Никифорова Анна, 9302

Исследование и реализация алгоритма Snow Ablation Optimizer (SAO)

1. Теоретическое обоснование алгоритма Snow Ablation Optimizer (SAO)

Алгоритм Snow Ablation Optimizer (SAO) основан на процессах испарения и таяния снега. Данный алгоритм состоит из двух ключевых стадий – исследование и эксплуатация. Подобный подход позволяет предотвратить преждевременную сходимость алгоритма. Алгоритм SAO применим в решении задач оптимизации и в инженерном проектировании. Впервые данный алгоритм был предложен китайскими исследователями Линьюнь Дэном и Саньяном Ли в 2023 году. SAO относится к алгоритмам роевого интеллекта [1].

Если стадия исследования направлена на нахождение агентами поиска несколько локальных минимум, то стадия эксплуатации служит для быстрого достижения сходимости к глобальному оптимальному решению. Иными словами, алгоритм SAO призван найти баланс между разнообразием роя и его преждевременной сходимостью.

В основе алгоритма лежит физический процесс сублимации снега на стадии исследования и процессах таяния снега, а также испарения на стадии эксплуатации. Данный процесс представлен на рисунке 1.

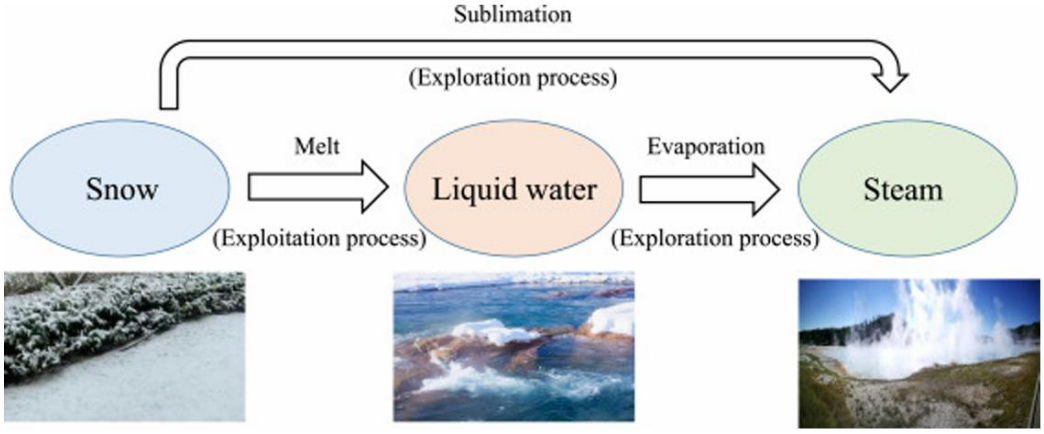

Рисунок 1 – Физические процессы, протекающие в снеге

Алгоритм SAO можно свести к двум большим этапам – инициализация двух подпопуляций агентов поиска и непосредственно сам алгоритм SAO. Представим псевдокод инициализации популяции.

Алгоритм 1. Инициализация двух популяций

Вход: N - размер популяции.

Выход: две случайно сгенерированные подпопуляции одинакового размера.

1. Initialization: $t = 0, t_{max}, t_{min}, N_a = N_b = \frac{N}{2}$, где N обозначает размер популяции
2. while ($t < t_{max}$) do
3. if $N_a < N$ then
4. $N_a = N_a + 1$, $N_b = N_b + 1$
5. end if
6. $t = t + 1$
7. end while

Популяция P разделяется на две подпопуляции одинакового размера – P_a и P_b с целью сохранения баланса между стадией исследования и эксплуатации. P_a участвует в стадии исследования, а P_b - в стадии исследования.

Популяция представляет собой матрицу $N * D$, где N – размер роя, а D – размерность пространства поиска. Матрица имеет следующий вид:

$$Z = L + \phi * (U - L) = \begin{bmatrix} z_{1,1} & \cdots & z_{1,Dim} \\ \vdots & \ddots & \vdots \\ z_{N,1} & \cdots & z_{N,Dim} \end{bmatrix}_{N*Dim}, \quad (1)$$

где L и U обозначают нижнюю и верхнюю границы пространства поиска, а ϕ – случайное число, полученное в диапазоне [0,1].

На стадии исследования снег превращается в пар, а агенты поиска децентрализованы и характеризуются неравномерным движением. Для описания данного физического процесса используется броуновское движение:

$$f_{BM}(x;0,1) = \frac{1}{\sqrt{2\pi}} * \exp(-\frac{x^2}{2}) \quad (2)$$

Броуновское движение позволяет исследовать потенциальные области в пространстве поиска и описывает распространение пара в пространстве поиска. Для вычисления позиций агентов поиска на стадии исследования используется следующая формула [2]:

$$Z_i(t+1) = Elite(t) + BM_i(t) \oplus \left(\theta_1\left(G(t) - Z_i(t)\right) + (1 - \theta) * \left(\bar{Z}(t) - Z_i(t)\right)\right), \quad (3)$$

где $Z_i(t)$ - i-ый индивид на i-ой итерации; $BM_i(t)$ - вектор, содержащий случайные числа на основе нормального распределения и обозначающий броуновское движение; θ – случайное число из диапазона [0,1]; *G(t)* – текущее наилучшее решение; *Elite(t)* – случайная особь из набора нескольких элит в рое; $\bar{Z}(t)$ – положение центра тяжести всего роя.

Для вычисления положения центра тяжести всего роя используется следующая формула:

$$\bar{Z}(t) = \frac{1}{N}\sum_{i=1}^{N} Z_i(t), Elite(t) \in [G(t), Z_{second}(t), Z_{third}(t), Z_c(t)], \quad (4)$$

где $Z_{second}(t)$ и $Z_{third}(t)$ – второй и третий лучший индивид в текущей популяции; $Z_c(t)$ – центроидное положение индивидуумов с высокими физическими показателями. Для упрощения вычислений первые 50% индивидуумов признаются лидерами.

Для вычисления центроидного положения индивидуумов с высокими физическими показателями используется следующая формула:

$$Z_c(t) = \frac{1}{N_1}\sum_{i=1}^{N_1} Z_i(t), \quad (5)$$

где N_1 – количество лидеров (половина размера всего роя), $Z_i(t)$ – i-ый лучший лидер. На каждой итерации $Elite(t)$ выбирается случайным образом из набора, состоящего из наилучшего решения, второго по результативности

индивидуума, третьего по результативности индивидуума и центральной позиции лидеров.

На стадии эксплуатации агенты поиска находят наилучшие решения. Данная стадия основана на физическом процессе таяния снега. Для моделирования данного процесса используется **методика подсчета градусо-дней**:

$$M = DDF * (T - T_1), \tag{6}$$

где M –скорость таяния снега, T – средняя дневная температура, T_1-базовая температура, обычно устанавливаемая в 0. Таким образом, формула приобретает следующий вид:

$$M = DDF * T,$$

где *DDF* обозначает коэффициент градусо-дней в диапазоне [0.35, 0.6].

На каждой итерации значение DDF обновляется следующим образом:

$$DDF = 0.35 + 0.25 * \frac{e * \frac{t}{t_{max}} - 1}{e - 1} * T(t), T(t) = e^{\frac{-t}{t_{max}}}, \tag{6}$$

На стадии эксплуатации решается следующее уравнение обновления позиций агентов:

$$Z_i(t+1) = M * G(t) + BM_i(t) \oplus \left(\Theta_2 * \left(G(t) - Z_i(t)\right) + (1 - \Theta_2) * \left(\bar{Z}(t) - Z_i(t)\right)\right), \tag{7}$$

где M – скорость таяния снега, Θ_2 – случайное число в диапазоне [-1, 1], $\Theta_2 * \left(G(t) - Z_i(t)\right)$ и $(1 - \Theta_2) * \left(\bar{Z}(t) - Z_i(t)\right)$ – перекрестные члены, необходимые для поиска решений в пространствах поиска.

Таким образом, алгоритм SAO можно представить следующим образом:

$$Z_i(t+1) = \begin{cases} Elite(t) + BM_i(t) \oplus \left(\Theta_1\left(G(t) - Z_i(t)\right) + (1 - \Theta) * \left(\bar{Z}(t) - Z_i(t)\right)\right), \\ i \in index_a \\ M * G(t) + BM_i(t) \oplus \left(\Theta_2 * \left(G(t) - Z_i(t)\right) + (1 - \Theta_2) * \left(\bar{Z}(t) - Z_i(t)\right)\right), \\ i \in index_b \end{cases}. \tag{7}$$

Представим псевдокод алгоритма SAO.

Алгоритм 2. Алгоритм SAO

Вход: N - размер популяции.

Выход: две случайно сгенерированные подпопуляции одинакового размера.

1. Initialization: swarm $Z_i(i = 1,2, \dots, N), t = 0, t_{max}, N_a = N_b = \frac{N}{2}$
2. Fitness evaluation
3. Record the current best individual $G(t)$
4. **while** ($t < t_{max}$) **do**
5. Calculate the snowmelt rate *M*
6. Randomly divide the whole population *P* into two subpopulations P_a and P_b
7. **for** each individual **do**
8. Update each individual's position
9. **end for**
10. Fitness evaluation
11. Update *G(t)*
12. t = t + 1
13. **end while**
14. Return *G(t)*

Для вычисления скорости таяния снега на шаге 5 используется следующая формула, которая выводится из формулы (6):

$$\mathrm{M} = \left(0.35 + 0.25 * \frac{e * \frac{t}{t_{max}} - 1}{e - 1}\right) * T(t), \text{где} T(t) = e^{\frac{-t}{t_{max}}}, \qquad (8)$$

На шаге 8 обновление позиций каждого индивидуума производится с помощью формулы (7).

2. Исследование алгоритма SAO

Алгоритм SAO был реализован на языке программирования Python с целью исследования его эффективности его работы при поиске минимума функции. Алгоритм был протестирован на 6 различных тестовых функциях. Приведем примеры данных функций. Полный проект доступен по соответствующей ссылке [3]. Был реализован в текстовом редакторе Visual Studio Code на Python 3.8.0.

Функция Химмельбрау представляет собой мультимодальную функцию двух переменных и определяется формулой [4]:

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2. \quad (9)$$

Функция имеется четыре равнозначных локальных минимума:

- $f(3{,}2) = 0,$
- $f(-2{,}805118\ldots, 3.131312\ldots) = 0,$
- $f(-3{,}779310\ldots, -3{,}283186\ldots) = 0,$
- $f(3{,}584428\ldots, -1{,}848126\ldots) = 0.$

График нахождения локального минимума функции показан на рисунке 2.

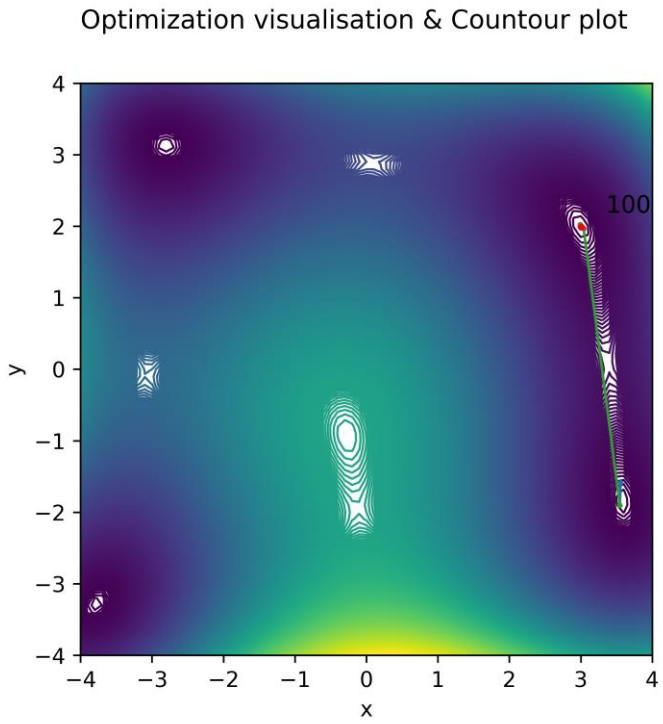

Рисунок 2 – Результат нахождения локального минимума функции Химмельбрау на диапазоне [-4, 4]

В результате работы алгоритма был найден минимум в точке xmin = (3.004962015164989 , 1.9888963735952516) со значением функции f_min = 0.0018962174384222292), что соответствует локальному минимуму $f(3{,}2) = 0$. Были заданы параметры: SearchAgents_no = 200, Max_iteration = 100, диапазон [-4, 4].

При задании другого диапазона был найден другой локальный минимум x_min = (-2.8226935592873694 , 3.133119746859477), f_min = 0.010183366953105233, что соответствует локальному минимуму $f(-2{,}805118\ldots, 3.131312\ldots) = 0$. Как видно из результатов, удалось достичь точность вычислений 10^{-1} для x и 10^{-2} для y. Результат работы алгоритма показан на рисунке 3.

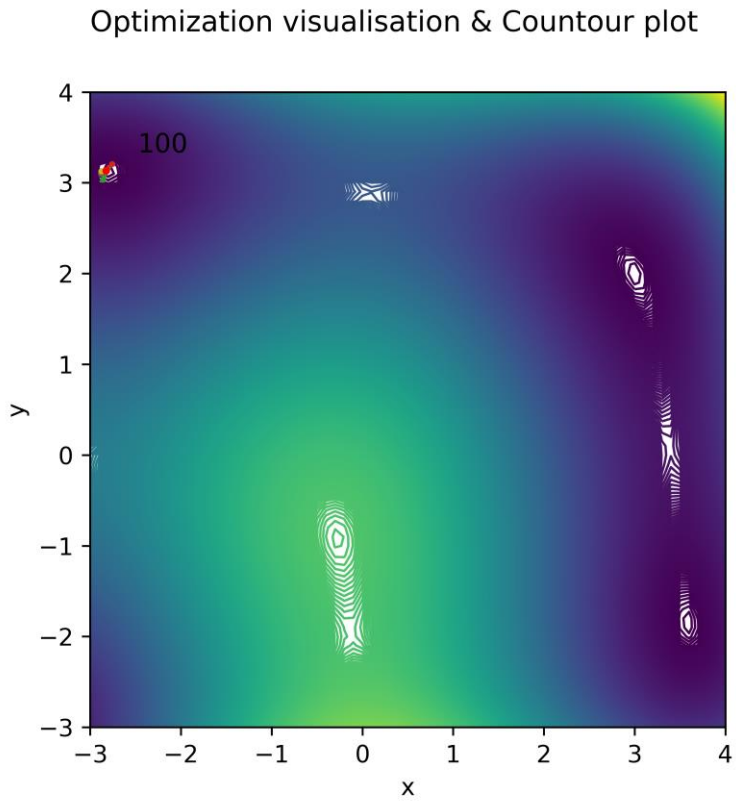

Рисунок 3 – Результат работы алгоритма на примере функции Химмельбрау на диапазоне [-3, 4]

Функция Розенброка представляет собой невыпуклую функцию. Данная функция имеет узкие долины, в которых трудно найти глобальный минимум. Имеет локальный минимум в точке $(x, y = (1{,}1)$, где $f(x, y) = 0$. Определяется следующим образом:

$$f(x,y) = (1 - x^2)^2 + 100(y - x^2)^2. \quad (10)$$

Наиболее оптимальных результатов удалось достичь при следующих параметрах: SearchAgents_no = 100, Max_iteration = 100, диапазон [-2, 2]. Был найден минимум в точке x_min = (1.0134041589034808, 1.0284917440860266) со значением f_min = 0.00040579932671104. Удалось достичь точности 10^{-1}. При увеличении числа итераций до 300 теряется точность вычислений: при SearchAgents_no = 100, Max_iteration = 300, диапазон [-2, 2] был найден минимум в точке x_min = (0.9969552191143654, 0.9951433211800746), f_min = 0.00015899338709566082 за 100 шагов.

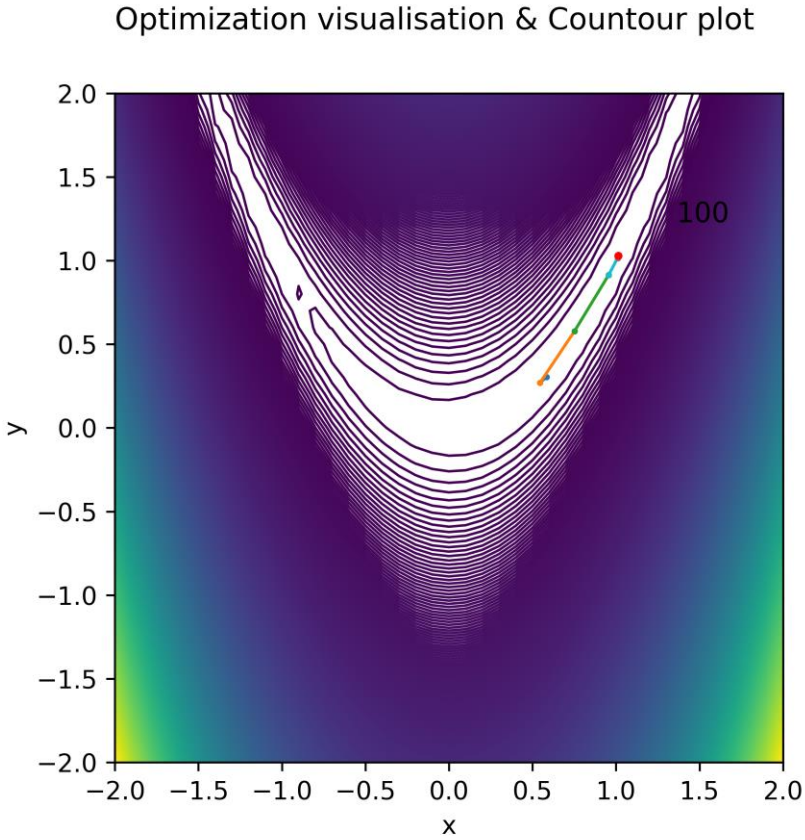

Рисунок 3 – Результат работы алгоритма на примере функции Розенброка на диапазоне [-2, 2]

Сферическая функция имеет один глобальный минимум в точке $f(x) = 0, \text{при } x = (0, \dots, 0)$. Она непрерывна, выпукла и определяется следующей формулой:

$$f(x) = \sum_{i=1}^{d} x^2{}_i. \quad (11)$$

Данная функция гладкая и унимодальная. Обладает одним глобальным минимумом.

Был найден минимум в точке x_min = (-2.2275964555040132e-11, -3.7236982367255786e-11), f_min = 1.8828114526767226e-21 при заданных параметрах SearchAgents_no = 100, Max_iteration = 100, диапазон [-1, 1]. Стоит отметить, что алгоритм SAO хорошо справляется с нахождением минимума в точке (0,0). Удалось достичь точность вычислений 10^{-10}.

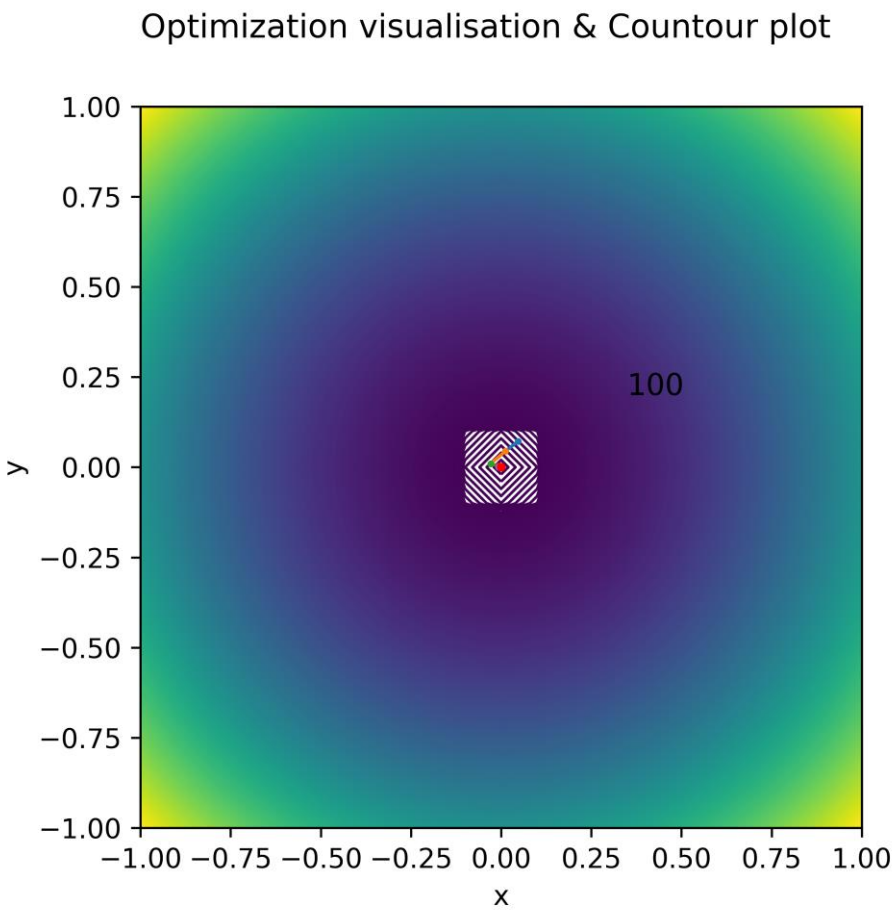

Рисунок 4 – Результат работы алгоритма на примере сферической функции на диапазоне [-1, 1]

При увеличении числа итераций алгоритм справляется лучше с нахождением глобального минимума функции сферы. Так, при задании максимального количества итераций 500 был найден минимум в точке x_min = (1.1448038935148984e-38, -1.2933363099038383e-38), f_min =

2.983294765122548e-76. Удалось достичь точность в 10^{-37}. Результат работы алгоритма показан на рисунке 5.

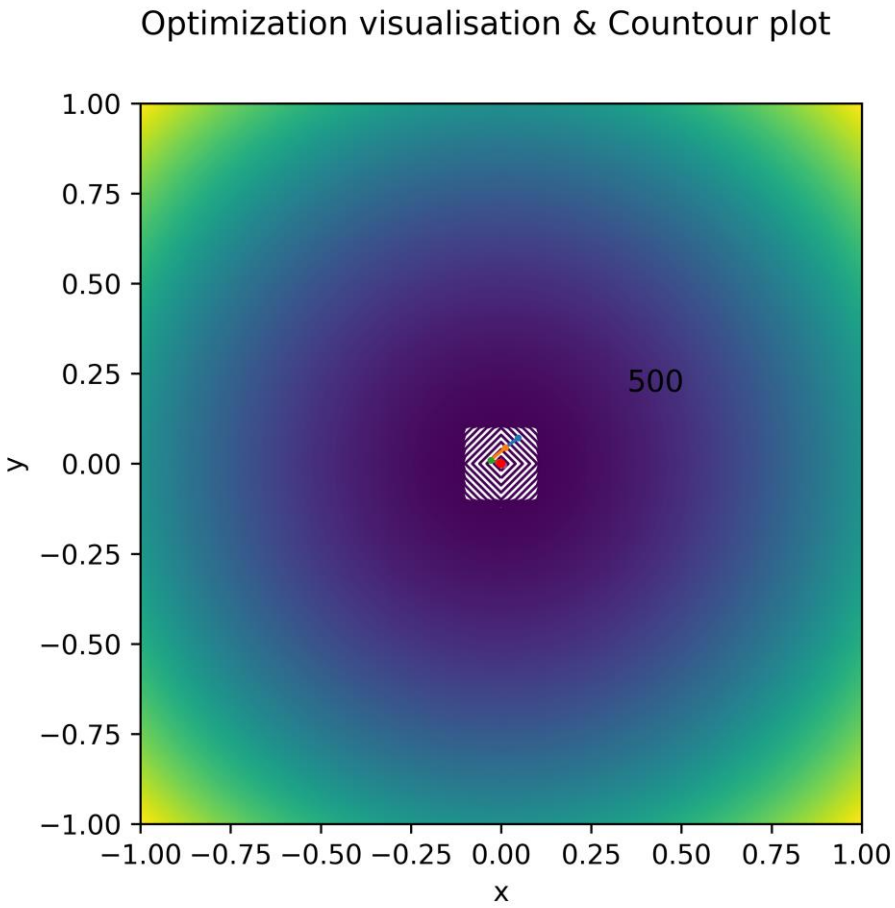

Рисунок 4 – Результат работы алгоритма на примере сферической функции на диапазоне [-1, 1] на 500 итерациях

Функция Михалевича – это мультимодальная функция. Она сложна для минимизации, т.к. обладает несколькими локальными минимумами и несколькими плоскими областями, что затрудняет поиска одного глобального минимума функции. У функции Михалевича есть свободный параметр m, который контролирует крутизну долин. Большие значения m делают функцию более сложной для оптимизации. Наиболее распространённое значение для m — 10. Функция определяется следующим образом:

$$f(x) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}(\frac{ix_i^2}{\pi}). \quad (12)$$

Поскольку функция обладает «крутыми долинами», для поиска ее минимума и подходят метаэвристические алгоритмы.

Глобальный минимум функции $f(x) = -1.8013$ в точке x (2.20,1.57).

Был найден минимум в x_min = (2.216606081713704, 1.5671971060840053), f_min = -1.7977167976479416 при заданных параметрах SearchAgents_no = 100, Max_iteration = 100, диапазон [0, 4]. Удалось достичь точность вычислений 10^{-1}. Результат работы алгоритма показан на рисунке 5.

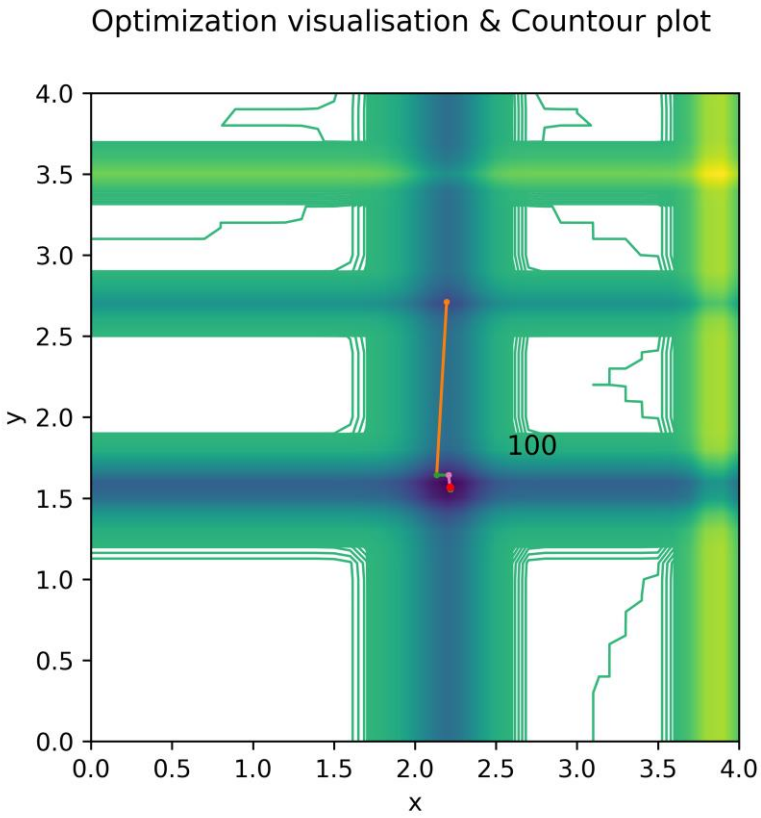

Рисунок 5 - Результат работы алгоритма на примере сферической функции на диапазоне [0, 4] на 100 итерациях

Стоит отметить, что алгоритм SAO хорошо справляется с нахождением минимума в точке (0,0). Удалось достичь точность вычислений 10^{-10}. Однако при параметрах SearchAgents_no = 100, Max_iteration = 400, диапазон [0, 4] удалось достичь точности 10^{-1} при x_min = (2.2048649098914597 , 1.5726717295941484), f_min = -1.8010983590858507. Результат работы алгоритма показан на рисунке 6.

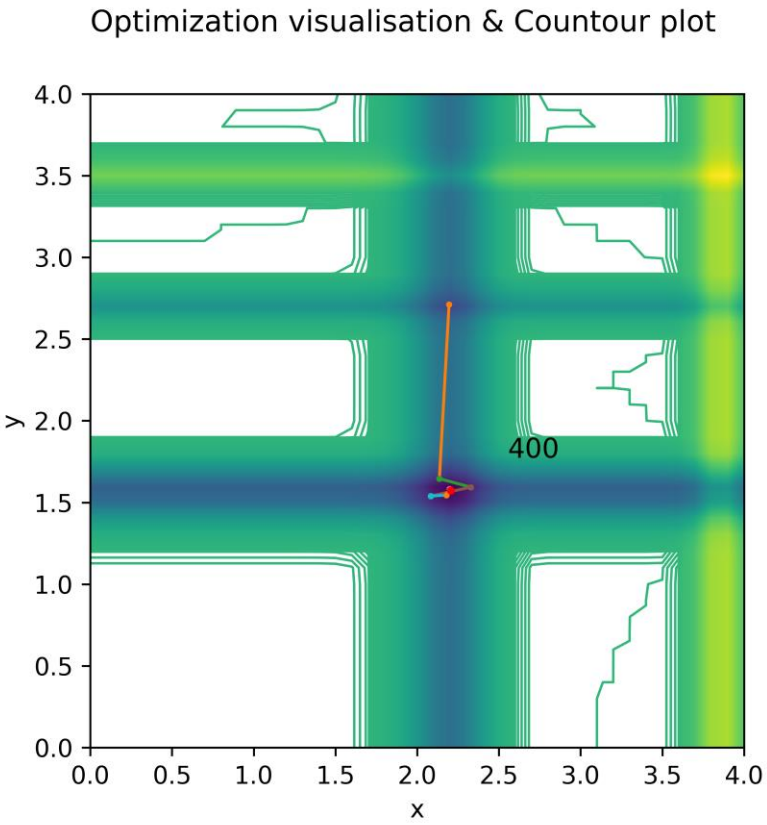

Рисунок 6 - Результат работы алгоритма на примере сферической функции на диапазоне [0, 4] на 400 итерациях

Функция Захарова имеет только один глобальный минимум в точке $f(x) = 0$ в точке $x(0, \dots, 0)$.

$$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4. \quad (13)$$

Был найден минимум в x_min = (3.9582117071204123e-10, -1.1273854644619378e-10), f_min = 1.7663865525070851e-19 при заданных параметрах SearchAgents_no = 100, Max_iteration = 100, диапазон [-5, 10]. Удалось достичь точность вычислений 10^{-9}. Результат работы алгоритма показан на рисунке 7.

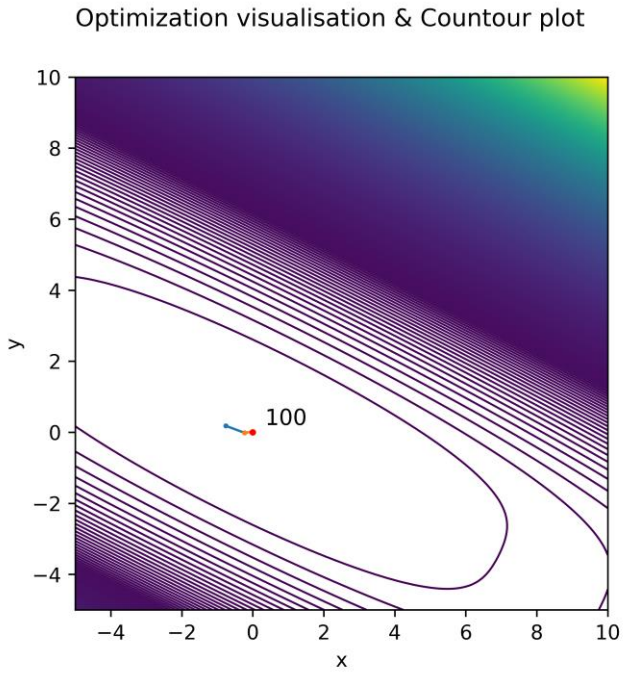

Рисунок 7 - Результат работы алгоритма на примере функции Захарова на диапазоне [-5, 10] на 100 итерациях

Однако при параметрах SearchAgents_no = 100, Max_iteration = 500, диапазон [-5, 10] удалось достичь точности 10^{-37} при x_min = (-2.3233707458124533e-38, 7.570570888754846e-40), f_min = 6.583134807517403e-76 Результат работы алгоритма показан на рисунке 8.

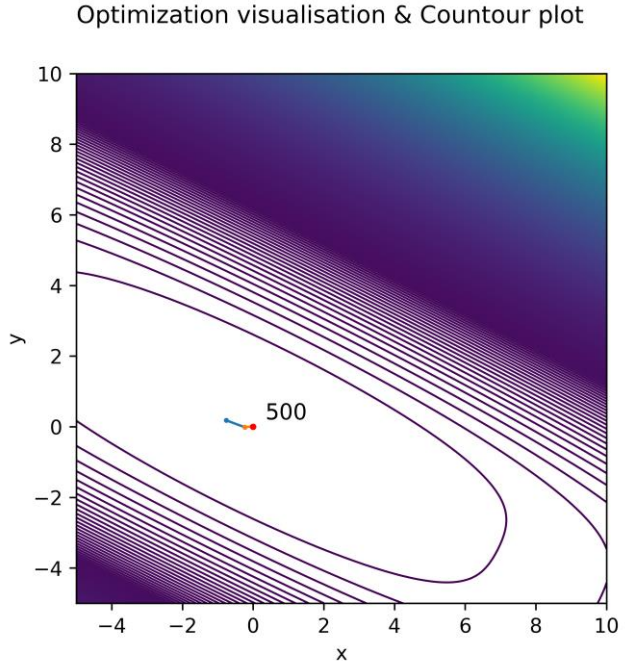

Рисунок 8 - Результат работы алгоритма на примере функции Захарова на диапазоне [-5, 10] на 500 итерациях

Функция Гриуэнка имеет множество локальных минимумов. Имеет один глобальный минимум $f(x) = 0$ при $x = (0, \dots, 0)$. Функция определяется следующим образом:

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \left(\frac{x_i}{\sqrt{i}}\right) + 1. \quad (14)$$

Был найден минимум в x_min = (2.97500906453891e-09, -9.289639364840276e-09), f_min = 0.0 при параметрах SearchAgents_no = 100, Max_iteration = 100, диапазон [-50, 50]. Удалось достичь точность вычислений 10^{-8}. Результат работы алгоритма показан на рисунке 9.

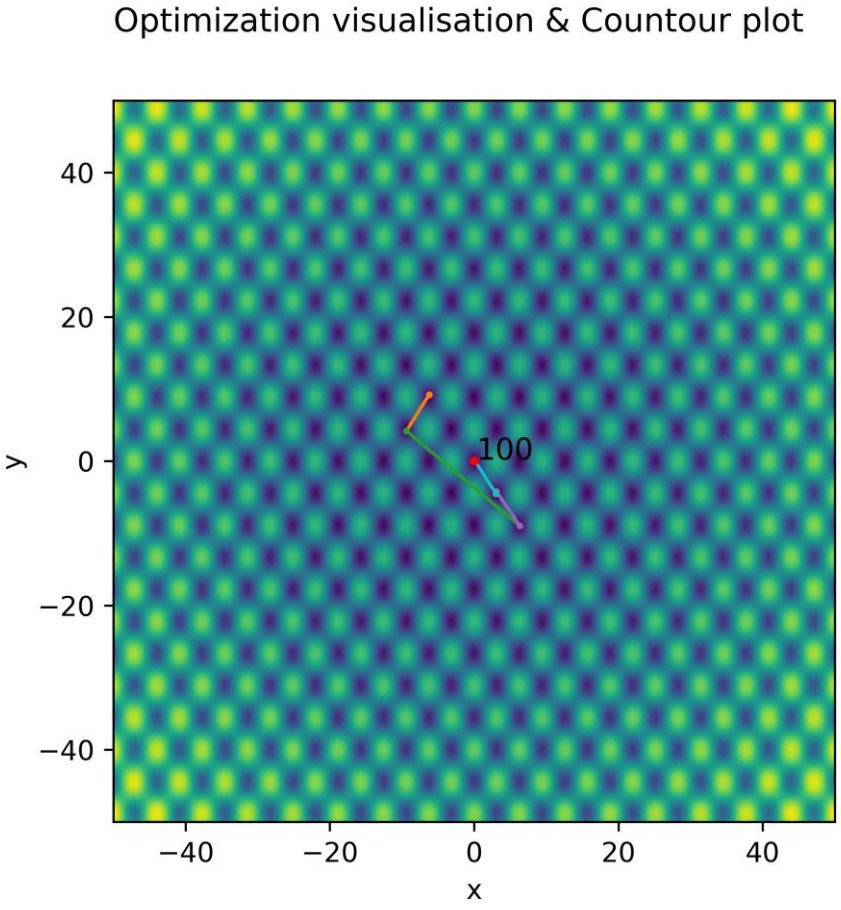

Рисунок 9 - Результат работы алгоритма на примере функции Гриуэнка на диапазоне [-50, 50] на 100 итерациях

Однако при параметрах SearchAgents_no = 100, Max_iteration = 500, диапазон [-50, 50] не удалось достичь большей точности вычислений. Был найден минимум в точке 10^{-37} при x_min = (5.6183702238020725e-09,

-5.594429732884627e-09), f_min = 0.0 , neval = 500. Результат работы алгоритма показан на рисунке 10.

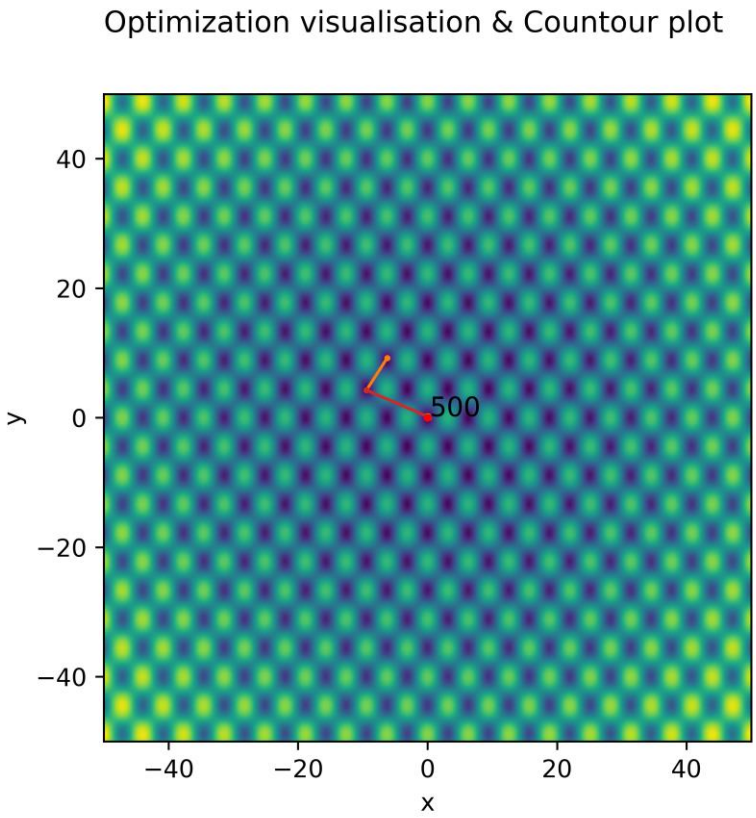

Рисунок 10 - Результат работы алгоритма на примере функции Гриуэнка на диапазоне [-50, 50] на 500 итерациях

3. Оценка сходимости функций

Для более точного определения точности сходимости функций в задаче поиска минимума с помощью алгоритма SAO был построен 3D-график зависимости *погрешности вычислений* от *количества агентов и количества итераций*.

Представим графики для вычисления минимума различных функций:

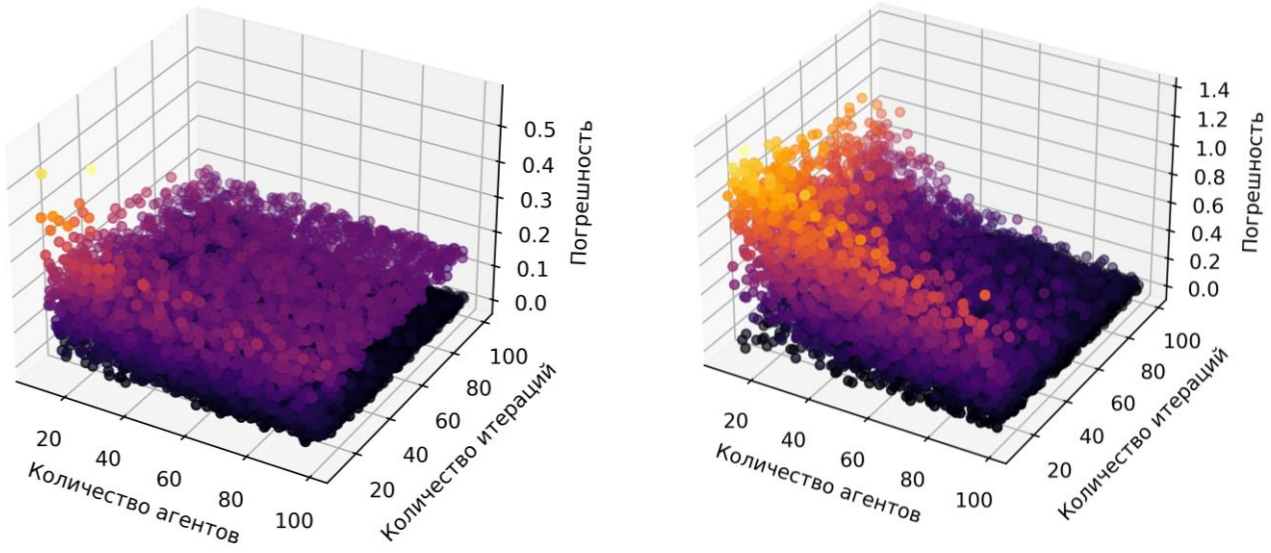

Рисунок 11 – Сравнение точности сходимости алгоритма для поиска минимума функции Розенброка и значения функции (слева направо)

Как видно из графиков, для функции Розенброка погрешность вычислений при поиске значения функции выше, чем при поиске минимума функции.

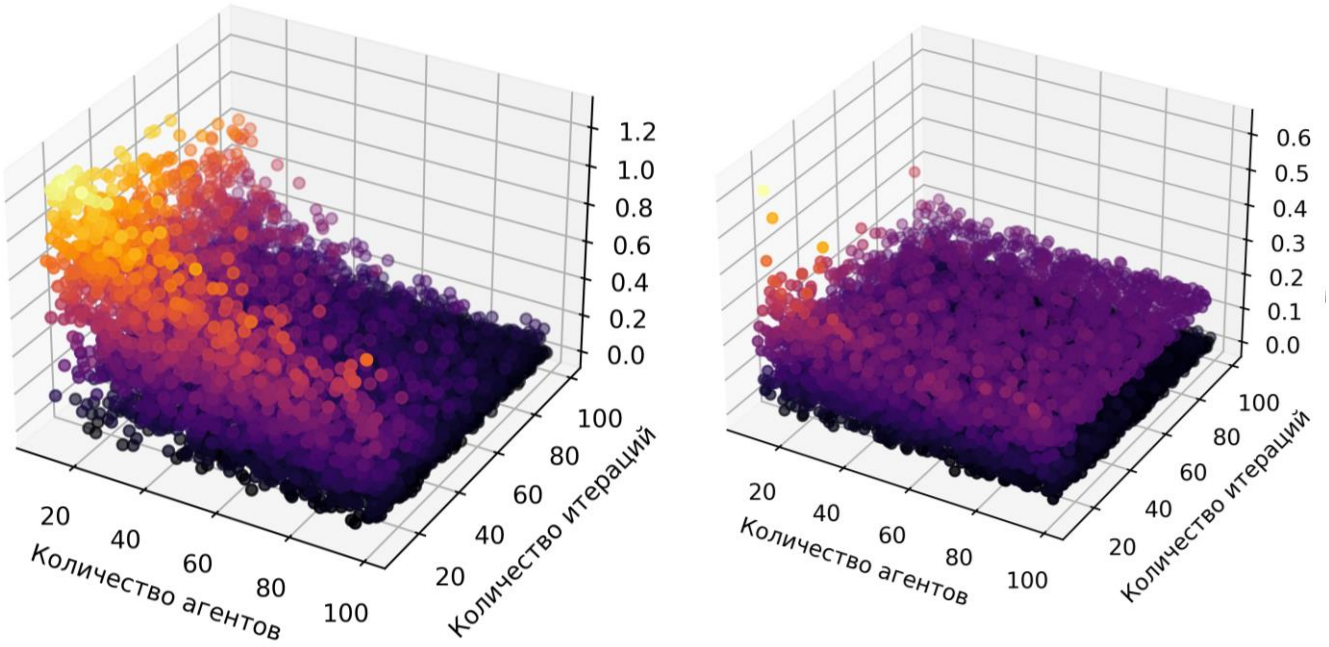

Рисунок 12 – Сравнение точности сходимости алгоритма для поиска минимума функции Химмельбрау и значения функции

Для функции Химмельбрау погрешность вычислений при поиске значения функции ниже, чем при поиске минимума функции.

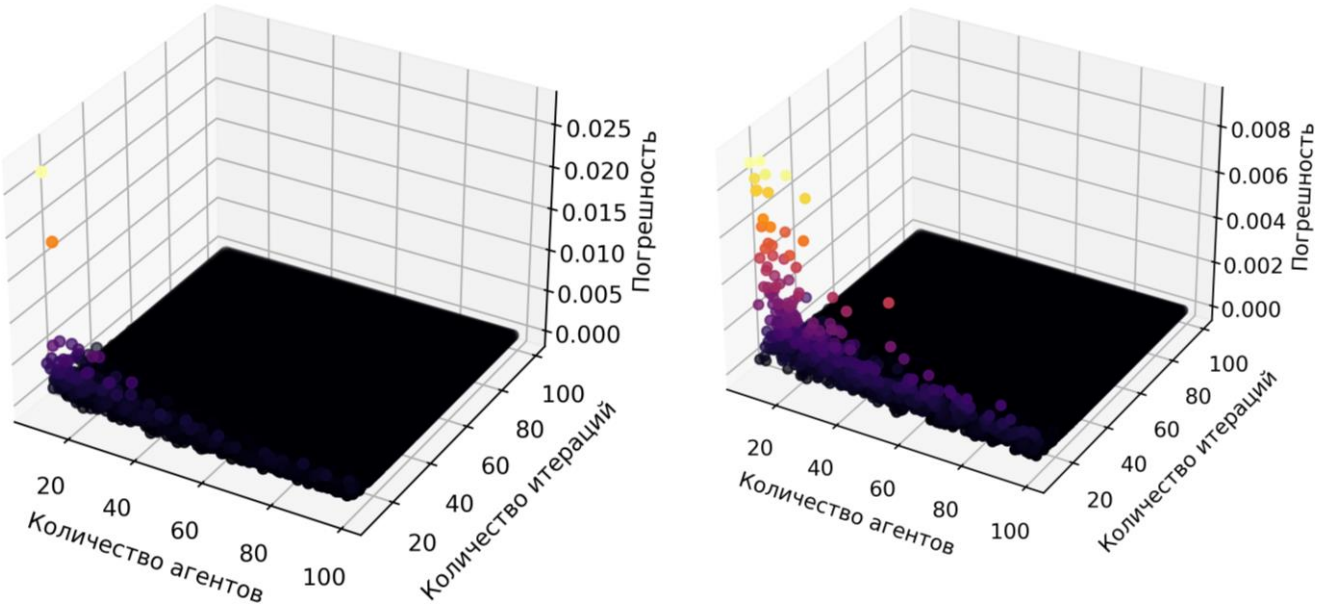

Рисунок 13 – Сравнение точности сходимости алгоритма для поиска минимума функции сферы и значения функции

Для функции сферы погрешность вычислений при поиске значения функции выше, чем при поиске минимума функции. Погрешность настолько мала, что это не сказывается на точности вычислений. Это может объясняться формой функции в виде гладкого шара.

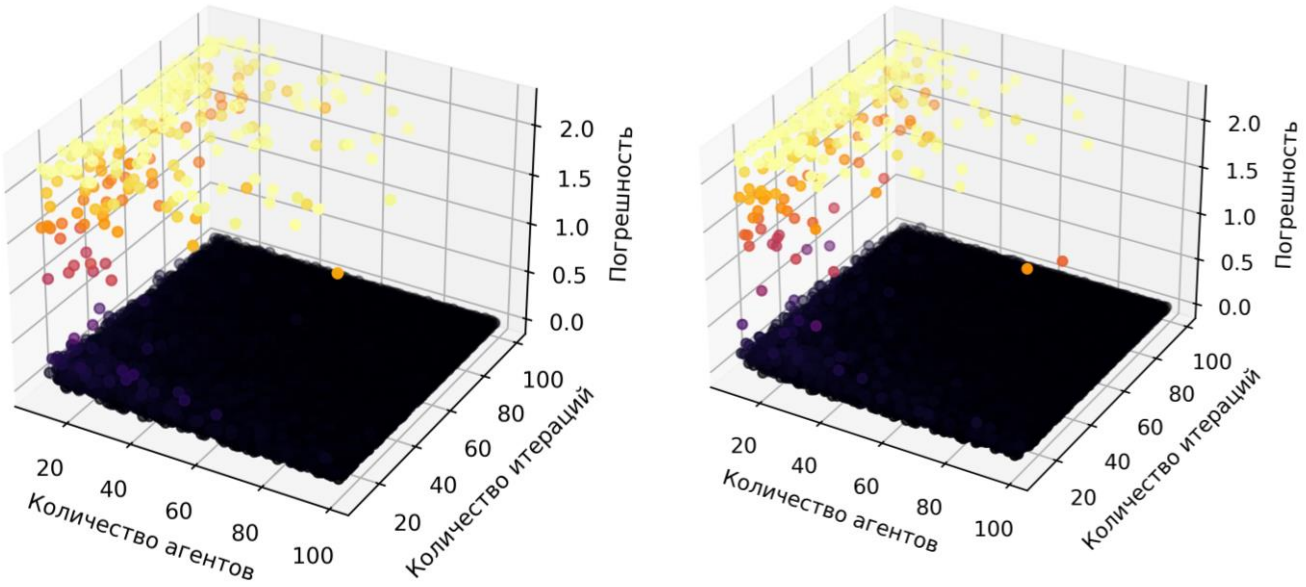

Рисунок 14 – Сравнение точности сходимости алгоритма для поиска минимума функции Михалевича и значения функции

Для функции Михалевича почти не заметны различия в точности при вычислении минимума функции или ее значения. Однако стоит отметить, что при малом количестве агентов погрешность вычислений довольно высока.

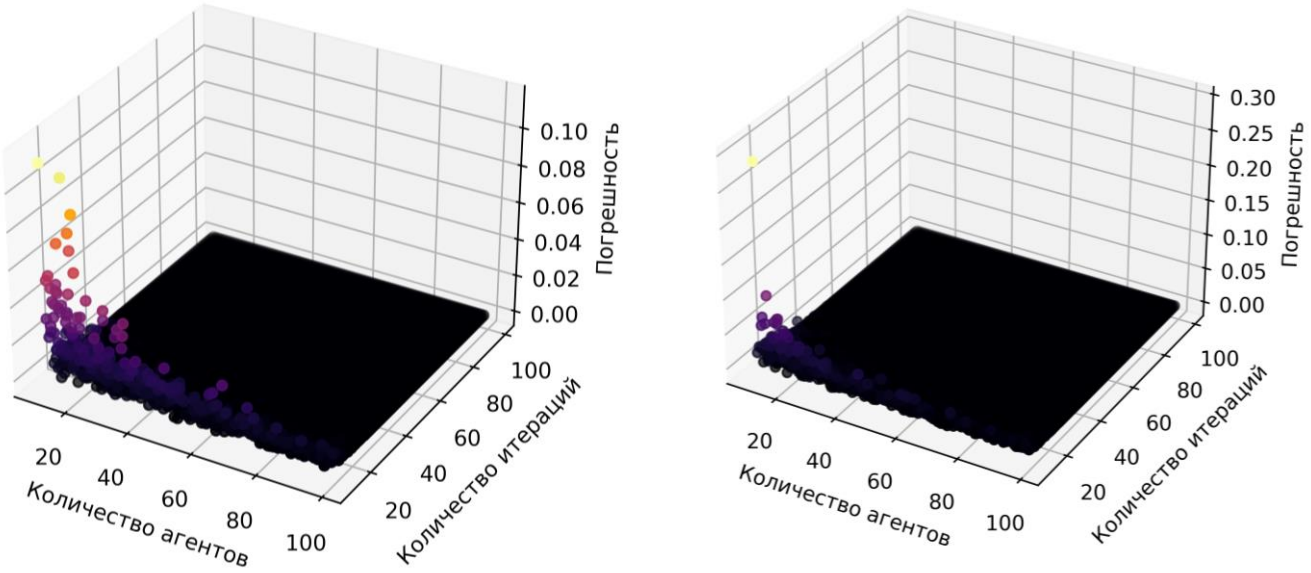

Рисунок 15 – Сравнение точности сходимости алгоритма для поиска минимума функции Захарова и значения функции

Для функции Захарова точность вычислений при вычислении минимума функции и ее значения довольно мала, что объясняется ее регулярной структурой, а именно формой в виде гладкой чашей с глобальным минимумом.

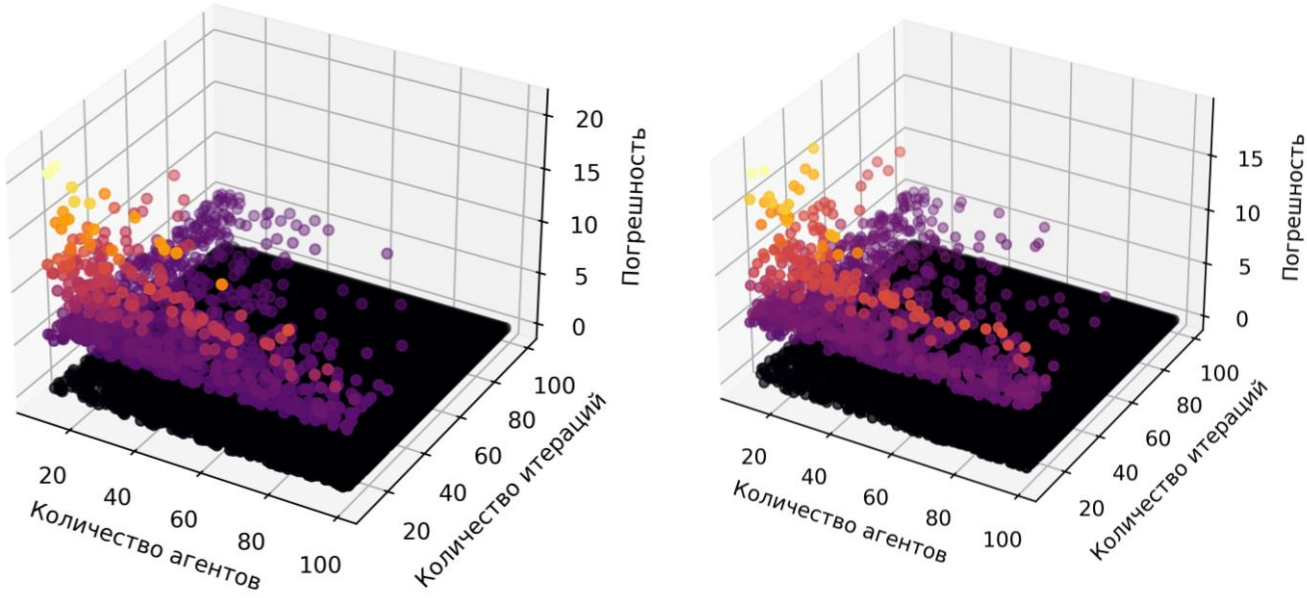

Рисунок 16 – Сравнение точности сходимости алгоритма для поиска минимума функции Гриуэнка и значения функции

Для функции Гриуэнка точность вычислений при вычислении значения функции выше, чем при вычислении ее минимума. Погрешность вычислений при малом количестве итераций и малом количестве агентов довольно значительная.

Кроме того, были построены двумерные графики для более наглядного представления точности вычислений и сходимости функции. Опишем некоторые из них.

График зависимости погрешности вычислений от количества агентов и итераций для функции Химмельбрау имеет следующий вид:

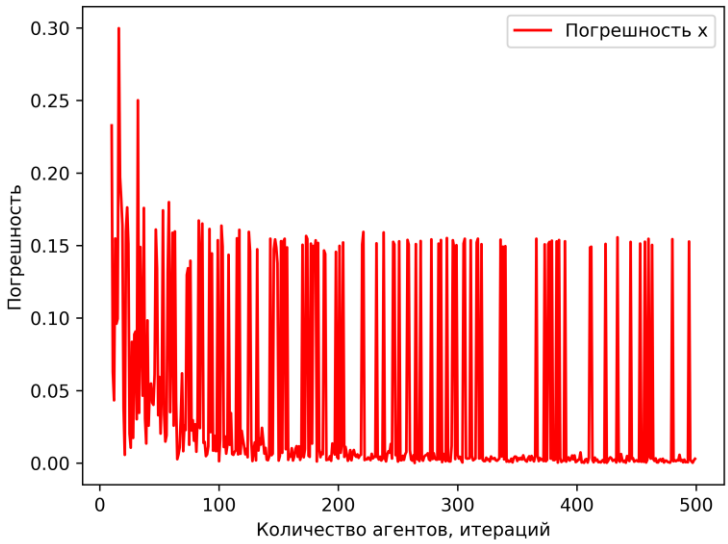

Рисунок 17 – Результат поиска минимума функции Химмельбрау

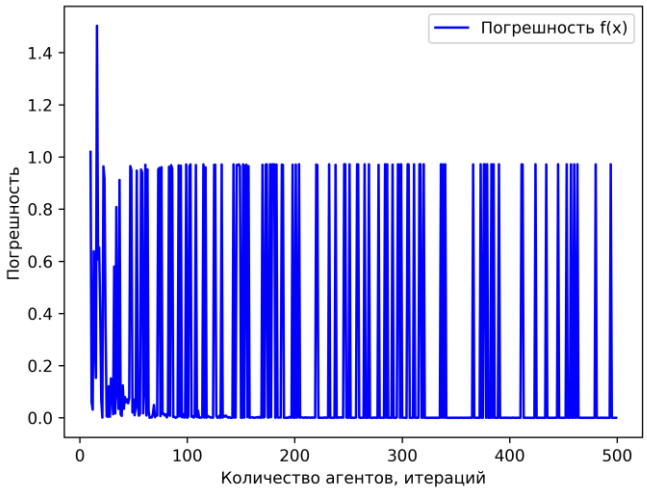

Рисунок 18 – Результат вычислений значения функции Химмельбрау

Как видно из рисунков, при 100 агентах и итераций погрешность вычислений перестает улучшаться. Это означает, что оптимальное количество агентов и итераций – 100. Однако на любом этапе алгоритм может не вычислить значение.

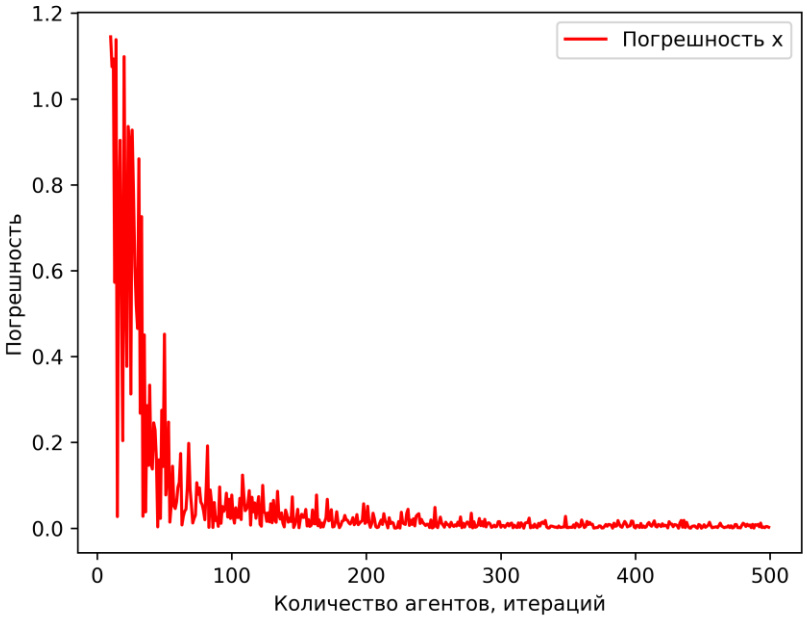

Рисунок 19 - Результат поиска минимума функции Розенброка

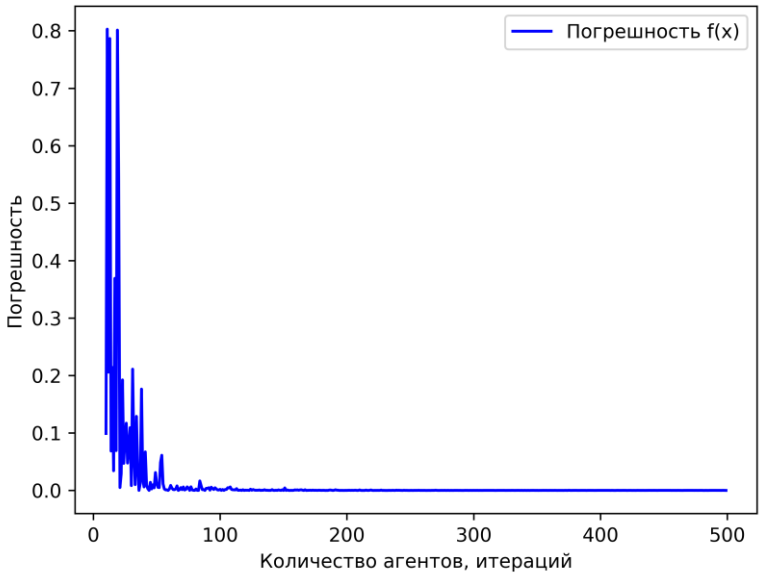

Рисунок 20 - Результат вычисления значения функции Розенброка

Как видно из рисунков, функция Розенброка более стабильна, чем функция Химмельбау. При 100 агентах и итерации она начинает давать стабильные результаты.

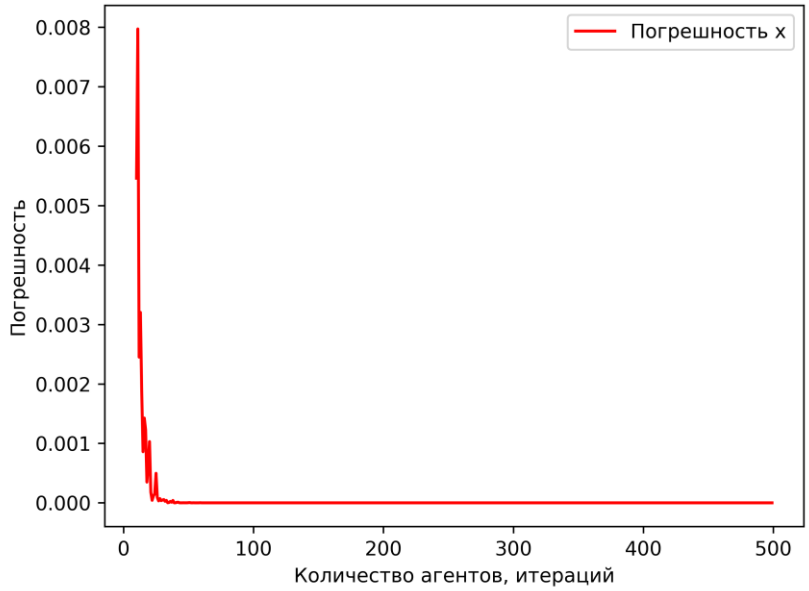

Рисунок 21 - Результат поиска минимума функции сферы

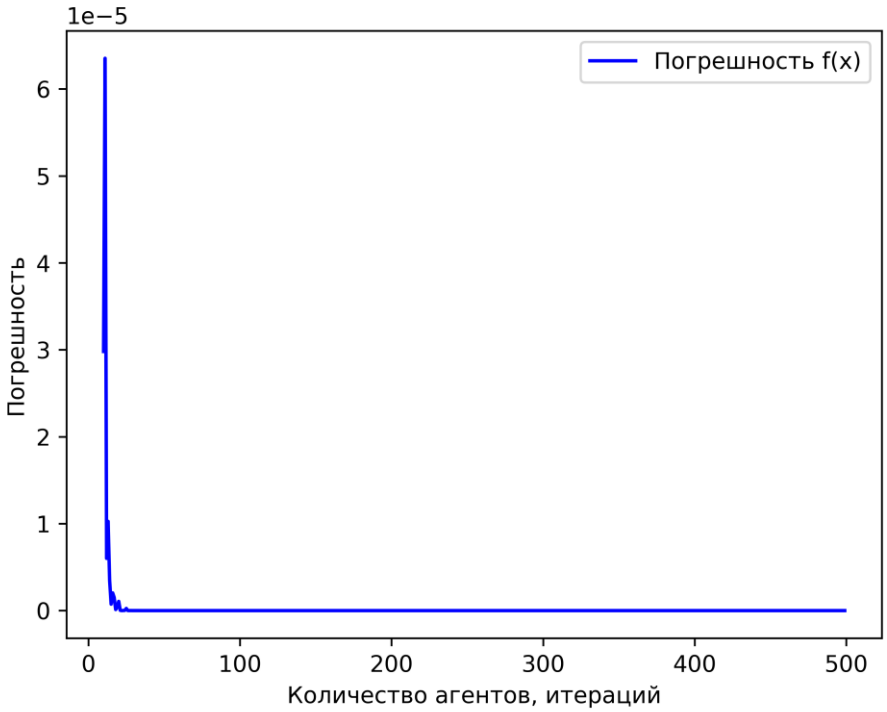

Рисунок 22 - Результат поиска значения функции сферы

Как видно из рисунков, функция сферы довольно проста в оптимизации. Она достигает оптимальных значений менее, чем за 100 итераций.

Вывод

В результате изучения алгоритма SAO были получены практические навыки по реализации метаэвристического алгоритма роевого интеллекта, вдоховленного процессами сублимации и таяния снега. К преимуществам алгоритма можно отнести его способность справляться с преждевременной сходимостью. К недостаткам можно отнести необходимость подбора параметров под каждую функцию. Для получения более точного значения минимума функции и значения функции необходимо задавать или менять не только количество агентов, но и количество итераций. Такой подход позволяет достичь более точных вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. Lingyun D., Sanyang L. Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design. *Expert Systems with Applications*, 2023, vol. 225.

2. Зорич В.А. К броуновскому движению // ИПМех РАН. – URL: https://istina.ipmnet.ru/collections/304331907/ (дата обращения: 07.01.2025 г.).

3. SAO Project // Github. – URL: https://github.com/annochka2710/SAO-problem-solution (дата обращения: 11.01.2025 г.).

4. Virtual Library of Simulation Experiments: Test Functions and Datasets // Simon Fraser University. – URL: https://www.sfu.ca/~ssurjano/optimization.html (дата обращения: 07.01.2025 г.).