**ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ**
**INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS**

УДК 519.872.7

DOI: 10.21822/2073-6185-2026-53-1-186-192 Оригинальная статья/ Original article

# Математическая модель микросервисной системы на основе байесовской сети для задачи оптимизации конфигурации с учётом каскадных зависимостей

**В.Р. Четвертухин**
Белгородский государственный технологический университет имени В. Г. Шухова,
308012, г. Белгород, ул. Костюкова, 46, Россия

**Резюме. Цель.** Целью работы является разработка математической модели микросервисной системы для оптимизации конфигурации этой системы. **Метод.** Для моделирования использовались методы теории графов, теории вероятностей и теории принятия решений, такие вероятностно-графические модели, как байесовские сети, сочетающие элементы этих теорий. **Результат.** Описана математическая модель микросервисной системы на основе байесовской сети; предложен модифицированный алгоритм вывода в этой сети и алгоритм оптимизации конфигурации с учётом каскадов. Оценена временная сложность вывода оптимальной конфигурации. Проведён эксперимент применения предложенных алгоритмов на модельной системе из 15 микросервисов, который показал, что конфигурация, выбираемая с помощью предложенной модели и алгоритмов, имеет наименьший процент времени, когда микросервисная система находилась в состоянии нарушения параметров соглашения об уровне обслуживания (Service Level Agreement, SLA). **Вывод.** Предложенная модель и алгоритмы могут помочь в выборе более оптимальной конфигурации благодаря учёту каскадных эффектов и одновременному учёту дискретных и непрерывных параметров. Для дополнительных выводов необходимо провести эксперимент по интеграции модели с существующими системами оркестрации.

**Ключевые слова:** микросервисная архитектура, байесовские сети, каскадные зависимости, вероятностный вывод, оптимизация конфигурации

**Для цитирования:** В.Р. Четвертухин. Математическая модель микросервисной системы на основе байесовской сети для задачи оптимизации конфигурации с учётом каскадных зависимостей. Вестник Дагестанского государственного технического университета. Технические науки. 2026;53(1):186-192. DOI:10.21822/2073-6185-2026-53-1-186-192.

# Mathematical model of a microservice system based on a Bayesian network for the task of optimizing configuration taking into account cascading dependencies

**V. R. Chetvertukhin**
V. G. Shukhov Belgorod State Technological University,
46 Kostyukova St., Belgorod 308012, Russia

**Abstract. Objective.** The aim of the work is to develop a mathematical model of a microservice system to optimize the configuration of this system. **Method.** Methods of graph theory, probability theory, and decision theory were used for modeling, namely probabilistic graphical models such as Bayesian networks that combine elements of all these theories. **Result.** A mathematical model of a microservice system based on a Bayesian network is described. A modified inference algorithm for this network and a configuration optimization algorithm taking cascades into account are proposed. The time complexity of inferring the optimal configuration is estimated. An experiment applying the proposed algorithms to a model system of 15 microservices was conducted, demonstrating that the configuration selected using the proposed model and algorithms has the lowest percentage of time the microservice system was in a state of service level agreement violation. **Conclusion.** The model and algorithms will help in selecting the optimal configuration

by taking into account cascading effects, discrete and continuous parameters. Further conclusions should be drawn by integrating the model with existing orchestration systems.

**Keywords:** microservice architecture, Bayesian networks, cascade dependencies, probabilistic inference, configuration optimization

**For citation:** V.R. Chetvertukhin. Mathematical model of a microservice system based on a Bayesian network for the task of optimizing configuration taking into account cascading dependencies. Herald of Daghestan State Technical University. Technical Sciences. 2026;53(1):186-192. (In Russ) DOI:10.21822/2073-6185-2026-53-1-186-192.

**Введение.** В задачах построения распределённых масштабируемых систем всё чаще отдаётся предпочтение микросервисной архитектуре, так как монолитные решения не способны отвечать вызову роста сложности таких систем. Поэтому в последние годы микросервисная архитектура стала доминирующим подходом проектирования [1].

Однако декомпозиция монолитного программного продукта на множество связанных между собой микросервисов порождает новый класс задач управления. Эти задачи связаны с оптимизацией конфигурации микросервисной системы, так как такая система может состоять из десятков или даже сотен взаимодействующих друг с другом компонентов, каждый из которых может иметь собственные требования к ресурсам и характеристики производительности. Классические методы теории массового обслуживания [2] и теории управления [3] разработаны для систем с фиксированной структурой и количественными параметрами. То есть, параметры, независимо от их природы, сводятся к некоторой количественной шкале оценивания.

Конфигурирование микросервисных систем требует одновременного учёта параметров, принимающих значения из конечного множества, не сводимых к какой-либо количественной шкале (например, сервер для обработки запросов в задаче балансировки нагрузки, может принимать значение из конечного множества серверов), так и количественных параметров (например, количество памяти, размер кластера, количество реплик). Необходимо учитывать тот факт, что между сервисами могут существовать сложные зависимости: перегрузка одного компонента может вызвать каскадный эффект, способный привести к деградации всей системы [4]. Применение методов искусственного интеллекта (ИИ), в частности, обучения с подкреплением [5], может быть перспективным направлением в задаче оптимального выбора конфигурации микросервисной системы. Однако такие методы, как и большинство ИИ-решений, страдают проблемой «чёрного ящика»: решения, принимаемые интеллектуальной моделью, сложно не только предсказать, но и интерпретировать. Во многих работах (например, в [6]) предлагается и реализуется на практике использование вероятностных графических моделей, в частности байесовских сетей, для описания зависимостей сложных систем. Однако в существующих моделях значения величин, ассоциированных с вершинами графов байесовских сетей, либо дискретны, либо предполагают использование гауссовские распределения для непрерывных величин [7], что недостаточно для описания модели микросервисной системы.

**Постановка задачи.** Целью исследования является разработка модели байесовской сети, способной одновременно учитывать дискретные и количественные аспекты микросервисных систем, а также сопроводить модель алгоритмом оптимизации, учитывающим каскадные эффекты.

**Методы исследования.** Математически микросервисную систему можно представить в виде кортежа:

$$\mathcal{M} = \langle S, D, R, L, Q \rangle$$

Где, $S = \{s_1, \dots, s_n\}$ – множество сервисов; $D \subseteq S \times S$ – граф зависимостей между сервисами; $R: S \to P(R^+)$ – функция требований к ресурсам; $L_i: S \times R^+ \to R^+$ – функция нагрузки; $Q: S \to R^+$ – требования к качеству обслуживания (SLA).

*Конфигурацией* сервиса $c_i$ будем считать пару:

$$c_i = (d_i, r_i)$$

Где, $d_i \in \mathcal{D}_i$ – набор значений дискретных параметров сервиса; $r_i \in R_+^{k_i}$ – набор количественных параметров сервиса.

*Состоянием* сервиса в момент времени *t* будем считать набор параметров:

$$x_i(t) = \big(p_i(t), \tau_i(t), e_i(t)\big)$$

Где, $p_i(t)$ – производительность (количество запросов в секунду); $\tau_i(t)$ – латентность (мс); $e_i(t)$ – частота ошибок.

Опишем модель байесовской сети для микросервисной системы. Сама по себе байесовская сеть представляет собой граф вида:

$$\mathcal{B} = (V, E, P)$$

Где, $V$ – множество вершин, $E$ – множество дуг, и $P$ – вероятностное распределение над множеством вершин.

Множество вершин $V$ в нашем случае представляет собой объединение следующих множеств:

$$V = V_D \cup V_R \cup V_X \cup V_L$$

Где, $V_D = \{D_1, \dots, D_n\}$ – множество вершин, соответствующих дискретным параметрам конфигурации; $V_R = \{R_1, \dots, R_n\}$ – множество вершин, соответствующих количественным параметрам ресурсов; $V_X = \{X_1, \dots, X_n\}$ – множество вершин, соответствующих состояниям сервисов, и $V_L = \{L_1, \dots, L_n\}$ – множество вершин, соответствующих входящей нагрузке на систему.

Теперь для каждого сервиса $s_i$ определим структуру входящих в него зависимостей:

$$Parents(X_i) = \{D_i, R_i, L_i\} \cup \big\{X_j : (s_j, s_i) \in D\big\}$$

Это означает, что состояние сервиса зависит от его конфигурации, выделенных ресурсов, входящей нагрузки и состояния сервисов, от которых он зависит. Опишем одну из ключевых особенностей модели, а именно связь дискретных и количественных переменных. Для дискретных параметров конфигурации вероятностное распределение будет выглядеть следующим образом:

$$P(D_i = d \mid Parents(D_i)) = \frac{exp\left(\theta_i^T \phi\big(d, Parents(D_i)\big)\right)}{\sum_{d' \in \mathcal{D}_i} exp\left(\theta_i^T \phi\big(d', Parents(D_i)\big)\right)}$$

Где, $\phi$ – функция признаков, генерирующая характеристики дискретного значения, связывающие его с тем или иным родителем, $\theta_i$ – вектор параметров (веса для признаков).

Опишем вероятностное распределение количественных ресурсов, связывающее эти ресурсы с дискретными параметрами конфигурации:

$$P(R_i \mid D_i = d) = \mathcal{N}\left(\mu_r(d), \sum\nolimits_r (d)\right)$$

Где, $\mathcal{N}$ – нормальное распределение, $\mu$ – математическое ожидание.

Опишем кусочно-определённое вероятностное распределение для состояния микросервиса:

$$P\big(X_i \mid Parents(X_i)\big) = \begin{cases} \mathcal{N}(\mu_{normal}, \sigma^2_{normal}), L_i < \gamma \cdot p_i^{max} \\ Beta\big(\alpha_{degraded}, \beta_{degraded}\big), L_i \geq \gamma \cdot p_i^{max} \end{cases}$$

Где, Beta – бета-распределение, $p_i^{max}$ – максимальная производительность при заданной конфигурации, $\gamma$ – порог перегрузки, принимающий значение от 0 до 1, определяющий, при какой доле от максимальной производительности система считается перегруженной.

После описания взаимодействия дискретных и количественных параметров системы опишем модель ещё одной проблемы, обозначенной ранее в работе, – проблемы каскадных зависимостей микросервисов. Каскадная зависимость между сервисами $s_i$ и $s_j$ характеризуется функцией распространения нагрузки:

$$\lambda_{ij}\big(x_i, x_j\big) = \begin{cases} 1, x_i^{normal} \\ 1 + \alpha_{ij} \cdot \dfrac{\tau_i - \tau_i^{SLA}}{\tau_i^{SLA}}, x_i^{degraded} \end{cases}$$

Где, $\alpha_{ij}$ – коэффициент усиления нагрузки.

Опишем вероятность каскадного отказа, начинающегося с сервиса $s_i$:

$$P(cascade \mid fail_i) \leq 1 - \prod_{path \in Paths(i)} \left(1 - \prod_{(j,k) \in path} p_{jk}^{prop}\right)$$

Где, Paths($i$) – множество путей из $i$, $p_{jk}^{prop}$ – вероятность распространения отказа по ребру ($j$, $k$).

Важно учитывать то, что граф байесовской сети ацикличен. Вероятность каскадного отказа ограничена сверху. Сформулируем задачу оптимизации конфигурации микросервисной системы:

$$\boldsymbol{c}^* = \arg\min_c \mathbb{E}[J(X, c)]$$

Где, функция потерь:

$$J(X, c) = \sum_{i=1}^{n} \left[\omega_i^{cost} \cdot Cost(c_i) + \omega_i^{SLA} \cdot \mathbb{I}\left[\tau_i > \tau_i^{SLA}\right] + \omega_i^{cascade} \cdot P(cascade_i)\right]$$

Где, $\omega$ – коэффициент важности, Cost – функция стоимости.

Докажем, что оптимальная конфигурация $\mathbf{c}^*$ вообще существует. Обозначим условия системы: дискретные множества вариантов параметров конфигурации $D_i$ конечны, что и так выполняется для микросервисной системы; функции стоимости Cost ограничены снизу, что также выполняется для количественных значений ресурсов системы; граф зависимостей ацикличен – это исходит из определения байесовских сетей [8].

Если дискретные множества параметров конфигурации конечны, а ресурсы ограничены, а функция потерь $J$ является непрерывной, поскольку это обеспечивается свойствами непрерывных распределений, на которых она основана, то по теореме Вейерштрасса о функции на компакте [9] минимум достигается. Для вычисления оптимальной конфигурации микросервисной системы был модифицирован байесовский вывод (алгоритм распространения доверия), учитывающий дискретные и количественные параметры. Листинг псевдокода модификации алгоритма приведён ниже.

**Листинг 1. Алгоритм вывода в гибридной (учитывающей дискретные и количественные параметры) байесовской сети**
**Listing 1. Inference algorithm in a hybrid (taking into account discrete and quantitative parameters) Bayesian network**

```
function HybridInference(B, evidence):
// Инициализация сообщений
for (i,j) in E:
m[i→j] = uniform
// Итеративное обновление
repeat until convergence:
for v in V in topological order:
if v in V_D:  // Дискретная вершина
m[v] = DiscreteUpdate(v, incoming_messages)
else:  // Количественная вершина
m[v] = ContinuousUpdate(v, incoming_messages)
// Распространение сообщений
for u in Children(v):
m[v→u] = ComputeMessage(m[v], P(u|v))
// Вычисление маргиналов
for v in V:
P(v) = Normalize(Product(incoming_messages))
return P
```

Алгоритм оптимизации с учётом каскадных зависимостей приведён в листинге 2. Алгоритм начинает свою работу с уже имеющейся оптимальной конфигурации без учёта каскадных эффектов. Функция *EvaluateConfiguration* использует ранее описанную функцию-модификацию байесовского вывода.

**Обсуждение результатов.** Проанализируем временную сложность алгоритма. Обозначим $n$ – количество вершин графа байесовской сети, $d$ – степень вершины, $k$ – количество дискретных значений на вершину. Каждая вершина графа в алгоритме

обрабатывается один раз: для вершин, величина которых принимает дискретные значения, необходимо вычислить $k^d$ элементов таблицы условных вероятностей этих вершин.

**Листинг 2. Алгоритм оптимизации конфигурации микросервисной системы с учётом каскадных зависимостей**

**Listing 2. Algorithm for optimizing the configuration of a microservice system, taking into account cascading dependencies**

```
function CascadeAwareOptimization(M, budget):
// Инициализация: независимая оптимизация
c_best = IndependentOptimization(M, budget)
J_best = EvaluateConfiguration(c_best)
// Итеративное улучшение с учётом каскадов
for iter in 1..max_iterations:
// Выявление критических путей
critical_paths = IdentifyCriticalPaths(c_best)
// Усиление узких мест
for path in critical_paths:
c_candidate = ReinforceBottlenecks(c_best, path)
// Проверка бюджетных ограничений
if TotalCost(c_candidate) <= budget:
J_candidate = EvaluateConfiguration(c_candidate)
// Обновление лучшего решения
if J_candidate < J_best:
c_best = c_candidate
J_best = J_candidate
// Проверка сходимости
if improvement < ε:
break
return c_best
```

Для количественных вершин с нормальным распределением операции выполняются за $O(d^3)$, поскольку такой является временная сложность обращения ковариационной матрицы [10]. При разумных значениях $k$ эта сложность не должна превышать $O(k^d)$. Поэтому можно заключить, что временную сложность алгоритма можно оценить как $O(n \cdot k^d)$. Для тестирования и валидации алгоритма был проведён эксперимент на модельной микросервисной системе онлайн-покупок, состоящей из 15 микросервисов: клиентские микросервисы: мобильное приложение, web-приложение и интерфейс вызова API-запросов напрямую; серверные микросервисы бизнес-логики: авторизация, каталог продуктов, корзина, заказы и платежи; серверные вспомогательные микросервисы: уведомления, аналитика, алгоритм рекомендаций; хранилища данных: базы данных пользователей, продуктов, заказов и кэш. Граф зависимостей микросервисов показан на рис. 1.

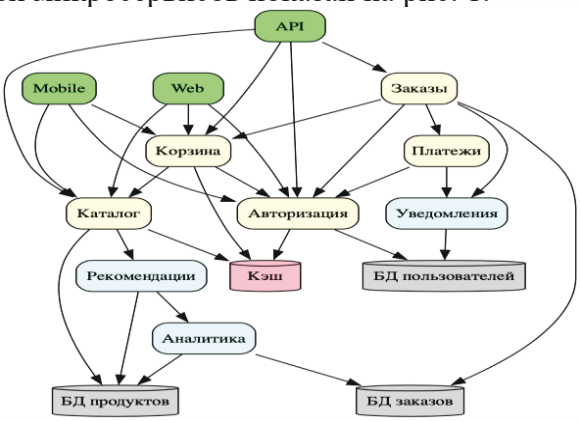


**Рис. 1 – Граф зависимостей микросервисной системы**

**Fig. 1 – Graph of microservice system dependencies**

Для каждого микросервиса были определены следующие параметры: дискретные параметры конфигурации: алгоритм балансировки (round-robin, least-connections, IP-hash), политика обработки ошибок (повторная попытка, логирование, перенаправление);) количественные ресурсы: CPU (1-4 ядра), объём памяти (512 МБ-8ГБ);) SLA-требования:

латентность 50-500 мс в зависимости от микросервиса. Коэффициент усиления нагрузки $\alpha_{ij}$ для каскадных зависимостей выбирался из исторических данных: 1,5 – для синхронных вызовов, 0,8 – для асинхронных, 0,3 – для кэшируемых. Система была размещена в облачном хранилище, после чего подвергалась трём сценариям нагрузки: нормальная нагрузка: 1000 запросов в секунду, равномерное распределение запросов; пиковая нагрузка: 5000 запросов в секунду, всплески с интервалом в 10 минут от нормальной нагрузки; аномальная нагрузка: 10000 запросов в секунду на каталог продуктов. Результаты оценивались по следующим параметрам: процент времени, когда система находилась в состоянии нарушения SLA; средняя латентность системы; вероятность каскадных эффектов выбранной конфигурации.

В эксперименте сравнивались следующие модели: статическая конфигурация – когда параметры были заданы в самом начале и не менялись с течением времени; Kubernetes Horizontal Pod Autoscaling (HPA) – модель реактивного увеличения ресурсов в ответ на повышение нагрузки на систему [11]; предложенная модель оптимизации конфигурации. Результаты сравнения моделей приведены в табл.1.

**Таблица 1. Результаты эксперимента по применению подходов к конфигурации микросервисной системы**
**Table 1. Results of an experiment on the application of approaches to the configuration of a microservice system**

| **Подход Approach** | **Время нарушения SLA Time of violation (%)** | **Средняя латентность (мс) Average latency** | ***P*(cascade)** |
|---|---|---|---|
| Статическая конфигурация Static Configuration | 18,3 | 187 | 0,23 |
| Kubernetes HPA Kubernetes HPA | 8,7 | 142 | 0,15 |
| Предложенный подход Proposed Approach | 5,8 | 124 | 0,08 |

Выбираемая конфигурация действительно является оптимальной: она имеет наименьшую среднюю латентность и наименьший процент времени нахождения в состоянии нарушения SLA. Среди преимуществ предлагаемой модели можно выделить: единая модель для учёта параметров разного рода, что позволяет решать задачу выбора параметров конфигурации с учётом имеющихся ресурсов (например, в работе [12] предлагается только дискретная модель байесовских сетей, как и в большинстве других); явный учёт каскадных зависимостей, повышающий устойчивость системы, влияет на принятие решения о выборе оптимальной конфигурации (например, в работе [13] каскадные эффекты участвуют только в аналитике отказов); интерпретируемость решений: байесовская сеть в качестве модели принятия решений позволяет проследить принятое решение в отличие от моделей, основанных на ИИ, в частности на обучении с подкреплением [14].

Среди ограничений можно привести следующие: байесовская сеть предполагает ацикличность графа зависимостей, однако в реальных системах такие циклы возможны; статичность параметров: модель не учитывает возможное изменение характеристик сервисов во времени.

**Вывод.** Предложенная модель выбора оптимальной конфигурации может быть эффективной. Для закрепления результатов необходимы эксперименты на более масштабных микросервисных системах. Исследования в направлении преодоления ацикличности в графах зависимостей, динамической изменяемости параметров модели и интеграция с существующими системами оркестрации могут помочь улучшить модель и протестировать её в большем количестве существующих реальных микросервисных систем.

**Библиографический список:**

1. Ньюмен С. Создание микросервисов. 2-е изд. – СПб.: Питер, 2023 – 624 с.
2. Кендалл Д. Стохастические процессы, встречающиеся в теории очередей, и их анализ методом вложенных цепей Маркова // Математика. 1959. Т. 3. № 6. С. 97-112.
3. Цыпкин Я.З. Основы теории автоматических систем. М.: Наука, 1977. 560 с.

4. Abraham I., Alvisi L., Halpern J. Distributed computing meets game theory: combining insights from two fields // ACM SIGACT News. 2011. Vol. 42. No. 2. P. 69-76.
5. Mao H., Alizadeh M., Menache I., Kandula S. Resource management with deep reinforcement learning // Proceedings of the 15th ACM Workshop on Hot Topics in Networks. 2016. P. 50-56.
6. Taran V.N. Bayesian networks for modeling complex systems. Proceedings of 2017 IEEE 2nd International Conference on Control in Technical Systems, CTS St. Petersburg, 25–27 October 2017. St. Petersburg, 2017; 240-243. DOI 10.1109/CTSYS.2017.8109535. EDN XXUEQX.
7. Koller D., Friedman N. Probabilistic Graphical Models: Principles and Techniques. MITPress, 2009:1266.
8. Pearl J. Probabilistic reasoning in intelligent systems: networks of plausible inference. CA: Morgan Kaufmann, 1988.
9. Ильин В. А., Позняк Э. Г. Основы математического анализа. Часть I. - М., 1998. - С. 248-251.
10. Амосов А.А., Дубинский Ю.А., Копченова Н.П. Вычислительные методы для инженеров. М.:Мир, 1998.
11. Kubernetes Documentation. Horizontal Pod Autoscaler [Электронный ресурс]. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (дата обращения 26.07.2025).
12. Xu B, Li H, Pang W, et al. Bayesian network approach to fault diagnosis of a hydroelectric generation system. Energy Sci Eng. 2019;7:1669–1677. https ://doi.org/10.1002/ese3.383.
13. Guoqiang L., Zengru D., Ying F. Cascading failures in complex networks with community structure. International Journal of Modern Physics C(IJMPC), World Scientific Publishing Co.Pte. Ltd., 2014;25(5):1-10.
14. https://doi.org/10.1145/3190507. Tao Chen, Rami Bahsoon, and Xin Yao. 2018. A Survey and Taxonomy of Self-Aware and Self-Adaptive Cloud Autoscaling Systems. ACM Comput.Surv.51,3,Article 61.June 2018:40.

**References:**

1. Newman S. Creation of microservices. 2nd ed. – St. Petersburg: Peter, 2023 – 624 p.
2. Kendall D. Stochastic processes occurring in queue theory and their analysis by the nested Markov chain method. *Matematika*. 1959; 3(6):97-112.
3. Tsypkin Ya.Z. Fundamentals of the theory of automatic systems. Moscow: Nauka, 1977. 560 p.
4. Abraham I., Alvisi L., Halpern J. Distributed computing and game Theory: Combining knowledge from two fields. *ACM SIGACT News*. 2011;42(2): 69-76.
5. Mao H., Alizadeh M., Menache I., Kandula S. Resource management through deep reinforcement learning. *Proceedings of the 15th ACM Seminar on current topics in networks*. 2016:50-56.
6. Taran V.N. Bayesian networks for modeling complex systems. Proceedings of 2017 IEEE 2nd International Conference on Control in Technical Systems, CTS St. Petersburg, 25–27 October 2017. St. Petersburg, 2017; 240-243. DOI 10.1109/CTSYS.2017.8109535. EDN XXUEQX.
7. Koller D., Friedman N. Probabilistic graphical models: principles and methods. Publishing House of the Massachusetts Institute of Technology, 2009;1266 p.
8. Pearl J. Probabilistic reasoning in intelligent systems: plausible conclusion networks. California: Morgan Kaufman, 1988.
9. Ilyin V.A., Poznyak E.G. Fundamentals of mathematical analysis. Part I., Moscow, 1998, pp. 248-251.
10. Amosov A.A., Dubinsky Yu.A., Kopchenova N.P. Computational Methods for Engineers, Moscow: Mir, 1998.
11. Kubernetes documentation. Horizontal module of automatic scaling [Electronic resource]. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale / (publication date 07/26/2025).
12. Xu B., Li H., Pang U. et al. Bayesian network approach to fault diagnosis of hydroelectric power plants. Energy Science, 2019;7:1669-1677. https ://doi.org/10.1002/ese3.383.
13. Guoqiang L., Zengru D., Ying F. Cascading failures in complex networks with a community structure. International Journal of Modern Physics C(IJMPC), World Scientific Publishing Co.Pte.Ltd.2014;25(5)1-10.
14. https://doi.org/10.1145/3190507. Tao Chen, Rami Bahsoon, and Xin Yao. 2018. A Survey and Taxonomy of Self-Aware and Self-Adaptive Cloud Autoscaling Systems. ACM Comput.Surv.51,3,Article 61.June 2018:40.

**Сведения об авторе:**

Четвертухин Виктор Романович, аспирант, кафедра программного обеспечения вычислительной техники и автоматизированных систем; victor.chet@mail.ru; ORCID 0009-0003-6110-1227

**Information about author:**

Viktor R. Chetvertukhin, Graduate Student, Department of Computer Engineering and Automated Systems Software; victor.chet@mail.ru; ORCID 0009-0003-6110-1227

**Конфликт интересов/Conflict of interest.**

**Автор заявляет об отсутствии конфликта интересов/The author declare no conflict of interest.**

**Поступила в редакцию/Received** 20.08.2025**.**

**Одобрена после рецензирования/Reviced** 29.10.2025.

**Принята в печать/Accepted for publication** 26.01.2026.