today, there are several tools and approaches to configuration automation, each with its own advantages and disadvantages:

1. **Custom Scripts:**

- These are easy to create but poorly scalable and challenging to maintain in dynamic environments. Updates often require manual intervention, increasing the risk of errors.

2. **Configuration Management Tools (e.g., Puppet, Chef):**

- These tools are powerful but often require the installation of specialized agents on each node, complicating their use and increasing resource consumption.

3. **Ansible Push:**

- Ansible's push model eliminates the need for agents, simplifying configuration management. However, this approach can create bottlenecks when managing a large number of nodes due to the centralized execution of commands.

4. **Ansible Pull:**

- Ansible Pull offers a decentralized approach where managed nodes retrieve configurations from a Git repository. This approach ensures scalability and reduces dependency on a central control node but comes with its own challenges:
- **Dependency on Git:** Any issues with the repository can halt configuration updates.
- **Network Stability:** Requires a stable SSH connection for initial setup and updates.
- **Version Management:** Ensuring consistent configurations across all nodes can be challenging without proper control.

Thus, Ansible is not the only tool in the configuration automation space. Several other leading tools predate Ansible's release in 2012: SaltStack in 2011, Puppet in 2005, and Chef in 2008. A comparison of the popularity of search queries among these tools is shown in Figure 2.1.

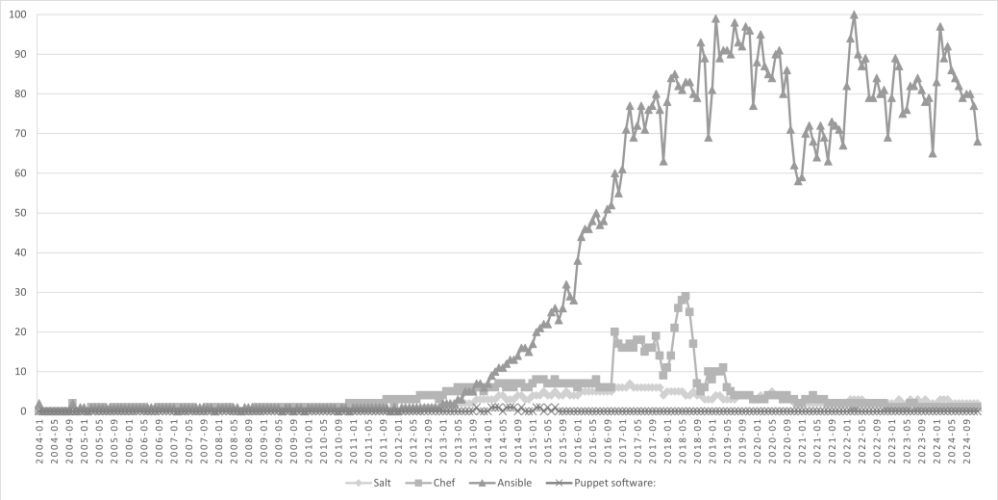


Fig.2.1 Comparing of the search requests Salt, Chef, Ansible ma Puppet popularity in Google.

The choice of tools and methods for self-configuration depends on specific requirements such as scalability, ease of use, and integration capabilities. With Ansible Pull, self-configuration becomes a robust, scalable, and repeatable process that ensures the consistent provisioning of systems according to desired specifications described in code. Storing configurations as code provides significant flexibility and simplifies maintenance.

2.2 Ansible Pull Analysis: Advantages and Disadvantages

Ansible is a popular tool in the category of Infrastructure-as-Code (IaC) that enables automation of deployment and configuration of IT infrastructure. Its agentless architecture, intuitive YAML syntax, and scalability have made Ansible a top choice for organizations seeking to improve process efficiency. Ansible developers create playbooks containing a series of tasks that can be automatically executed on a set of hosts to achieve the desired infrastructure state.

As an automation technology, Ansible is based on the following principles:

- **Agentless Architecture.** Ansible does not require installing software on managed nodes (except in the pull model).
- **Simplicity.** YAML-based playbooks allow easy creation and comprehension of automation scripts. Ansible is also decentralized, using SSH with existing OS credentials to access remote machines.
- **Scalability and Flexibility.** Its modular design allows for rapid scaling, supporting a wide range of platforms and environments.
- **Error-Free, Predictable, and Idempotent.** Ansible files describe infrastructure as code, ensuring the same outcome regardless of how many times the playbooks are executed. [1]

Ansible Pull offers a unique model for self-configuring environments, where managed nodes independently fetch and apply configurations from a repository. This approach enables:

- Avoiding dependence on a central control node.
- Reducing risks associated with single points of failure.
- Efficient operation in distributed infrastructure environments.

Ansible Pull is ideal for dynamic environments that require regular updates, meeting the demands of modern agile systems. [2]

Ansible Pull utilizes a decentralized configuration management method where each node independently fetches and applies updates from a central Git repository. This approach eliminates the need for a centralized control server, enhancing system fault tolerance. Nodes periodically execute the ansible-pull command to download and apply the latest playbooks. Additionally, playbooks can include tasks to update Ansible Pull itself, increasing the autonomy of the process.

Red Hat, the developer of Ansible, provides tools to implement this model but not out-of-the-box solutions. Automating the execution of ansible-pull requires the use of task schedulers such as cron jobs, systemd timers, or similar tools [3]. This modular approach is characteristic of Ansible, offering flexible tools rather than rigid solutions.

Advantages:

1. Autonomy: Each node fetches and applies its configuration independently, minimizing reliance on a central control server and ensuring resilience against network disruptions or server downtime.
2. Scalability: Adding new nodes is simplified since the self-configuration process is integrated into deployment.
3. Flexibility: Changes in the repository are automatically applied during the next execution cycle of ansible-pull, ensuring prompt and consistent updates across all nodes.
4. Version Control: Git integration provides robust version management, allowing effective collaboration, tracking changes, and reverting to previous configurations when needed.
5. No Single Point of Failure: Eliminating reliance on a central server reduces risks associated with potential unavailability.

Disadvantages:

1. Network Dependency: Nodes require connectivity to the Git repository for updates. This can be mitigated by setting up local repository mirrors or using caching mechanisms.
2. Lack of Centralized Management: Unlike the push model, task execution control is more complex, requiring additional monitoring tools such as Ansible callback modules or third-party systems like Prometheus or ELK Stack.

3. Error Debugging: Error logs are stored locally on each node, complicating centralized troubleshooting, especially in large-scale environments. Integration with centralized log collection systems can address this issue.
4. Initial Node Configuration: For ansible-pull to work, each node must be pre-configured with access to the Git repository and necessary dependencies. While this adds steps to configuration, it is a common practice in decentralized systems.
5. Security Challenges: Granting nodes access to the Git repository can pose risks related to data leakage or unauthorized access. Recommendations include restricting access rights, using SSH keys, and secure communication channels (e.g., VPN).

Ansible Pull is a powerful solution for decentralized configuration management, offering autonomy and flexibility. However, its implementation requires additional setup for monitoring, security, and debugging. When properly configured, the system demonstrates high scalability and reliability.

2.3 Comparative Analysis of Existing Solutions

We compare popular configuration management systems such as Chef, Puppet, SaltStack, and Ansible Pull based on key criteria: autonomy, flexibility, security, ease of updates, and scalability to highlight their strengths and weaknesses.

Chef: Uses a pull model where nodes fetch configurations (cookbooks) from a central server. While effective for managing complex configurations, its reliance on the Ruby programming language and the need for a dedicated central server complicates setup and usage.

Puppet: Operates similarly, with nodes fetching manifests from a Puppet master. It supports scalability and consistency but requires installing the Puppet agent and master, increasing complexity. Its domain-specific language (Puppet DSL) reduces accessibility for teams unfamiliar with it.

SaltStack: Primarily uses a push model with a master-minion architecture. While SaltStack offers significant scalability, its setup and usage are complex, though it is compatible with YAML.

Ansible Pull: Unlike the others, Ansible Pull eliminates the need for centralized servers or agents. Nodes independently fetch configurations from a Git repository, making it lightweight and straightforward. Its YAML-based approach and Git integration align with modern DevOps practices, ensuring an intuitive process.

Key Comparative Insights:

- Autonomy: Ansible Pull ensures node autonomy, as each node independently manages updates. Chef and Puppet also support autonomous operation but rely on central servers, creating a potential single point of failure.
- Flexibility: Ansible Pull offers high flexibility with its Git-based workflow, facilitating seamless integration with existing tools and processes. Salt is versatile but more complex to configure, whereas Chef and Puppet require specialized languages or infrastructure, limiting flexibility in some environments.
- Security: Chef and Puppet provide robust authentication and access control through master-agent models. Ansible Pull leverages Git for security, requiring careful management of credentials and repository access control. Salt also provides high security but is harder to configure in pull mode.
- Ease of Updates: Periodic Ansible Pull synchronization with Git ensures quick and consistent updates. Puppet and Chef support automatic updates but with potential delays. Salt’s push model provides real-time updates but loses this advantage in pull mode.
- Scalability: Puppet and Chef scale well due to their master-agent architecture. Ansible Pull is suitable for distributed systems as nodes operate independently. However, large-scale deployments can strain Git traffic during updates, which can be mitigated through update scheduling. Salt is efficient in push scenarios but less performant in pull mode.

Ansible Pull is an excellent choice for environments favoring lightweight, decentralized, and straightforward self-configuration processes. It is especially effective where teams use Git, which was adopted by approximately 93% of developers in 2023 [4]. However, environments requiring highly centralized control or low-latency updates may benefit from solutions like Puppet, Chef, or Salt. Ultimately, Ansible Pull is ideal for organizations seeking scalability, flexibility, efficiency, and seamless integration with modern DevOps workflows.

3 Description of the Self-Configuring Virtual Machine Environments solution

3.1 Architecture

After analyzing all possible approaches to architecture design, we propose a solution distinguished by enhanced flexibility and efficiency.

The proposed architecture utilizes several Git repositories to organize and manage different aspects of the system:

- Ansible Repository: Contains roles and corresponding configurations for automation.
- Ansible-Inventory Repository: Includes inventory files necessary for accurate system configuration.
- Terraform Repository: Stores infrastructure-as-code (IaC) definitions, enabling automated infrastructure creation and management.
- Terraform-Modules Repository: Used for reusable modules to enhance scalability and code reusability.

To manage secrets, Hashicorp Vault is integrated, providing a high level of security. EC2 instances are configured to authenticate with Vault using AWS IAM roles, ensuring secure retrieval of secrets and facts required for configuration processes.

This architecture ensures:

- Separation of responsibilities through specialized repositories.
- Secure handling of sensitive data.
- Scalability and flexibility through the use of Terraform modules.

An example of the proposed architecture is shown in Fig. 3.1.

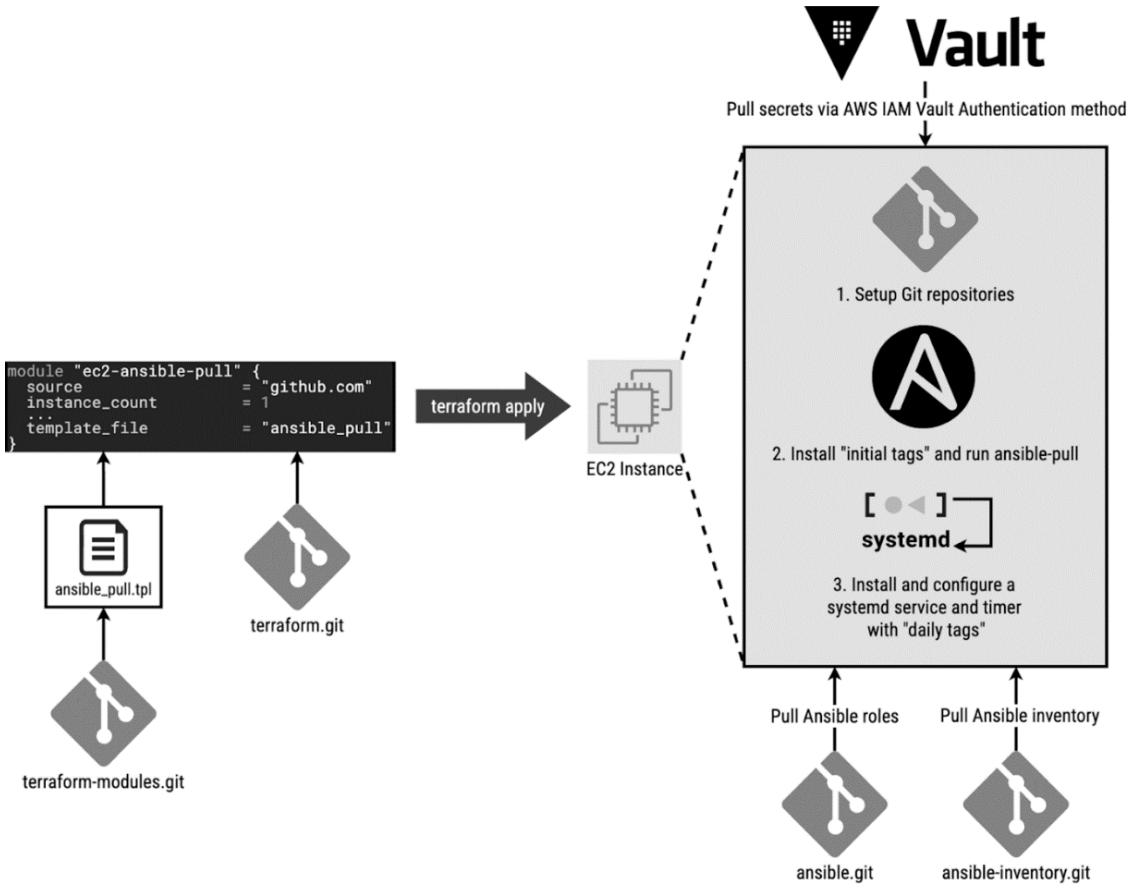


Fig.3.1. The overall architecture of the Ansible Pull solution using Terraform, Ansible, Git, and Vault

When an EC2 virtual machine instance is created using Terraform, a cloud-init script embedded in the configuration is executed during its launch. Cloud-init facilitates the connection between VM deployment and configuration without requiring additional user intervention [5]. This script plays a pivotal role in initializing the Ansible Pull model, retrieving secrets, and preparing the VM environment

for automated configuration. The architecture seamlessly integrates with AWS services, delivering a scalable and secure self-configuring solution.

Environment Configuration:

- **Git Repository Setup:** The ansible repository contains playbooks and roles, while the ansible-inventory repository holds inventory configurations. The terraform and terraform-modules repositories organize infrastructure code.
- **Vault Setup:** Vault is configured to securely store secrets, ensuring EC2 instances can authenticate using AWS IAM roles [6].

Provisioning and Initialization:

- **Terraform Deployment:** Terraform creates EC2 instances and configures them using a cloud-init script.
- **Dependency Installation:** Dependencies such as Git, Python, Ansible, Ansible Collections, and Pip modules are installed.
- **Repository Cloning:** The ansible and ansible-inventory repositories are cloned onto the VM.
- **Environment Preparation:** The ansible-pull role prepares the environment by fetching the latest updates and applying the necessary configurations.

Self-Configuration:

- The ansible-pull command is executed with specific tags to configure the VM based on its purpose (e.g., database server).
- Following initial configuration, the Ansible Pull role sets up a systemd service to execute the ansible-pull command daily with “daily” tags, ensuring configurations remain up to date.

3.2 Use Cases

Database Server Configuration: using database-specific tags, Ansible Pull installs and configures database software, such as MySQL. Credentials and access permissions are securely retrieved from Vault, and monitoring and logging systems are initialized. Daily pulls ensure configurations remain intact, and security settings are consistently reapplied.

Application Server Configuration: application servers are configured to install runtime dependencies, deploy application code, and set environment variables using secrets from Vault. Daily execution of Ansible Pull ensures the application environment remains consistent and dependencies stay updated.

General Maintenance: a daily system service verifies critical configurations such as SSH settings and dependency versions, applies updates from the Ansible repository, and ensures the system maintains its desired state without manual intervention.

This architecture allows organizations to establish a fault-tolerant and automated environment for managing virtual machines, ensuring consistency, scalability, and security across their infrastructure.

3.3 Key metrics

Ansible Pull can be evaluated using several key metrics, including setup time, failure rate, and maintenance costs. The setup time for Ansible Pull is minimal compared to traditional push-based configuration management systems, as it enables nodes to autonomously fetch configurations without requiring centralized orchestration. The failure rate is typically low, provided proper error-handling mechanisms are included in the playbooks.

Ansible Pull avoids the network bottlenecks and scalability issues inherent in push models. Maintenance costs are reduced because nodes are self-sufficient, decreasing the need for continuous monitoring or administrator intervention. In contrast, Ansible Push requires a control node and is less efficient in large-scale environments, where connectivity issues can disrupt processes. Such a control node may lack centralization, complicating dependency unification and management.

Comparatively, tools like Puppet or Chef offer similar performance but often require more complex setup and incur ongoing licensing or support costs. Research comparing Ansible, Chef, SaltStack, and

Puppet in configuration management highlights metrics such as deployment speed, reliability, and ease of use.

Roman Kostromin conducted a detailed analysis of these tools in heterogeneous distributed computing environments, emphasizing that Ansible outperforms others in simplicity and agentless architecture, reducing setup complexity and human errors. While Chef and Puppet excel in state management and scalability, they demand higher configuration and maintenance effort [7]. Moreover, a 2022 study noted that Ansible's reliance on SSH infrastructure and its agentless nature make it a secure and efficient choice for configuration management, particularly for small to medium-sized deployments, while Chef and Puppet are more suitable for complex infrastructures with frequent configuration changes [8].

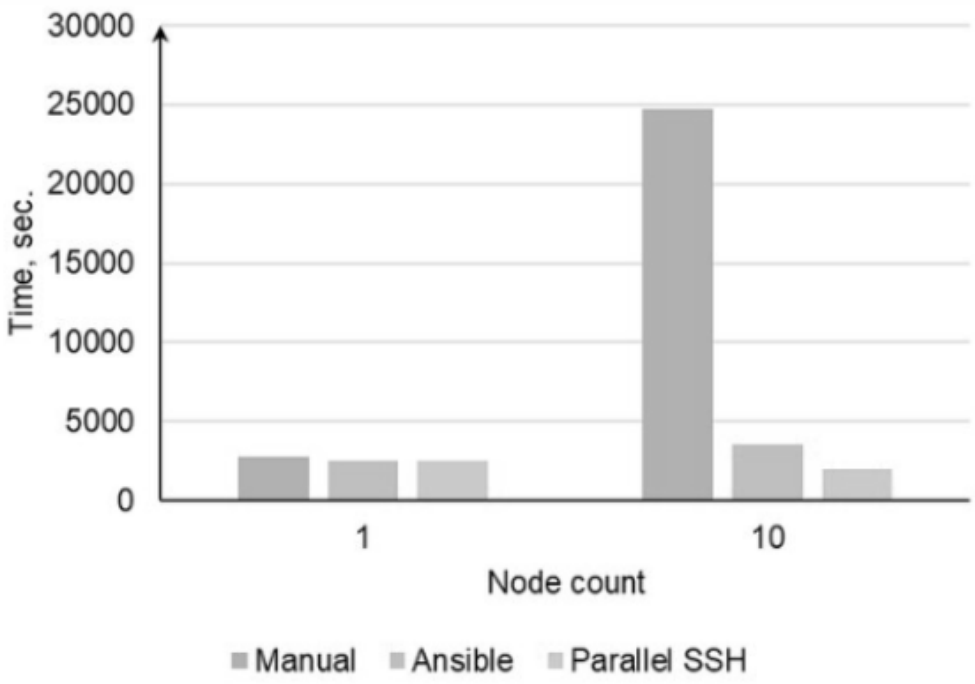


Fig.3.2. Comparison of Node Configuration Time: Manual Setup, Ansible, and Parallel SSH Connections [8]

Ansible Pull is a streamlined, autonomous solution for self-configuring virtual environments, particularly in scenarios where simplicity, scalability, and cost-efficiency are key priorities. Its decentralized model reduces dependence on a central control node, providing enhanced resilience and deployment flexibility.

REFERENCES

1. Introduction to ansible. Introduction to Ansible - Ansible Community Documentation. (2024, November 20). https://docs.ansible.com/ansible/latest/getting_started/introduction.html
2. Ansible concepts. Ansible concepts - Ansible Community Documentation. (2024, November 20). https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#id1
3. ansible-pull – Ansible Community Documentation. Ansible Documentation. URL: https://docs.ansible.com/ansible/latest/cli/ansible-pull.html (date of access: 08.12.2024)
4. Beyond Git: The other version control systems developers use - Stack Overflow. The Stack Overflow Blog - Stack Overflow. URL: https://stackoverflow.blog/2023/01/09/beyond-git-the-other-version-control-systems-developers-use/ (date of access: 15.12.2024).
5. cloud-init 24.4 documentation. cloud-init 24.4 documentation. URL: https://cloudinit.readthedocs.io/en/latest/ (date of access: 18.12.2024).
6. AWS - Auth Methods | Vault | HashiCorp Developer. AWS - Auth Methods | Vault | HashiCorp Developer. URL: https://developer.hashicorp.com/vault/docs/auth/aws (date of access: 18.12.2024).

7. Kostromin, R. O. (2020). Survey of software configuration management tools of nodes in heterogeneous distributed computing environment. The International Workshop on Information, Computation, and Control Systems for Distributed Environments. https://doi.org/10.47350/iccs-de.2020.15
8. S, L. (2022). Automation of server configuration using Ansible. International Journal for Research in Applied Science and Engineering Technology, 10(6), 4109–4113. https://doi.org/10.22214/ijraset.2022.44840

Dmitry Bulavin *PhD, associate professor*
Kharkiv National University named V. N. Karazin 61000
e-mail: d.bulavin@karazin.ua
ORCID: 0000-0002-4840-4763
Vladyslav Samoilenko *Student*
Kharkiv National University named V. N. Karazin 61000
e-mail: vladyslav.samoilenko@student.karazin.ua;
ORCID: 0009-0001-3217-4473

Self-Configuring Virtual Machine Environments Using Ansible Pull: A Review of Approaches and Challenges

Relevance. Automation of virtual machine (VM) configuration is a key component of modern IT infrastructure, ensuring scalability, reliability, and security. With the increasing adoption of cloud technologies, there is a growing demand for standardized approaches to infrastructure management that overcome the limitations of traditional methods such as manual configuration or standalone scripts.
Goal. To design an architecture for automating VM self-configuration based on Ansible Pull, integrated with tools like Terraform, Hashicorp Vault, and Git, to ensure flexibility, autonomy, and security.
Research methods. The proposed system utilizes an analytical approach to evaluate existing configuration tools, models architecture using Git repositories, Terraform, and Vault, and implements cloud-init to initialize Ansible Pull on VMs.
Results. The proposed architecture automates VM configuration processes, ensuring scalability and reducing dependency on centralized management servers. Ansible Pull implementation with tagging supports idempotent task execution, regular updates, and maintaining the desired state of the system.
Conclusions. Combining Ansible Pull with Terraform, Git, and Vault provides ease of implementation, scalability, and secure configuration handling. The approach is effective for dynamic environments requiring regular updates and aligns with modern DevOps practices.

Keywords: *Ansible Pull, configuration automation, Terraform, Vault, self-configuration, DevOps, Ansible*

Надійшла у першій редакції 22.08.2024, в останній - 05.10.2024.
The first version has been received on 22.08.2024, the final version - on 05.10.2024.

Наукове видання

Вісник Харківського національного університету імені В. Н. Каразіна

Серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління

Випуск 64

Збірник наукових праць

Українською та англійською мовами

Комп'ютерне верстання О.О. Афанасьєва

Підписано до друку 30.11.2024 р.
Формат 60х84/8. Папір офсетний. Друк цифровий.
Ум. друк. арк. – 11,34.
Обл.– вид. арк. – 9,07.
Наклад 50 пр. Зам. № 64/2024
Безкоштовно

Видавець і виготовлювач
Харківський національний університет імені В. Н. Каразіна
61022, м. Харків, майдан Свободи, 4
Свідоцтво суб'єкта видавничої справи ДК №3367 від 13.01.09

Видавництво Харківський національний університет імені В. Н. Каразіна
тел.: 705-24-32