МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

METHODICAL MATERIALS

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика. 2025.
Выпуск 4 (57)
Bulletin of Syktyvkar University.
Series 1: Mathematics. Mechanics. Informatics. 2025; 4 (57)

Научная статья

УДК 519.6, 378
https://doi.org/10.34130/1992-2752_2025_4_73

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ РОЕВОГО ИНТЕЛЛЕКТА: GWO И ChOA

Надежда Николаевна Бабикова,
Надежда Олеговна Котелина
Сыктывкарский государственный университет
имени Питирима Сорокина, valmasha@mail.ru, nad7175@yandex.ru

Аннотация. Метаэвристические алгоритмы роевого интеллекта находят широкое применение в практических задачах: от определения оптимального положения группы захвата полиции до сегментации изображений и определения оптимальных параметров обучения нейросетей. В статье рассматриваются и сравниваются два алгоритма непрерывной оптимизации, моделирующие процесс коллективной охоты животных: алгоритм оптимизации серых волков и алгоритм оптимизации шимпанзе. Также обсуждаются методические вопросы обучения студентов метаэвристическим алгоритмам.

Ключевые слова: роевой интеллект, алгоритм оптимизации серых волков, алгоритм оптимизации шимпанзе, GWO, ChOA, обучение

Для цитирования: Бабикова Н. Н., Котелина Н. О. Сравнительный анализ алгоритмов роевого интеллекта: GWO и ChOA // *Вестник*

Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2025. Вып. 4 (57). С. 73–92. https://doi.org/10.34130/1992-2752_2025_4_73

Article

COMPARATIVE ANALYSIS OF SWARM INTELLIGENCE ALGORITHMS: *GWO* AND *ChOA*

Nadezhda N. Babikova, Nadezhda O. Kotelina
Pitirim Sorokin Syktyvkar State University,
valmasha@mail.ru, nad7175@yandex.ru

Abstract. Metaheuristic swarm intelligence algorithms are widely used in solving problems, ranging from determining the optimal position of a police raiding party to image segmentation and determining optimal training parameters for neural networks. This article examines and compares two continuous optimization algorithms modeling the collective hunting process of animals: the grey wolf optimization algorithm and the chimpanzee optimization algorithm. It also discusses methodological aspects of teaching metaheuristic algorithms to students.

Keywords: swarm intelligence, Grey Wolf Optimizer, Chimp Optimization Algorithm, GWO, ChOA, learning

For citation: Babikova N. N., Kotelina N. O. Comparative analysis of swarm intelligence algorithms: GWO and ChOA. *Vestnik Syktyvkarskogo universiteta. Seriya 1: Matematika. Mekhanika. Informatika* [Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Informatics], 2025, no 4 (57), pp. 73–92. (In Russ.) https://doi.org/10.34130/1992-2752_2025_4_73

1. Введение

Спектр метаэвристических алгоритмов оптимизации разнообразен и активно расширяется: за последние три десятилетия было предложено более 500 новых алгоритмов [1]. Метаэвристики можно разделить на три основных класса: эволюционные алгоритмы; алгоритмы, основанные на физических законах; алгоритмы роевого интеллекта [2]. Среди алгоритмов, вдохновленных концепциями эволюции в природе, самым популярным является генетический алгоритм (Genetic Algorithm, GA), который

имитирует дарвиновские концепции эволюции [3]. Среди мета-эвристик, реализующих поиск оптимального решения в соответствии с принципами природных физических процессов, наиболее известным является алгоритм имитации отжига (Simulated Annealing, SA) [4]. К наиболее распространенным алгоритмам роевого интеллекта, имитирующим групповое поведение живых организмов, относятся алгоритм муравьиной колонии (Ant Colony Optimization, ACO) [5] и алгоритм роя частиц (Particle Swarm Optimization, PSO) [6].

По типу имитируемых поведенческих стратегий среди алгоритмов роевого интеллекта можно выделить шесть групп, в которых моделируются [7]:

1) поведение особей в процессе поиска пищи;

2) процессы размножения или спаривания;

3) социальное взаимодействие в рамках определенной цели;

4) навигация роя, группы, стаи, стада;

5) поведение в процессе охоты;

6) процессы выживания.

Метаэвристические алгоритмы оптимизации применяются в различных практических задачах: сегментации и кластеризации изображений [8], оптимизации инвестиционного портфеля [9], выбора рабочих частот радиотехнических средств [10], определения оптимального размещения группы задержания полиции [11], структурно-параметрической оптимизации в строительном проектировании [12], определения оптимальной начальной скорости обучения нейросети [13] и многих других.

В статье рассматриваются два алгоритма роевого интеллекта, основанные на имитации поведения особей в процессе охоты, — алгоритм оптимизации серых волков и алгоритм оптимизации шимпанзе, а также обсуждаются методические вопросы, связанные с обучением студентов метаэвристическим алгоритмам роевого интеллекта.

2. Материалы и методы

Метаэвристические алгоритмы могут изучаться в вузах как в рамках самостоятельных дисциплин, так и в качестве отдельных тем в составе комплексных дисциплин. В Сыктывкарском университете мета-

эвристические алгоритмы входят в программу дисциплины «Математические методы в экономике» (направления «Прикладная информатика», бакалавриат; «Математика и компьютерные науки», магистратура). Также метаэвристические алгоритмы используются студентами в самостоятельной научно-исследовательской работе, при выполнении курсовых и дипломных проектов.

Для рассмотрения алгоритмов непрерывной оптимизации были выбраны два алгоритма: алгоритм оптимизации серых волков и алгоритм оптимизации шимпанзе. Алгоритм оптимизации серых волков сравнительно прост в реализации, иерархическая система организации стаи интуитивно понятна. Это позволяет использовать алгоритм как базовый для обучения студентов. Алгоритм оптимизации шимпанзе гораздо сложнее для понимания и реализации. Этот алгоритм был предложен в 2020 году, и по нему нет научных статей на русском языке, тем интереснее авторам было его изучать. Также на выбор алгоритма оптимизации шимпанзе повлияло то, что для большого набора тестовых целевых функций алгоритм шимпанзе продемонстрировал один из лучших результатов по данным сравнительного анализа эффективности 21 алгоритма роевого интеллекта, представленного K. Warnakulasooriya, A. Segev в 2024 году [7].

3. Результаты

3.1. Метаэвристический алгоритм оптимизации серых волков (Grey Wolf Optimizer, GWO) — алгоритм глобальной непрерывной оптимизации — был предложен S. Mirjalili, S. M. Mirjalili, A. Lewis в 2014 году [2] и был одним из первых алгоритмов, моделирующих поведение особей в процессе охоты. Алгоритм основан на имитации социальной иерархии волчьей стаи и процесса групповой охоты — поиска, окружения и нападения на добычу. В стае волков выделяются три лидера — альфа, бета, гамма, все остальные волки (омеги) в процессе охоты следуют за лидерами.

Пусть задана целевая функция $f(\overrightarrow{X}) = f(x_1, x_2, \ldots, x_n)$ на множестве $D \subset \mathbb{R}^n$. Для математического моделирования окружения волками добычи (оптимального решения) предложены следующие уравнения:

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t)|, \tag{1.1}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A} \cdot \overrightarrow{D}, \tag{1.2}$$

$$\overrightarrow{A} = 2\overrightarrow{f} \cdot \overrightarrow{r_1} - \overrightarrow{f}, \tag{1.3}$$

$$\overrightarrow{C} = 2\overrightarrow{r_2}, \tag{1.4}$$

где $\overrightarrow{A}$, $\overrightarrow{C}$ — векторы коэффициентов, $\overrightarrow{X_p}$ — вектор, определяющий позицию добычи; $\overrightarrow{X}$ — вектор, определяющий позицию волка на текущей итерации t; $\overrightarrow{r_1}$, $\overrightarrow{r_2}$ — случайные векторы с координатами на интервале $[0, 1]$; координаты вектора $\overrightarrow{f}$ линейно убывают от 2 до 0.

В формулах (1.1)–(1.4) и далее указан знак произведения между векторами так, как в оригинальной статье, но по контексту понятно, что имеется в виду поэлементное произведение векторов по Адамару ($\otimes$).

Интерпретация параметров итерационного процесса охоты для случая двух переменных представлена на рис. 1. Компоненты вектора $\overrightarrow{f}$ на каждой итерации определяются по формуле $2 - 2\,t/T$, где t — номер итерации, T — максимальное число итераций. То есть компоненты убывают линейно от 2 до 0. Соответственно координаты вектора $\overrightarrow{A}$ изменяются случайным образом на каждой итерации от $-f_j$ до f_j ($j = \overline{1, n}$). Вектор $\overrightarrow{A}$ влияет на величину смещения волка из текущей позиции. Если $|\overrightarrow{A}| < 1$, новая позиция волка будет расположена между ним и добычей, если $|\overrightarrow{A}| > 1$ — дальше от добычи (в случае нулевого вектора $\overrightarrow{A}$ волк переместился бы непосредственно на позицию добычи).

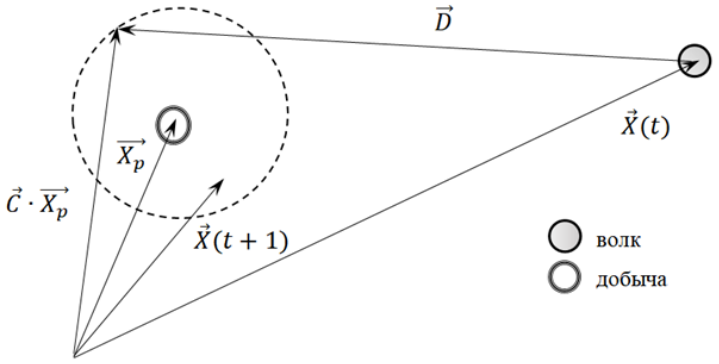

Рис. 1. Математическая модель процесса охоты

Координаты случайного вектора $\overrightarrow{C}$ также вычисляются на каждой итерации и изменяются от 0 до 2. Вектор $\overrightarrow{C}$ задает радиус влияния добычи на определение смещения $\overrightarrow{D}$ волка из текущей позиции: чем больше по абсолютной величине компоненты вектора, тем больше влияет положение добычи на новое положение волка.

Начальная популяция волков $\{\overrightarrow{X_1},\ \overrightarrow{X_2},\ \ldots,\ \overrightarrow{X_N}\}$ генерируется случайным образом. Для каждого волка вычисляется значение целевой функции, определяются альфа, бета, гамма, соответствующие трем лучшим значениям целевой функции.

В абстрактном пространстве поиска местоположение добычи неизвестно, поэтому на каждом шаге итеративного процесса охоты позиции всех волков (включая лидеров) обновляются согласно математической модели (1.1–1.4) в соответствии с позициями альфы, беты, гаммы по формулам:

$$\overrightarrow{D_\alpha} = |\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \overrightarrow{X}|, \quad \overrightarrow{D_\beta} = |\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \overrightarrow{X}|, \quad \overrightarrow{D_\gamma} = |\overrightarrow{C_3} \cdot \overrightarrow{X_\gamma} - \overrightarrow{X}|, \quad (1.5)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot \overrightarrow{D_\alpha}, \quad \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \overrightarrow{D_\beta}, \quad \overrightarrow{X_3} = \overrightarrow{X_\gamma} - \overrightarrow{A_3} \cdot \overrightarrow{D_\gamma}, \quad (1.6)$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3}, \quad (1.7)$$

где параметры $\overrightarrow{A_i},\ \overrightarrow{C_i}$ вычисляются в соответствии с формулами (1.3–1.4) для $i = \overline{1,3}$; $\overrightarrow{X_\alpha},\ \overrightarrow{X_\beta},\ \overrightarrow{X_\gamma}$ — векторы, определяющие позиции лидеров; $\overrightarrow{X}$ — вектор, определяющий текущую позицию волка (итерация t), t — номер итерации. Заметим, что случайные векторы $\overrightarrow{r_1},\ \overrightarrow{r_2}$ для вычисления $\overrightarrow{A_i},\ \overrightarrow{C_i}$ генерируются для каждого значения i.

Ниже приведена программа реализации алгоритма для функции Растригина двух переменных $f(x_1, x_2) = 20 + x_1^2 - 10\cos(2\pi x_1) + x_2^2 - -10\cos(2\pi x_2)$, где $x_i \in [-5.12, 5.12]$, на языке Python (листинг). Функция Растригина — известная невыпуклая функция, которая используется для тестирования алгоритмов оптимизации. Функция имеет глобальный минимум, равный 0, в точке $(0, 0)$, а также 120 локальных минимумов в точках с целочисленными координатами.

Для наглядного представления работы алгоритма в двумерном случае можно использовать визуализацию перемещения «стаи точек» с ростом числа итераций (рис. 2).

Листинг

Алгоритм оптимизации волков

```
import numpy as np
def rastrigin_2d(x):
    A = 10
    return (A * 2 + (x[0]**2 - A * np.cos(2 * np.pi * x[0]))
            + (x[1]**2 - A * np.cos(2 * np.pi * x[1])))
```

Окончание листинга

```
dim = 2
pop_size = 30
max_iter = 100
lb, ub = -5.12 * np.ones(dim), 5.12 * np.ones(dim)
# счетчики фаз нападений и исследований
attack = 0
exploration = 0
L = 3                                          # количество лидеров
# Инициализация популяции
population = np.random.uniform(lb, ub, (pop_size, dim))

for iteration in range(max_iter):
    f = 2.0 - 2.0 * (iteration / max_iter)
    fitness = np.array([rastrigin_2d(ind) for ind in population])
    # Определяем лидеров
    sorted_indices = np.argsort(fitness)
    leaders_indices = sorted_indices[:L]        # Индексы лидеров
    leaders = population[leaders_indices].copy()  # Копии лидеров

    for i in range(pop_size):
        r1, r2 = np.random.rand(L, dim), np.random.rand(L, dim)
        A = 2 * f * r1 - f
        C = 2 * r2

        # Подсчет фаз атак и исследований
        if np.any(np.abs(A) > 1):               # исследование
            exploration += 1
        else:                                   # атака
            attack += 1

        D = np.abs(C * leaders - population[i])
        new_pos = np.sum(leaders - A * D, axis=0) / L
        population[i] = np.clip(new_pos, lb, ub)

    if (iteration + 1) % 10 == 0:
        best_fitness = np.min(fitness)
        print(f"Iteration {iteration + 1}/{max_iter}, "
              f"Best Fitness: {best_fitness:.9f}")
print('количество атак = ', attack,
      'количество исследований = ', exploration)
best_idx = np.argmin(fitness)
best_solution, best_value = population[best_idx], fitness[best_idx]

print("Алгоритм стаи серых волков")
print("=" * 70)
print("\n" + "=" * 70)
print("РЕЗУЛЬТАТЫ ОПТИМИЗАЦИИ:")
print(f"Найденное лучшее решение: ({best_solution[0]:.9f}, "
      f"{best_solution[1]:.9f})")
print(f"Значение функции: {best_value:.9f}")
print(f"Истинный минимум: (0.000000, 0.000000), "
      f"значение: {rastrigin_2d([0, 0]):.9f}")
print(f"Ошибка: {abs(best_value - 0):.9f}")
```

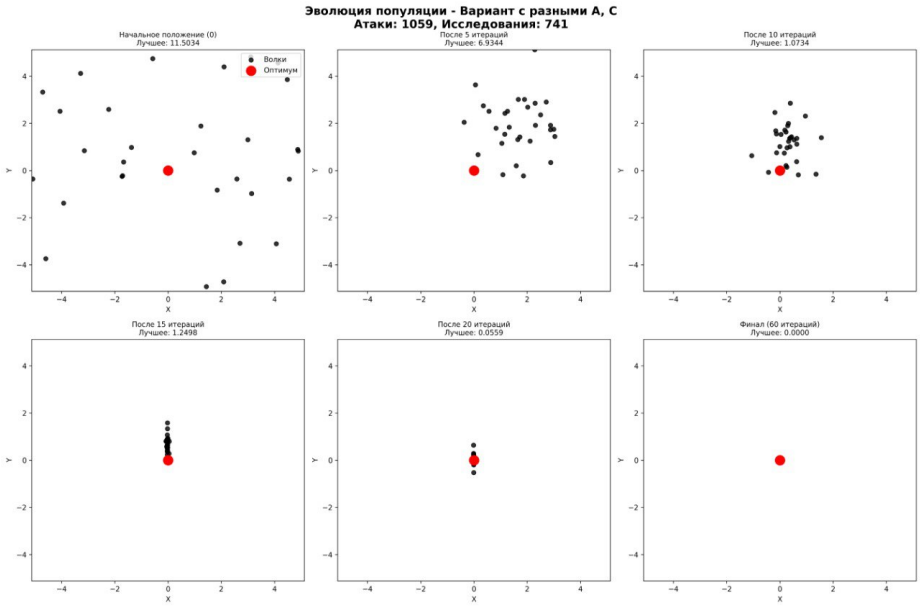

Рис. 2. Визуализация работы GWO для функции Растригина (листинг)

Алгоритм для функции Растригина находит решение довольно быстро, обычно уже к 20-й итерации (при размере популяции 30, максимальном количестве итераций 100), очень редко попадая в один из локальных оптимумов. Для более сложной тестовой функции Швефеля $f(x_1, x_2) = 837.9658 - x_1 \sin(\sqrt{|x_1|}) - x_2 \sin(\sqrt{|x_2|})$, где $x_i \in [-500, 500]$, лучшее решение было получено при размере популяции 1000, максимальном количестве итераций 300 — $(420.705926187, 421.098038253)$, значение функции: 0.000089958. Глобальный минимум функции Швефеля равен 0 и находится в углу пространства поиска, в точке с координатами $(420.96875, 420.96875)$, в то время как второй по глубине локальный минимум расположен вблизи противоположного угла (рис. 3). Полученные при тестировании алгоритма на функции Швефеля результаты аналогичны результатам, приведенным в статье А. В. Пантелеева, И. А. Белякова [14].

3.2. Алгоритм оптимизации шимпанзе (Chimp Optimization Algorithm, ChOA) был разработан M. Khishe, M. R. Mosavi в 2020 году [15], является алгоритмом непрерывной оптимизации.

Алгоритм шимпанзе моделирует коллективную охоту стаи шимпанзе. В охоте принимают участие четыре группы особей с разными ролями: Attackers — атакующие, Barriers — блокирующие, Chasers — преследователи, Drivers — загонщики.

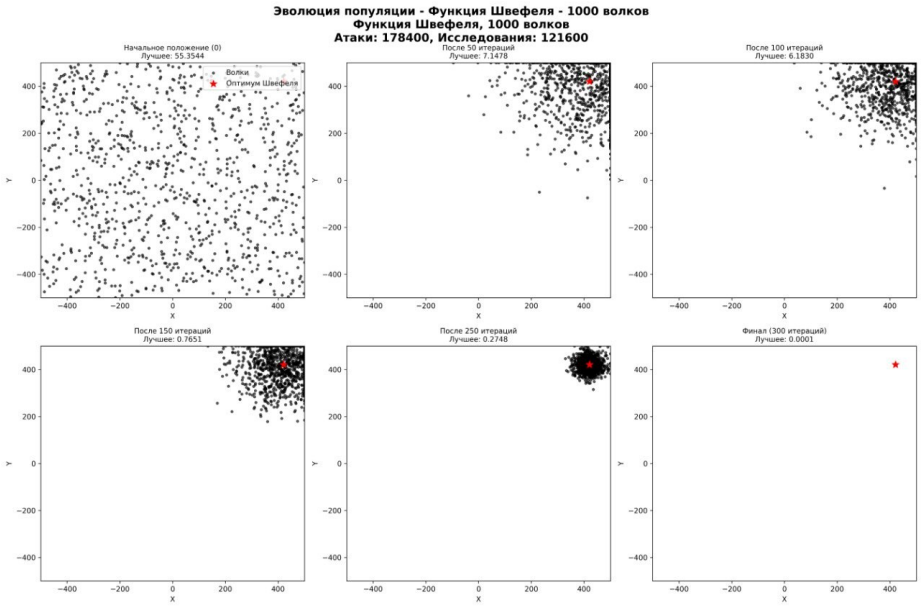

Рис. 3. Визуализация работы GWO для функции Швефеля

Рассмотрим математическую модель охоты шимпанзе в сравнении с охотой серых волков (табл.). Отличия два: во-первых, при определении смещения шимпанзе к новой позиции вводится дополнительный элемент случайности — хаотический вектор $\overrightarrow{m}$; во-вторых, компоненты вектора $\overrightarrow{f}$ убывают нелинейно (у волков — линейно) от 2.5 до 0 (у волков — от 2), причем этот параметр авторы предлагают изменять по разным правилам для каждой из четырех групп шимпанзе.

Таблица

Сравнение моделей GWO и ChOA

Серые волки	**Шимпанзе**	
$\overrightarrow{D} = \left\lvert \overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t) \right\rvert$	$\overrightarrow{D} = \left\lvert \overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{\boldsymbol{m}} \cdot \overrightarrow{X}(t) \right\rvert$	(2.1)
$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A} \cdot \overrightarrow{D}$	$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A} \cdot \overrightarrow{D}$	(2.2)
$\overrightarrow{A} = 2\overrightarrow{f} \cdot \overrightarrow{r}_1 - \overrightarrow{f}$	$\overrightarrow{A} = 2\overrightarrow{f} \cdot \overrightarrow{r}_1 - \overrightarrow{f}$	(2.3)
$\overrightarrow{C} = 2\overrightarrow{r}_2$	$\overrightarrow{C} = 2\overrightarrow{r}_2$	(2.4)
$\overrightarrow{f}$ убывает линейно от 2 до 0	$\overrightarrow{f}$ убывает **не**линейно от **2.5** до 0	

Начальная популяция шимпанзе $\{\overrightarrow{X_1},\ \overrightarrow{X_2},\ \ldots,\ \overrightarrow{X_N}\}$ генерируется случайным образом. Популяция случайным образом делится на четыре группы приблизительно равного размера. Для каждого шимпанзе вычисляется значение целевой функции, определяются лидеры: A (Attacker), B (Barrier), C (Chaser), D (Driver), соответствующие четырем лучшим значениям целевой функции.

На каждом шаге итеративного процесса охоты позиции всех шимпанзе обновляются согласно математической модели (2.1–2.4) в соответствии с позициями A, B, C, D по формулам:

$$\begin{aligned}
\overrightarrow{D_A} &= |\overrightarrow{C_1} \cdot \overrightarrow{X_A} - \overrightarrow{m_1} \cdot \overrightarrow{X}|, \\
\overrightarrow{D_B} &= |\overrightarrow{C_2} \cdot \overrightarrow{X_B} - \overrightarrow{m_2} \cdot \overrightarrow{X}|, \\
\overrightarrow{D_C} &= |\overrightarrow{C_3} \cdot \overrightarrow{X_C} - \overrightarrow{m_3} \cdot \overrightarrow{X}|, \\
\overrightarrow{D_D} &= |\overrightarrow{C_4} \cdot \overrightarrow{X_D} - \overrightarrow{m_4} \cdot \overrightarrow{X}|,
\end{aligned} \tag{2.5}$$

$$\begin{aligned}
\overrightarrow{X_1} &= \overrightarrow{X_A} - \overrightarrow{A_1} \cdot \overrightarrow{D_A}, \\
\overrightarrow{X_2} &= \overrightarrow{X_B} - \overrightarrow{A_2} \cdot \overrightarrow{D_B}, \\
\overrightarrow{X_3} &= \overrightarrow{X_C} - \overrightarrow{A_3} \cdot \overrightarrow{D_C}, \\
\overrightarrow{X_4} &= \overrightarrow{X_D} - \overrightarrow{A_4} \cdot \overrightarrow{D_D},
\end{aligned} \tag{2.6}$$

$$\overrightarrow{X}(t+1) = \begin{cases}
\dfrac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3} + \overrightarrow{X_4}}{4}, & \text{если } \mu < 0.5,\ |A| < 1 \\
\overrightarrow{X_{rand}} - \overrightarrow{A} \cdot \overrightarrow{D}, & \\
\quad \text{где } \overrightarrow{D} = \left|\overrightarrow{C} \cdot \overrightarrow{X_{rand}} - \overrightarrow{m} \cdot \overrightarrow{X}\right|, & \text{если } \mu < 0.5,\ |A| > 1 \\
\text{хаотический вектор,} & \text{если } \mu \geq 0.5
\end{cases}, \tag{2.7}$$

где параметры $\overrightarrow{A},\ \overrightarrow{C},\ \overrightarrow{A_i},\ \overrightarrow{C_i},\ i = \overline{1,4}$ вычисляются в соответствии с формулами (2.3), (2.4); $\overrightarrow{m},\ \overrightarrow{m_i},\ i = \overline{1,4}$ — хаотические векторы; $\overrightarrow{X_A},\ \overrightarrow{X_B},\ \overrightarrow{X_C},\ \overrightarrow{X_D}$ — векторы, определяющие позиции лидеров; $\overrightarrow{X}$ — вектор, определяющий текущую позицию шимпанзе (итерация t); $\overrightarrow{X_{rand}}$ — вектор, определяющий позицию случайно выбранного шимпанзе из популяции; t — номер итерации; μ — случайное значение от 0 до 1, которое генерируется на каждой итерации для каждого шимпанзе.

Аналогично действиям стаи волков, поведение шимпанзе направлено как на исследование пространства решений при $|A| > 1$ (exploration, фаза исследования, глобальный поиск), так и на использование уже найденных хороших решений при $|A| < 1$ (exploitation, фаза нападения на добычу, локальный поиск).

Формула (2.7) составлена на основе псевдокода, представленного в оригинальной статье [15]. Параметр μ авторы алгоритма трактуют как влияние случайностей во время охоты, т. е. в половине случаев шимпанзе могут отвлечься от процесса охоты (побежать за самкой, подраться и т. п.), в этом случае позиция шимпанзе обновляется хаотически ($\mu \geq 0.5$). Когда $\mu < 0.5$ & $|A| > 1$, текущий шимпанзе обновляет свою позицию, ориентируясь на случайно выбранного шимпанзе.

4. Обсуждение

Сравнительный анализ алгоритмов GWO и ChOA показывает, что они базируются на сходной абстрактной модели процесса охоты, но алгоритм ChOA характеризуется существенно более высокой структурной сложностью, что делает его достаточно трудным для понимания и реализации. Один из возможных методов изучения алгоритма оптимизации шимпанзе заключается в постепенном переходе к нему от более простого алгоритма оптимизации серых волков. Такой подход позволяет наглядно продемонстрировать развитие метаэвристических концепций.

Метаэвристических алгоритмов оптимизации более 500, но среди них не только оригинальные алгоритмы, но и модифицированные, а также гибридные (которые объединяют идеи нескольких алгоритмов).

Существует много модификаций GWO, например, в 2016 году N. Mittal, U. Singh, B. S. Sohi предложили свою версию алгоритма и показали его эффективность для инженерных приложений [16]. Если в оригинальном алгоритме параметр $\overrightarrow{f}$ убывает линейно по правилу $2 - 2t/T$ (t, T — текущий и максимальный номер итерации), то в модифицированном этот параметр убывает нелинейно по правилу $2 - 2t^2/T^2$.

Координаты вектора $\overrightarrow{A} = 2\overrightarrow{f} \cdot \overrightarrow{r_1} - \overrightarrow{f}$ изменяются случайным образом на каждой итерации от $-f_j$ до f_j ($j = \overline{1, n}$). То есть диапазон изменения координат вектора $\overrightarrow{A}$ с ростом номера итерации уменьшается. Поскольку $|\overrightarrow{A}| < 1$ соответствует фазе локального поиска (нападения на добычу), $|\overrightarrow{A}| > 1$ — фазе глобального поиска (исследования), то, меняя правило убывания параметра $\overrightarrow{f}$, мы меняем соотношение между количеством итераций нападения и итераций исследования. Напри-

мер, отношение среднего числа нападений при многократном запуске программы для функции Растригина (листинг) к среднему числу исследований — 3:2. А если использовать для $\overrightarrow{f}$ нелинейное убывание $2 - 2t^2/T^2$, то отношение среднего числа нападений к среднему числу исследований — 2:3. То есть количество фаз исследования возрастает, что снижает вероятность стагнации в локальных оптимумах.

Авторы ChOA предлагают следующие формулы изменения параметра $\overrightarrow{f}$ для каждой из четырех групп шимпанзе: $2.5 - \dfrac{2\log(t)}{\log(T)}$, $(-2t^3/T^3) + 2.5$, $0.5 + 2\exp(-(4t/T)^2)$, $2.5 + 2t^2/T^2 - 2 \cdot 2t/T$.

Количество лидеров, определяющих позицию предполагаемой добычи в метаэвристических алгоритмах, может быть различным и даже меняться динамически в ходе работы алгоритма. Всего один лидер используется, например, в алгоритме летучих мышей (Bat Algorithm, BA), разработанном X.-S. Yang в 2010 году [17], в алгоритме оптимизации китов (Whale Optimization Algorithm, WOA), представленном S. Mirjalili, A. Lewis в 2016 году [18]. В приведенной выше программе (листинг) можно изменить количество лидеров (переменная L) и проследить, как меняется при этом работа алгоритма.

Добавление хаотического фактора является популярным приемом модификации метаэвристических алгоритмов. Хаотические карты (отображения) генерируют детерминированные, но трудно предсказуемые последовательности. Хаос может быть интегрирован в алгоритм разными способами. Например, в 2023 году коллектив авторов предложил модифицированную версию алгоритма оптимизации синус-косинуса (Sine Cosine Algorithm, SCA) [19], в которой хаотические карты используются для инициализации начальной популяции [20]. Инициализация с помощью хаотических отображений обеспечивает более равномерное и разнообразное начальное распределение особей в пространстве поиска по сравнению со случайной инициализацией, что увеличивает шансы найти глобальный оптимум.

В ChOA хаотический вектор $\overrightarrow{m}$ вносит хаос в траекторию перемещения шимпанзе к оптимуму (8), «выталкивая» их из локальных оптимумов. При этом разные карты по-разному меняют паттерны движения шимпанзе, например, большая амплитуда колебаний может приводить к резким скачкам позиций шимпанзе и, как следствие, к пропуску оптимума, но, с другой стороны, позволяет добиться большей эффективности на ранних этапах перемещения.

Авторы ChOA рекомендуют использовать одну из шести хаотических карт с начальным значением $x_0 = 0.7$ (для тент-карты $x_0 = 0.6$), для приведения к диапазону $[0, 1)$ используется взятие дробной части (рис. 4):

- Квадратичная карта: $x_{i+1} = \{x_i^2 - c\},\ c = 2.$
- Гауссова карта: $x_{i+1} = \begin{cases} 1, & x_i = 0, \\ 1/\{x_i\}, & x_i \neq 0. \end{cases}$
- Логистическая карта: $x_{i+1} = \alpha x_i(1 - x_i),\ \alpha = 4.$
- Карта Зингера:

$$x_{i+1} = \mu(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4),\ \mu = 1.07.$$

- Карта Бернулли: $x_{i+1} = \{2x_i\}.$
- Тент-карта: $x_{i+1} = \begin{cases} x_i/0.7, & x_i < 0.7, \\ \frac{10}{3}(1 - x_i), & 0.7 \leq x_i. \end{cases}$

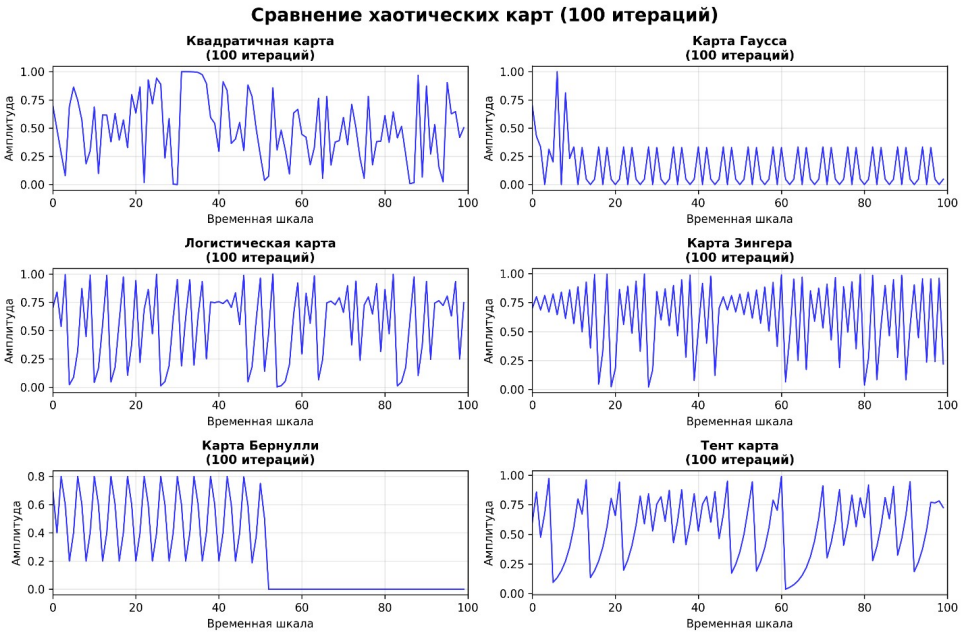

Рис. 4. Хаотические карты

Разделение популяции на группы встречается во многих метаэвристических алгоритмах. Например, в алгоритме оптимизации слонов (Elephant Herding Optimization, EHA) вся популяция разделена на кланы, каждый из которых ищет решение независимо от других кланов [21]. В ChOA независимые группы используют различные нелинейные стратегии обновления $\overrightarrow{f}$, поэтому шимпанзе могут исследовать пространство поиска с разными возможностями, обеспечивая балансировку между глобальным и локальным поиском.

Конечно, алгоритм оптимизации шимпанзе весьма сложный, не всегда целесообразно включать такие алгоритмы в программу обучения. В этом случае можно сравнивать некоторый базовый алгоритм с его модификациями. Если у студентов складывается понимание метаэвристического метода решения задач, то они могут (и возможно, захотят) изучать более сложные и интересные алгоритмы в рамках научно-исследовательской работы, курсовых и дипломных проектов.

Визуальная и поведенческая наглядность алгоритмов, моделирующих процесс охоты, делает их удобными для реализации в игровых и симуляционных проектах, где важна демонстрация принципов коллективного поведения агентов. Примером нестандартного применения алгоритма оптимизации серых волков может служить курсовая работа студента группы 131б-ПИо Д. И. Марченко (рис. 5), он разработал компьютерную игру «Поймай меня», в которой игрок убегал от стаи волков, преследующей его по правилам алгоритма оптимизации серых волков.

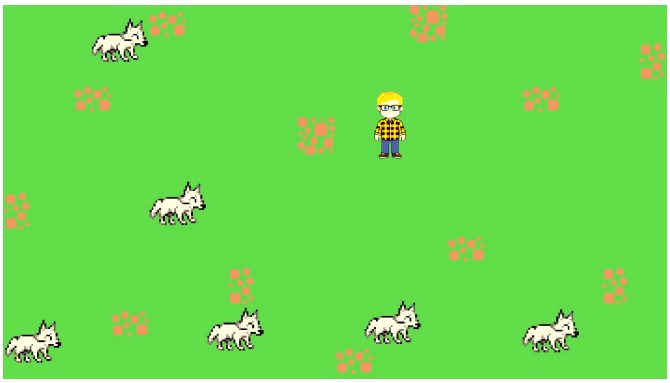

Рис. 5. Скриншот игры «Поймай меня»

5. Заключение

Применение сравнительного анализа метаэвристических алгоритмов, построенного по принципу перехода от простых моделей к более сложным, не только обеспечивает доступность материала, но и способствует формированию представления о способах совершенствования методов оптимизации. Обучение начинается с освоения базовых концепций на примере алгоритмов, отличающихся простой структурой и небольшим числом параметров. При переходе к более сложным алгоритмам становится возможным показать, каким образом решаются присущие первоначальной модели проблемы.

Список источников

1. **Tansel D., Canturk D., Kucukyilmaz T.** A survey on pioneering metaheuristic algorithms between 2019 and 2024 [Электронный ресурс]. URL: https://www.researchgate.net/publication/388421856_A_survey_on_ pioneering_metaheuristic_algorithms_between_2019_and_2024 (дата обращения: 01.10.2025).

2. **Mirjalili S., Mirjalili S. M., Lewis A.** Grey Wolf Optimizer // *Advances in Engineering Software*. 2014. Vol. 69. Pp. 46–61. DOI: 10.1016/j.advengsoft.2013.12.007.

3. **Katoch S., Chauhan S.S., Kumar V.** A review on genetic algorithm: past, present, and future // *Multimed Tools and Applications*. 2021. Vol. 80. Pp. 8091–8126. DOI: 10.1007/s11042-020-10139-6.

4. **Kirkpatrick S., Gelatt C. D., Vecchi M. P.** Optimization by Simulated Annealing // *Science*. 1983. Vol. 220. No 4598. Pp. 671–680. DOI: 10.1126/science.220.4598.671.

5. **Dorigo M., Birattari M., Stutzle T.** Ant colony optimization // *Computational Intelligence Magazine, IEEE*. 2006. Vol. 1. Pp. 28–39. DOI: 10.1109/MCI.2006.329691.

6. **Poli R., Kennedy J., Blackwell T.** Particle swarm optimization // *Swarm Intell*. 2007. Vol. 1. Pp. 33–57. DOI: 10.1007/s11721-007-0002-0.

7. **Warnakulasooriya K., Segev A.** Comparative analysis of accuracy and computational complexity across 21 swarm intelligence

algorithms // *Evolutionary Intelligence*. 2024. Vol. 18. Issue 18. DOI: 10.1007/s12065-024-00997-6.

8. **Родзин С. И., Эль-Хатиб С. А.** Совершенствование алгоритмов сегментации магнитно-резонансных изображений на основе роевого интеллекта // *Вестник Чувашского университета*. 2016. № 3. С. 217–226.

9. **Акиншин О. Н., Есиков Д. О., Акиншина Н. Ю.** Особенности решения задачи оптимизации инвестиционного портфеля предприятия методом роя частиц // *Известия Тульского государственного университета. Технические науки*. 2016. № 5. С. 109–116.

10. **Есиков О. В., Румянцев В. Л., Старожук Е. А.** Применение роевых алгоритмов для решения задачи выбора рабочих частот радиотехнических средств системы управления воздушным движением // *Известия Тульского государственного университета. Технические науки*. 2016. № 2. С. 85–92.

11. **Пьянков О. В., Попов А. В.** Модель принятия решения по повышению оперативности реагирования групп задержания с применением роевых алгоритмов // *Вестник Воронежского института МВД России*. 2020. № 4. С. 73–83.

12. **Ахмадиев Ф. Г., Маланичев И. В.** Популяционные алгоритмы структурно-параметрической оптимизации в строительном проектировании // *Известия Казанского государственного архитектурно-строительного университета*. 2018. № 2 (44). С. 215–223.

13. **Rahmatulloh A., Nugraha G. F., Darmawan I.** Hybrid PSO-Adam Optimizer Approach for Optimizing Loss Function Reduction in the Dist-YOLOv3 Algorithm // *International Journal of Intelligent Engineering and Systems*. 2024. Vol. 17. No 5. Pp. 199–209. DOI: 10.22266/ijies2024.1031.16.

14. **Пантелеев А. В., Беляков И. А.** Разработка программного обеспечения метода глобальной оптимизации, имитирующего поведение стаи серых волков // *Моделирование и анализ данных*. 2021. № 2. С. 59–73. DOI: 10.17759/mda.2021110204.

15. **Khishe M., Mosavi M. R.** Chimp optimization algorithm // *Expert Systems with Applications*. 2020. Vol. 149. P. 113338. DOI: 10.1016/j.eswa.2020.113338.

16. **Mittal N., Singh U., Sohi B. S.** Modified grey wolf optimizer for global engineering optimization // *Applied Computational Intelligence and Soft Computing*. 2016. Article ID 7950348. (4598). Pp. 1–16. DOI: 10.1155/2016/7950348.

17. **Yang X.-S.** A New Metaheuristic Bat-Inspired Algorithm // *Nature Inspired Cooperative Strategies for Optimization (eds. J. R. Gonzalez et al.)*, Studies in Computational Intelligence, Springer Berlin, 284, Springer. 2010. Pp. 65–74. DOI: 10.1007/978-3-642-12538-6_6.

18. **Mirjalili S., Lewis A.** The Whale Optimization Algorithm // *Advances in Engineering Software*. 2016. No 95. Pp. 51–67. DOI: 10.1016/j.advengsoft.2016.01.008.

19. **Mirjalili S.** SCA: a sine cosine algorithm for solving optimization problems // *Knowl. Based Syst.* 2016. Vol. 96. Pp. 120–133. DOI: 10.1016/j.knosys.2015.12.022.

20. **Pilcevic D., Djuric Jovicic M., Antonijevic M. et al.** Performance evaluation of metaheuristics-tuned recurrent neural networks for electroencephalography anomaly detection // *Front Physiol*. 2023. Nov 14; 14: 1267011. DOI: 10.3389/fphys.2023.1267011.

21. **Almufti S. M., Asaad R. R., Salim B. W.** Review on Elephant Herding Optimization Algorithm Performance in Solving Optimization Problems // *International Journal of Engineering & Technology*. 2018. 7 (4). Pp. 6109–6114. DOI: 10.14419/ijet.v7i4.23127.

References

1. **Tansel D., Canturk D., Kucukyilmaz T.** *A survey on pioneering metaheuristic algorithms between 2019 and 2024.* 2024. Available at: https://www.researchgate.net/publication/388421856_A_survey_on_pioneering_metaheuristic_algorithms_between_2019_and_2024 (accessed: 01.10.2025).

2. **Mirjalili S., Mirjalili S. M., Lewis A.** Grey Wolf Optimizer. *Advances in Engineering Software*. 2014. Vol. 69. Pp. 46–61. DOI: 10.1016/j.advengsoft.2013.12.007.

3. **Katoch S., Chauhan S.S., Kumar V.** A review on genetic algorithm: past, present, and future. *Multimed Tools and Applications*. 2021. Vol. 80. Pp. 8091–8126. DOI: 10.1007/s11042-020-10139-6.

4. **Kirkpatrick S., Gelatt C. D., Vecchi M. P.** Optimization by Simulated Annealing. *Science*. 1983. Vol. 220. No 4598. Pp. 671–680. DOI: 10.1126/science.220.4598.671.

5. **Dorigo M., Birattari M., Stutzle T.** Ant colony optimization. *Computational Intelligence Magazine. IEEE*. 2006. Vol. 1. Pp. 28–39. DOI: 10.1109/MCI.2006.329691.

6. **Poli R., Kennedy J., Blackwell T.** Particle swarm optimization. *Swarm Intell.* 2007. Vol. 1. Pp. 33–57. DOI: 10.1007/s11721-007-0002-0.

7. **Warnakulasooriya K., Segev A.** Comparative analysis of accuracy and computational complexity across 21 swarm intelligence algorithms. *Evolutionary Intelligence*. 2024. Vol. 18. Issue 18. DOI: 10.1007/s12065-024-00997-6.

8. **Rodzin S. I., El'-Khatib S. A.** Improvement of Magnetic Resonance Image Segmentation Algorithms Based on Swarm Intelligence. *Vestnik Chuvashskogo universiteta* [Bulletin of Chuvash University]. 2016. No 3. Pp. 217–226. (In Russ.)

9. **Akinshin O. N., Esikov D. O., Akinshina N. Yu.** Features of Solving the Enterprise Investment Portfolio Optimization Problem Using the Particle Swarm Method. *Izvestiya Tul'skogo gosudarstvennogo universiteta. Tekhnicheskiye nauki* [Izvestiya of Tula State University. Technical Sciences]. 2016. No 5. Pp. 109–116. (In Russ.)

10. **Esikov O. V., Rumyantsev V. L., Starozhuk E. A.** Application of Swarm Algorithms for Solving the Problem of Selecting Operating Frequencies of Radio Equipment of the Air Traffic Control System. *Izvestiya Tul'skogo gosudarstvennogo universiteta. Tekhnicheskiye nauki* [Izvestiya of Tula State University. Technical Sciences]. 2016. No 2. Pp. 85–92. (In Russ.)

11. **Pyankov O. V., Popov A. V.** Decision-Making Model for Improving the Responsiveness of Police Detention Groups Using Swarm Algorithms. *Vestnik Voronezhskogo instituta MVD Rossii* [Bulletin of the Voronezh Institute of the Ministry of Internal Affairs of Russia]. 2020. No 4. Pp. 73–83. (In Rus.)

12. **Akhmadiev F. G., Malanichev I. V.** Population Algorithms of Structural and Parametric Optimization in Construction Design. *Izvestiya Kazanskogo gosudarstvennogo arkhitekturno-stroitel'nogo universiteta* [Izvestiya of Kazan State University of Architecture and Engineering] 2018. No 2 (44). Pp. 215–223. (In Rus.)

13. **Rahmatulloh A., Nugraha G. F., Darmawan I.** Hybrid PSO-Adam Optimizer Approach for Optimizing Loss Function Reduction in the Dist-YOLOv3 Algorithm. *International Journal of Intelligent Engineering and Systems*. 2024. Vol. 17. No 5. Pp. 199–209. DOI: 10.22266/ijies2024.1031.16.

14. **Panteleev A.V., Belyakov I.A.** Development of Software for a Global Optimization Method Simulating the Behavior of a Grey Wolf Pack. *Modelirovaniye i analiz dannykh* [Modeling and Data Analysis]. 2021. No 2. Pp. 59–73. DOI: 10.17759/mda.2021110204. (In Rus.)

15. **Khishe M., Mosavi M. R.** Chimp optimization algorithm. *Expert Systems with Applications*. 2020. Vol. 149. P. 113338. DOI: 10.1016/j.eswa.2020.113338.

16. **Mittal N., Singh U., Sohi B. S.** Modified grey wolf optimizer for global engineering optimization. *Applied Computational Intelligence and Soft Computing*. 2016. Article ID 7950348. (4598). Pp. 1–16. DOI: 10.1155/2016/7950348.

17. **Yang X.-S.** A New Metaheuristic Bat-Inspired Algorithm. *Nature Inspired Cooperative Strategies for Optimization (eds. J. R. Gonzalez et al.)*, Studies in Computational Intelligence, Springer Berlin, 284, Springer, 2010. Pp. 65–74. DOI: 10.1007/978-3-642-12538-6_6.

18. **Mirjalili S., Lewis A.** The Whale Optimization Algorithm. *Advances in Engineering Software*. 2016. No 95. Pp. 51–67. DOI: 10.1016/j.advengsoft.2016.01.008.

19. **Mirjalili S.** SCA: a sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* 2016. Vol. 96. Pp. 120–133. DOI: 10.1016/j.knosys.2015.12.022.

20. **Pilcevic D., Djuric Jovicic M., Antonijevic M. et al.** Performance evaluation of metaheuristics-tuned recurrent neural networks for electroencephalography anomaly detection. *Front Physiol.* 2023. Nov 14; 14: 1267011. DOI: 10.3389/fphys.2023.1267011.

21. **Almufti S. M., Asaad R. R., Salim B. W.** Review on Elephant Herding Optimization Algorithm Performance in Solving Optimization Problems. *International Journal of Engineering & Technology.* 2018. 7 (4). Pp. 6109–6114. DOI: 10.14419/ijet.v7i4.23127.

Сведения об авторах / Information about authors
Бабикова Надежда Николаевна / Nadezhda N. Babikova
к.пед.н., доцент, доцент кафедры прикладной информатики / Candidate of Sciences in Pedagogical, Associate Professor, Associate Professor of Applied Informatics Department
Сыктывкарский государственный университет имени Питирима Сорокина / Pitirim Sorokin Syktyvkar State University
Россия, 167001, г. Сыктывкар, Октябрьский пр., 55 / 55, Oktyabrsky Ave., Syktyvkar, 167001, Russia

Котелина Надежда Олеговна / Nadezhda O. Kotelina
к.ф.-м.н., доцент кафеды прикладной математики и компьютерных наук /Candidate of Science in Physics and Mathematics, Associate Professor of Department of Applied Mathematics and Computer Science
Сыктывкарский государственный университет имени Питирима Сорокина / Pitirim Sorokin Syktyvkar State University
Россия, 167001, г. Сыктывкар, Октябрьский пр., 55 / 55, Oktyabrsky Ave., Syktyvkar, 167001, Russia

Статья поступила в редакцию / The article was submitted 28.11.2025
Одобрена после рецензирования / Approved after reviewing 01.12.2025
Принята к публикации / Accepted for publication 05.12.2025