UDC 004.056.55 **doi:** https://doi.org/10.20998/3083-6298.2025.01.04

Tetiana Laptieva[1], Oleksandr Laptiev[2]

[1] State University of Trade and Economics /
Kyiv National University of Trade and Economics, Kyiv, Ukraine
[2] Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

METHOD OF SYNTHESIS OF PROTECTED INFORMATION NETWORKS BASED ON THE INTRODUCTION OF CORRECTION LINES OF COMMUNICATION

Abstract. Topicality. Modern distributed software systems operate in environments characterized by continuous evolution of hardware platforms, heterogeneous computing resources, and dynamic failure conditions. Ensuring functional stability and rapid self-recovery in such environments is a critical challenge, especially under potential software or hardware failures, increased workloads, and external disruptions. **Objective:** This research aims to develop a novel method for synthesizing the structure of heterogeneous information networks based on the principle of maximizing the functional stability index. The proposed approach leverages the introduction of corrective communication links to enable self-recovery mechanisms that account for the heterogeneity and complexity of modern distributed systems. **Methods:** A mathematical model of a hypernetwork was developed using two interdependent hypergraphs to represent network components and their interactions. Based on this model, three algorithms (AI, AII, and AIII) were designed to optimize network connectivity, ensure vertex redundancy, and dynamically reconfigure the network in response to failures. These algorithms facilitate real-time fault detection, localization, and restoration through resource redistribution. **Results:** Simulation results demonstrate that the proposed method significantly improves system resilience and reduces recovery time. The maximum optimization effect was observed at a relative value of 0.869698, where the difference in the functional stability index reached 0.467232, confirming the effectiveness of the structural optimization process. **Conclusions:** The integration of corrective communication links into network topology enables enhanced robustness and fault tolerance. The adaptive algorithms developed within the hypernetwork framework support autonomous reconfiguration and self-healing, making them suitable for next-generation intelligent and resilient distributed systems.

Keywords: cybersecurity; self-recovery; functional stability; hypernetwork; corrective communication links; distributed systems; network resilience; fault tolerance; structural optimization.

Introduction

Problem relevance. Modern software systems are designed and operate under conditions of continuous evolution in hardware platforms and operating systems. Most of these systems have a distributed architecture, where software components are deployed across remote and heterogeneous computing resources. Over time, failures may inevitably occur — whether it be the malfunction of a particular software module, a computational resource, or communication equipment. Additionally, one or more existing services might fail to handle increased workloads effectively.

The operation of such systems often encounters challenges due to the growing number of interacting objects and the integration of diverse subsystems into a single information network, which is inherently heterogeneous. This heterogeneity introduces complexity in terms of compatibility, data exchange, and coordination among system components. Furthermore, in certain scenarios, there exists the possibility of external influences — such as cyberattacks, environmental disruptions, or human errors — that can negatively impact the reliability and availability of software components.

In all of these cases, the critical issue of restoring the functionality of the software system arises. Ensuring fault tolerance, rapid recovery, and consistent performance under unpredictable conditions becomes a key concern in the development and maintenance of modern distributed software systems. The problem of software recovery involves not only restoring failed components to a functional state but also preserving data integrity, maintaining service continuity, and minimizing downtime. Various approaches, including redundancy, checkpointing, rollback recovery, and self-healing mechanisms, are employed to address these issues and improve system resilience [1].

Literature review. Self-recovery systems can be categorized into three distinct levels depending on the type of resources being monitored and influenced: application-level software, system-level software, and hardware-level components [1–4].

Application-Level Self-Recovery refers to the ability of an individual application, software system, or service to restore its operational state from within, without external intervention. This level focuses on internal mechanisms embedded in the software itself, such as exception handling, automatic rollback to a stable state, or reinitialization of failed modules [3, 5, 6]. Application-level self-healing typically relies on predefined recovery procedures and fault detection algorithms that are integrated during the development phase.

System-Level Self-Recovery applies to all operating system-level services and applications, regardless of their internal structure or dependencies. It involves broader system management strategies aimed at maintaining overall system stability and availability. This type of self-recovery operates at the level of the entire computer system and includes actions such as restarting a failed process, reallocating memory resources, or switching between redundant components. A typical example is the automatic restart of a crashed process upon failure detection [4, 6]. System-level mechanisms often rely on monitoring tools and watchdog processes that continuously assess the health of system components.

When Hardware-Level Failures occur, self-recovery involves identifying functional nodes and redistributing software components accordingly, including the establishment of new network connections. Similar to system-level recovery, this requires continuous monitoring of various hardware components such as processors, memory units, storage devices, and communication interfaces. The response may include rerouting data through alternative paths, activating backup hardware, or migrating virtual machines to healthy hosts [4, 6]. Special attention to these issues has been given by researchers focusing on the property of functional resilience in complex technical systems capable of self-organization under destabilizing influences [7–9].

In [7], a general strategy for ensuring functional resilience in complex technical systems was proposed. This approach is based on the implementation of three principles introduced by Professor Oleg Mashkov:

Detection of abnormal situations;

Localization of the abnormal situation;

Restoration of network functionality through resource redistribution.

In [8], the authors analyzed self-organization mechanisms in heterogeneous information networks and proposed new indicators and criteria for identifying functionally resilient networks according to the Self-Organizing Network (SON) concept. They also developed methods for self-organizing information networks based on resource redistribution among remaining functional nodes. However, the issue of detecting faulty nodes was not addressed in this work.

Work [9] focuses on addressing the problem of ensuring functional resilience using Petri nets. Papers [10–11] examine network technologies for visual information transmission and the development of task distribution information technology for grid systems using the GRASS simulation environment. The methods described in these works also employ network restructuring mechanisms during anomalous overloads. Nevertheless, the issue of introducing redundancy — particularly determining optimal redundancy levels — remains open and insufficiently explored.

Problem Statement. In software systems, the term "self-recovery" implies that any application, service, or system is capable of detecting that it is not functioning properly and, without human intervention, making the necessary adjustments to restore itself to a normal or desired operational state [1, 2]. This capability can also be applied in the event of potential failures of the system's structural components [3]. Self-recovery involves equipping the system with the ability to make autonomous decisions by continuously monitoring and optimizing its internal state, as well as automatically adapting to changing environmental conditions.

The core challenge lies in developing a functionally resilient system — one that is capable of responding to both software- and hardware-related changes and either self-recovering after failures or executing appropriate preventive actions when failure is anticipated. Achieving functional resilience typically involves incorporating various forms of redundancy: structural, hardware-based, software-based, and temporal. To this end, an optimal network structure (in terms of maximizing functional resilience) is determined, characterized by redundant communication links aimed at establishing alternative transmission routes.

Research Objective. The objective of this research is to develop a method for synthesizing the structure of heterogeneous information networks through the introduction of corrective communication links. This method would enable the design of self-recovery mechanisms for information systems while accounting for the heterogeneity of network components. By implementing such an approach, it becomes possible to engineer functionally resilient information systems that significantly reduce recovery time following probable system failures. The proposed methodology aims to enhance fault tolerance and improve overall system availability under dynamic and potentially unstable operating conditions.

1. Model of a Heterogeneous Information Network

The mathematical model that enables the description of network processes — specifically, processes of reconfiguration, self-recovery, and self-adjustment under conditions of heterogeneous computing resources — is the hypernetwork.

Formally, an abstract hypernetwork can be described by a sextuple AS = (X, V, R; P, F, W), which includes the following components:

$X = (x_1, x_2, \dots, x_n)$ – a set of nodes (vertices);

$V = (v_1, v_2, ..., v\_g)$ – a set of branches;

$R = (r_1, r_2, ..., r\_m)$ – a set of edges;

P: V → 2^X – a mapping that assigns to each element $v \in V$ a subset $P(v) \subseteq X$ of its vertices. This mapping defines a hypergraph PS = (X, V; P) ;

F: R → 2^V – a mapping that assigns to each element $r \in R$ a subset $F(r) \subseteq V$ of its branches, such that the family of branch subsets contains only those whose branches form connected parts of the hypergraph PS.

This mapping defines another hypergraph FS = (V, R; F) ;W: R → 2^{P(F(r))}, for all $r \in R$ – a mapping that associates with each element $r \in R$ a subset $W(r) \subseteq P(F(r))$ of its vertices, where P(F(r)) is the set of vertices in PS incident to the branches $F(r) \subseteq V$. Thus, this mapping defines yet another hypergraph WS = (X, R; W) Assume two hypergraphs PS = (X, V; P) and WS = (Y, R; W) are given. Then the mapping Φ defines an abstract hypernetwork AS = (PS, WS; Φ) if both PS and WS form connected components of the hypergraph PS.

From this definition, it follows that different abstract hypernetworks may correspond to a pair of hypergraphs PS and WS. In what follows, we will use various notations for hypernetworks wherever convenient.

In Fig. 1, there is no pair of vertex-independent paths between nodes x_1 and x_5, yet these nodes are externally 2-connected. That means the external removal of any pair of nodes from the set $\{x_2, x_3, x_4\}$ disrupts all paths connecting x_1 and x_5.

The connectivity ω = ω(S) of a hypernetwork S is defined as the smallest number of nodes whose removal leads to a disconnected hypernetwork (as shown in Fig. 1 The ω-connectivity of a hypernetwork S cannot be computed using known methods of classical graph theory.

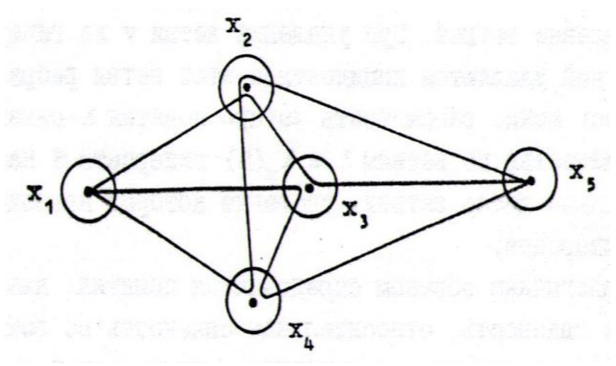

Fig. 1. Hypernet model

Network Structure Synthesis Method Based on the Criterion of Maximum Functional Stability Index Using Corrective Communication Links.

Let the graphs of the primary network PS = (X, Y) and the secondary network WS = (Y, R) be given. We denote:

- p(v) or l_ij – length of branch v_ij ;
- pl(r) or l'_ij – length of edge r_ij ;
- c(v) – capacity (throughput) of branch v_ij ;
- c(r) – capacity of edge r_ij ;
- ω(S) – vertex connectivity of the hypernetwork S .

The objective is to construct a hypernetwork S = (PS; WS; Φ) that satisfies the following conditions:

$$k \le \omega(S) \to \max,$$

$$\forall v \in V: \sum_{r \in \Phi^{-1}} \partial(r) \le \partial(v)$$

and minimizes a certain functional φ(S) — representing the cost of designing and operating the network S .

The solution to this problem is implemented via Algorithm AI , which performs a search for a feasible hypernetwork satisfying the previously described criterion of maximum functional stability.

I. Algorithm AI

Step 1. If μ_PS(xi, xj) ≥ μ_WS(xi, xj) and ω(S) = k·ω(PS) , proceed to Step 2; otherwise, go to Step 15.

Step 2. For all pairs of nodes xi, xj , compute the maximum flows in graphs PS and WS , i.e., calculate μ_PS(xi, xj) and μ_WS(xi, xj) . If there exists a pair x_i, x_j such that μ_PS(xi, xj) < μ_WS(xi, xj) , then go to Step 15; otherwise, proceed to Step 3.

Step 3. Implement edges of graph WS along the shortest path in graph PS . If condition (1) is satisfied, go to Step 8; otherwise, proceed to Step 4.

Step 4. Among all branches v_lt , select those with the maximum value of $\Delta_{lt} = \sum_{r \in \Phi^{-1}(v_{lt})} \partial(r) / \partial(v_{lt})$.

If for certain branches Δlt > 1 , they are considered overloaded.

Step 5. From the overloaded branches, choose the one with the maximum Δ_lt , and find the edge r' with minimal l(r') . For each branch v_ij , compute the value δ_ij using the formula:

$$\delta_{ij} = \begin{cases} \Delta_{ij}, \text{ якщо } v_{ij} \notin \Phi(r) \text{ і } \sum\limits_{r_{\alpha\beta} \in \Phi^{-1}(v_{ij})} (v_{ij}) \partial(r_{\alpha\beta}) \le \partial(v_{ij}) - \partial(r); \\ 1, \textit{ якщо } v_{ij} \notin \Phi(r) \text{ і } \sum\limits_{r_{\alpha\beta} \in \Phi^{-1}(v_{ij})} \partial(r_{\alpha\beta}) > \partial(v_{ij}) - \partial(r); \\ \Delta_{ij} - \dfrac{\partial(r)}{\partial(v_{ij})}, \textit{якщо } v_{ij} \in \Phi(r) \textit{ і гілка } v_{ij} \textit{ не є перенасиченою}; \\ 1, \textit{ якщо } v_{ij} \in \Phi(r) \textit{ і гілка } v_{ij} \textit{ перенасичена.} \end{cases}$$

Also, define *ρ (v_ij) = ρ(v_ij)/(1 - δ_ij) . If δ_ij = 1 , obviously ρ (v_ij) → ∞* .

Step 6. Find the shortest path between nodes x, y in graph PS with branch weights *ρ (v)* *. If this route has finite weight, proceed to Step 7; otherwise, go to Step 15.

Step 7. Re-route edge r along the shortest path found at Step 6. Recalculate all Δ_lt values using the appropriate formula.

$$\Delta_{lt} = \sum_{r_{\alpha\beta} \in \Phi^{-1}(v_{lt})} \frac{\partial(r)}{\partial(v_{lt})}.$$

If no overloaded branches remain in the resulting hypernetwork, proceed to Step 8; otherwise, return to Step 5.

Step 8. If condition (2) is met, go to Step 14; otherwise, proceed to Step 9.

Step 9. Identify the minimum cut $\{x_1, ..., x\_p\}$, where p < k , in the hypernetwork by nodes.

Step 10. Find all edges r whose routes include nodes from the cut $\{x_1, ..., x\_p\}$, and whose endpoints lie in different connected components S_1 and S_2 of the hypernetwork S \ $\{x_1, ..., xp\}$. At least one such edge must exist since WS is k-connected . Select among them the edge r = (x, y) with the highest value of c(r) .

Step 11. For each branch v_ij , compute δ_ij using the same formula as before

$$\delta_{ij} = \begin{cases} 1, \textit{ if } v_{ij} \notin \Phi(r) \text{ і } \sum\limits_{\text{r}' \in \Phi^{-1}(v_{ij})} \partial(\text{r}') > \partial(v_{ij}) - \partial(r); \\ \dfrac{1}{\partial(v_{ij})} \sum\limits_{\text{r}' \in \Phi^{-1}(v_{ij})} \partial(\text{r}'), \text{ in all other cases.} \end{cases}$$

Define *ρ (v_ij) = ρ(v_ij)/(1 - δ_ij) . If δ_ij = 1 , then ρ (v_ij) → ∞* .

Step 12. Find the shortest route between nodes x, y in graph PS with branch weights *ρ (v)* *. If this route has finite weight, proceed to Step 13; otherwise, go to Step 15.

Step 13. Re-route edge r along the shortest path found at Step 12. If all edges whose routes include nodes from the cut {x₁, ..., x_p} have been processed, and their endpoints lie in different connected components S₁ and S₂ , return to Step 9; otherwise, go to Step 10. Repeat Steps 9–13 until a hypernetwork satisfying conditions (1) and (2) is found, or until the minimum cut repeats. In the former case, go to Step 14; in the latter, go to Step 15.

Step 14. A hypernetwork satisfying the maximum functional stability criterion (1)–(2) has been found.

Step 15. No feasible solution was found.

In this work, a method for synthesizing a hypernetwork S , for which ω(S) = k·ω(PS) , is also considered. This can be achieved by introducing corrective edges (communication links) into the structure of graph WS .

Let the primary network PS = (X, V) and secondary network WS = (Y, R) be given. The objective is to find a hypernetwork S = (PS, WS', Φ) such that ω(S) = k·ω(PS) and condition (2) is satisfied, where WS' = (X, R ∪ U) and U is the set of corrective edges. Moreover, when adding a new corrective edge, the capacities of the edges in WS' are recalculated accordingly.

2. Algorithm AII

Step 1. Execute Algorithm AI. If the resulting hypernetwork S̃ satisfies condition (2) and ω(S̃) = k·ω(PS) , proceed to Step 5; otherwise, go to Step 2.

Step 2. If S̃ does not satisfy condition (2), proceed to Step 4; otherwise, proceed to Step 3.

Step 3. In S̃ , find the minimum cut {x₁, ..., x_p} . Identify connected components S₁ and S₂ in S̃ \ {x₁, ..., x_p} . Find in S₁ and S₂ the closest non-adjacent nodes x ∈ S₁ and y ∈ S₂ . Connect them with an edge. Recalculate edge capacities using the specified capacity calculation procedure. Return to Step 1.

Step 4. No feasible solution exists.

Step 5. The desired hypernetwork S̃ has been found.

The convergence of the algorithm is evident. The absence of a feasible solution is determined solely by condition (2).

Let the primary network PS = (X, V) and secondary network WS = (Y, R) be given. We seek a hypernetwork S = (PS, WS'; Φ) with maximum possible connectivity. The weight of each branch and node is equal to 1.

3. Algorithm AIII

Step 1. Sort the edges r ∈ R in descending order of distance d(r) in WS . Obtain a list of edges r₁, r₂, ..., r_m

Step 2. Implement edge r₁ in PS along the shortest path (shortest according to total node and edge weights).

Step 3. Increase the weights of nodes and edges along the route of edge r₁ by a certain amount Δ , which depends on the current state of the hypernetwork S .

Step 4. If all edges r₁, ..., r_m have been implemented (i.e., if i = m), proceed to Step 10; otherwise, increment i and return to Step 2.

Step 5. Find the minimum cut over vertices {x₁, ..., x_p} .

Step 6. Find all edges r weakly incident to nodes in {x₁, ..., x_p} , whose endpoints lie in different connected components S₁ and S₂ of the hypernetwork S \ {x₁, ..., x_p} . At least one such edge must exist because WS is k-connected .

Step 7. In the primary network PS , find the shortest paths between all pairs of terminal nodes x ∈ S₁, y ∈ S₂ . A route will certainly be found since PS is connected.

Step 8. Choose the pair x, y for which the following difference is minimal:

$$\rho_{PS\setminus\{x_1\}}(x_i, x_j) - \rho_{PS}(x_i, x_j)\,.$$

Step 9. Re-route edge r along the new path; this increases the connectivity between components S₁ and S₂ . Repeat Steps 3–6 until connectivity increases, then proceed to Step 10.

Step 10. A hypernetwork with quasi-maximal connectivity has been found. The adaptive shortest-path method used in the algorithm allows synthesizing an initial hypernetwork structure S with maximal connectivity by balancing the number of weakly incident edges across all nodes and branches.

On Figures 2 and 3 , unsynthesized and synthesized network structures with 12 switching nodes and 14 communication lines are shown. Their corresponding connectivity polynomials are:

$$P_{14}^{(1)}(p) = p^{14} + 6p^{13}(1-p) + 15p^{12}(1-p)^2 + 16p^{11}(1-p)^3,$$
$$P_{14}^{(2)}(p) = p^{14} + 14p^{13}(1-p) + 81p^{12}(1-p)^2 + 200p^{11}(1-p)^3,$$

Their graphs are shown in Figure 4 .

The maximum optimization effect is achieved at point x = 0.869698 , where the difference in the functional stability index reaches 0.467232 . This fully confirms the necessity of optimizing network structures.

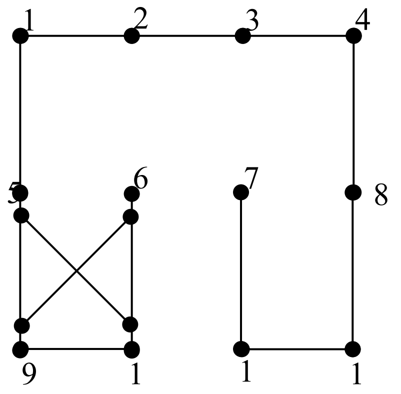

Fig. 2. Minimally functionally stable G (12, 14)-graph

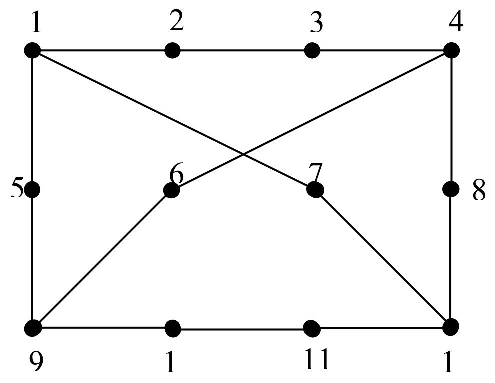

Fig. 3. Maximally functionally stable G (12, 14)-graph

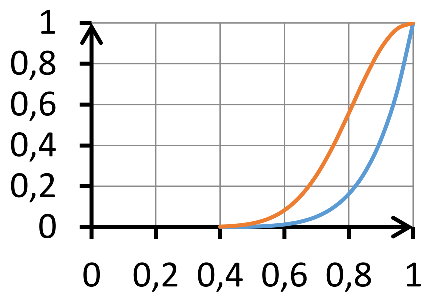

Fig. 4. Graphs of the connectivity polynomials of the graphs from Fig. 2 and Fig. 3

Schedule of experimental research on the method of analysis and synthesis of functional stability of branched information networks.

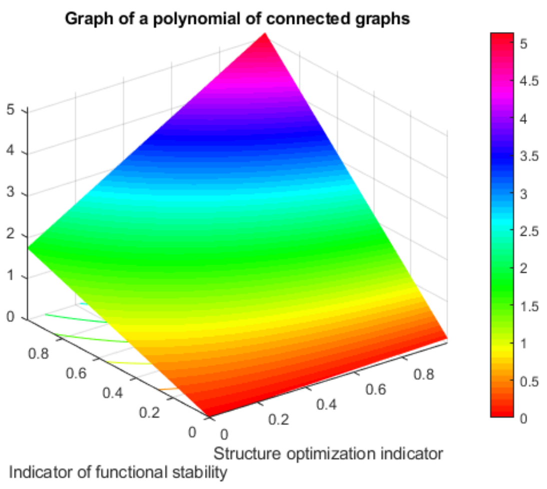

Fig. 5. Graph of connectivity polynomials of graphs from Fig. 2

Conclusions

In this work, a novel method for synthesizing network structures based on the criterion of maximizing the functional stability index has been proposed. This approach is grounded in the introduction of corrective communication links, enabling the design of self-recovery mechanisms for distributed software systems while taking into account the characteristics of heterogeneous computing resources. By leveraging redundancy at both structural and functional levels, the proposed method allows for the development of resilient software architectures capable of autonomously adapting to failures and maintaining operational integrity.

The core idea of the method lies in dynamically enhancing network connectivity through strategic addition of corrective edges, which serve as alternative communication pathways. This not only increases fault tolerance but also significantly reduces recovery time following probable system failures or under conditions of partial resource degradation. The ability to restore functionality without human intervention is particularly valuable in large-scale distributed environments where manual oversight is impractical or impossible.

To enhance the efficiency and applicability of the proposed synthesis method, the mathematical model of a hypernetwork was refined based on two interdependent hypergraphs. This advanced modeling framework enables the representation of complex interactions between different types of network components — including nodes, branches, and edges — and facilitates the precise evaluation of network resilience under various failure scenarios. It also supports the incorporation of diverse quality-of-service requirements, making it suitable for heterogeneous environments with varying performance constraints.

One of the key outcomes of the simulation conducted using the proposed model was the identification of an optimal point in the network structure optimization process. Specifically, the maximum positive effect from structural optimization was observed at the relative value of 0.869698 , where the difference in the functional stability index reached 0.467232 relative units. This result strongly confirms the validity and effectiveness of the developed method.

These findings demonstrate that the integration of corrective communication links into the network topology can significantly improve the robustness and reliability of distributed software systems. Furthermore, the use of adaptive algorithms within the hypernetwork model allows for real-time reconfiguration and self-healing, which are essential features for next-generation autonomous and intelligent systems.

In summary, the proposed method provides a solid theoretical and practical foundation for designing functionally stable software systems. It contributes to the advancement of self-managing network architectures and offers a promising solution for improving the availability and dependability of modern distributed computing environments.

References

1. Mani Kiran, C.V.N.S., Jagadeesh Babu, B. and Singh, M.K. (2023), "Study of Different Types of Smart Sensors for IoT Application Sensors", *Smart Innovation, Systems and Technologies*, vol. 290, pp. 101–107, doi: https://doi.org/10.1007/978-981-19-0108-9_11
2. Kuchuk, N., Kashkevich, S., Radchenko, V., Andrusenko, Y. and Kuchuk, H. (2024), "Applying edge computing in the execution IoT operative transactions", *Advanced Information Systems*, vol. 8, no. 4, pp. 49–59, doi: https://doi.org/10.20998/2522-9052.2024.4.07
3. Schneider, C., Barker, A., & Dobson, S. (2015), "A survey of self- healing systems frameworks". *Software: Practice and Experience*, 45(10), pp. 1375-1398.
4. Manzoor A., Rajput U, Phulpoto N, Abbas F, Rajput M. (2018), "Self-healing in Operating Systems", *IJCSNS International Journal of Computer Science and Network Security*, Vol.18 No.5, pp.92-98.
5. Wang, Z., & Wang, J. (2015), "Self-healing resilient distribution systems based on sectionalization into microgrids", *IEEE Transactions on Power Systems*, Vol. 30(6), pp.3139-3149.
6. Duarte, DP., Guaraldo, JC., Kagan, H., Nakata, BH., Pranskevicius, PC., Suematsu, AK., & Hoshina, MS. (2016), "Substation-based self-healing system with advanced features for control and monitoring of distribution systems. In Harmonics and Quality of Power (ICHQP)", *17th International Conference on 2016*, October, IEEE, pp. 301-305.
7. Ansari, B., Simoes, MG., Soroudi, A., & Keane A. (2016), "Restoration strategy in a self-healing distribution network with DG and flexible loads. In Environment and Electrical Engineering (EEEIC)", *IEEE 16th International Conference 2016*, June, IEEE, pp. 1-5.
8. Barabash O., Sobchuk V., Lukova-Chuiko N. and Musienko A. (2018), "Application of Petri Networks for Support of Functional Stability of Information Systems", *IEEE First International Conference on System Analysis & Intelligent Computing (SAIC*). 08-12 October, Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, Ukraine, pp. 36 – 39.
9. Bielievtsov, S., Ruban, I., Smelyakov, K., Sumtsov, D. (2018), "Network technology for transmission of visual information", *Selected Papers of the XVIII International Scientific and Practical Conference on Information Technologies and Security*, CEUR Workshop Processing, Kyiv, Ukraine, pp. 160-175.
10. Filimonchuk T., Volk M., Ruban I., Tkachov V. (2016), "Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment", *Eastern-European Journal of Enterprise Technologies. Information and controlling system*, Vol. 3/9 (81), pp. 45–53.
11. Li, G., Liu, Y., Wu, J., Lin, D. and Zhao, Sh. (2019), "Methods of Resource Scheduling Based on Optimized Fuzzy Clustering in Fog Computing", *Sensors*, vol. 19(9), doi: https://doi.org/10.3390/s19092122.
12. Deng, R., Lu, R., Lai, C., Luan, T. H. and Liang, H. (2016), "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption", *IEEE Internet Things*, Vol. 3, no. 6, pp. 1171–1181, doi: https://doi.org/10.1109/JIOT.2016.2565516.
13. Sobchuk V., Breslavsky V., Laptiev S., Laptieva T., Zahynei A., Kovalenko O. (2021), "Development of routing algoritm for self-organizing information networks", *German International Journal of Modern Science,* №7, Vol. 2, pp. 32–35.
14. Sobchuk V., Zamrii I., Sobchuk A., Laptiev S., Laptieva T., Samosyuk V. (2021), "Method of Data Processing in Information Systems Using Solutions of Differential Equations with Impulse Effect", *International Journal of Science and Engineering Investigations*, Denmark, Vol. 10, Issue 11. pp. 1–6.
15. Laptiev S., Laptieva T., Hrebennikov A., Kitura O., Marchenko V., Lutsenko M. (2021), "Advanced model of the protection system of insider informations", *Journal of science*, Lyon, Vol.1. №20, pp.39–44.
16. Sobchuk V., Laptiev S., Laptieva T., Barabash O., Drobyk O., Sobchuk A. (2024), "A modified method of spectral analysis of radio signals using the operator approach for the fourier transform", *IT, Automation, Measurements in Economy and Environmental Protection,* Vol. 14, No 2, pp.56–61, https://doi.org/10.35784/iapgos.5783.
17. Milevskyi S., Korol O., Hryschuk O., Laptieva T., Yevseiev S. (2024), "Crypto-code constructions on LDPC codes properties assessment", *Ukrainian Scientific Journal of Information Security*, vol. 30, issue 2, pp. 313-323, https://doi.org/: 10.18372/2225-5036.30.19244.
18. Svynchuk O., Barabash A., Laptiev S. and Laptieva T. (2021), "Modification of query processing methods in distributed databases using fractal trees", *1 International Scientific and Practical Conference "Information Security And Information Technologies": Conference Proceedings,* Kharkiv – Odesa, Ukraine. pp.32–37.

Received (Надійшла) 17.04.2024
Accepted for publication (Прийнята до друку) 07.05.2025

Відомості про авторів/ About the Authors

Лаптєва Тетяна Олександрівна – доктор філософії з кібербезпеки, Державний торговельно-економічний університет / Київський національний торговельно-економічний університет.Київ, Україна
Tetiana Laptieva – PhD in Cybersecurity, State University of Trade and Economics /
Kyiv National University of Trade and Economics,Kyiv, Ukraine
e-mail: Tetiana1986@ukr.net ;ORCID: https://orcid.org/0000-0002-5223-9078;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57468969100
Лаптєв Олександр Анатолійович – доктор технічних наук, старший науковий співробітник, доцент кафедри кібербезпеки та захисту інформації, Київський національний університет імені Тараса Шевченка, Київ,Україна

Oleksandr Laptiev – Dr. Sci. (Technical), Senior Researcher, Associate Professor the Department of Cyber Security and Information Protection, Taras Shevchenko National University of Kyiv
e-mail: Alaptev64@ukr.net ;ORCID https://orcid.org/0000-0002-4194-402X;
Scopus https://www.scopus.com/authid/detail.uri?authorId=57216163849

МЕТОД СИНТЕЗУ ЗАХИЩЕНИХ ІНФОРМАЦІЙНИХ МЕРЕЖ НА ОСНОВІ ВВЕДЕННЯ КОРИГУВАЛЬНИХ ЛІНІЙ ЗВ'ЯЗКУ

Т.О. Лаптєва, О.А. Лаптєв

Анотація. Актуальність. Сучасні розподілені програмні системи функціонують у середовищах, що характеризуються постійним розвитком апаратних платформ, неоднорідних обчислювальних ресурсах і динамічних умовах відмов. Забезпечення функціональної стійкості та швидкого самовідновлення в таких умовах є критично важливою задачею, особливо за наявності потенційних відмов програмного забезпечення або апаратних засобів, підвищеного навантаження чи зовнішніх впливів. **Мета дослідження.** Розробка нового методу синтезу структури гетерогенних інформаційних мереж на основі принципу максимізації показника функціональної стійкості. Пропонований підхід ґрунтується на введенні коригувальних ліній зв'язку, що дає можливість створювати механізми самовідновлення для розподілених програмних систем з урахуванням їхньої гетерогенності та складності. **Методи.** Для представлення компонентів мережі та їх взаємодії було розроблено математичну модель гіпермережі на основі двох взаємозв'язаних гіперграфів. На базі цієї моделі запропоновано три алгоритми (AI, AII та AIII), які призначені для оптимізації зв'язності мережі, забезпечення вершинної надмірності та динамічної реконфігурації мережі в умовах відмов. Ці алгоритми реалізують автоматичне виявлення несправностей, їхню локалізацію та відновлення шляхом перерозподілу ресурсів. **Результати.** Результати моделювання продемонстрували, що пропонований метод значно підвищує стійкість системи та скорочує час відновлення. Максимальний ефект оптимізації досягнуто при відносному значенні 0.869698, де різниця значень показника функціональної стійкості становила 0.467232 відносних одиниць, що підтверджує ефективність запропонованого методу. **Висновки.** Введення коригувальних ліній зв'язку в топологію мережі дозволяє значно підвищити її надійність і завадостійкість. Адаптивні алгоритми, розроблені в рамках моделі гіпермережі, забезпечують автономну реконфігурацію та самовідновлення, що робить їх придатними для наступного покоління інтелектуальних і стійких розподілених систем.

Ключові слова: кібербезпека; самовідновлення; функціональна стійкість; гіпермережа; коригувальні лінії зв'язку; розподілені системи; стійкість мережі; надійність; оптимізація структури.