Чугуй Владимир 9302

# Алгоритм оптимизации косяком рыб (Fish School Search, FSS)

## 1. Теоретическое обоснование алгоритма

Поиск косяком рыб (Fish School Search, FSS) – это подсемейство алгоритмов роевого интеллекта, относящихся к классу метаэвристических алгоритмов [1]. В FSS простые агенты называются рыбами, и каждая рыба имеет вес, который представляет "успех", достигнутый во время поиска. Значения и вариации весов влияют на индивидуальные и коллективные движения. Встроенные механизмы кормления и скоординированных действий заставляют косяк двигаться в направлении положительного градиента, чтобы набирать вес и находить лучшие места на местном и глобальном уровнях.

В алгоритме FSS упрощенно можно представить, что рыбы плавают в условном аквариуме, стенки которого являются границами области определения исследуемой функции. Рыб характеризует их вес как мера успешности в поиске пищи (решения) и играет роль памяти рыбы. Наличие веса у рыб является основной идеей алгоритма и отличием от роя частиц. Эта особенность алгоритма FSS позволяет отказаться от необходимости отыскивать и фиксировать глобально лучшие решения, как это происходит в рое частиц.

Операторы алгоритма FSS объединены в две группы:

- Оператор кормления, формализующий успешность исследования агентами тех или иных областей аквариума;
- Операторы плавания, реализующие алгоритмы миграции агентов

Оператор кормления представляет собой расчет веса рыбы. Основная идея состоит в том, чтобы заставить рыб "плыть" к положительному градиенту, чтобы "есть" и "набирать вес". В совокупности более тяжелые

рыбы оказывают большее влияние на процесс поиска в целом, что заставляет центр масс косяка рыб перемещаться в сторону лучших мест в пространстве поиска в течение итераций. Приращение веса на данной итерации пропорционально нормализованной разности значений фитнесс-функции.

Операторы плавания подразделяются по виду перемещений на три типа – индивидуальное, инстинктивно-коллективное и коллективно-волевое.

Индивидуальное плавание можно интерпретировать как локальный поиск в окрестностях текущего положения рыбы. Вектор движения каждой особи направлен случайно и имеет разную величину.

После индивидуального плавания происходит замер фитнес-функции. Если в результате индивидуального движения улучшение позиции рыбы не произошло, то считаем, что движения у этой конкретной рыбы не было и она остаётся на месте. То есть переместятся на новое положение только те рыбы, у которых произошло улучшение фитнес-функции.

После завершения индивидуального плавания выполняется оператор инстинктивно-коллективного перемещения. Инстинктивно-коллективное перемещение служит для корректировки общего положения косяка рыб с учетом изменения фитнес - функции каждой рыбы на предыдущей итерации. Инстинктивно-коллективное плавание формализует групповое синхронное перемещение косяка рыб на новое место, обеспечивая поиск новых мест корма, тогда как индивидуальное перемещение позволяет улучшать положение локально.

Коллективно-волевое плавание выполняется вслед за инстинктивно-коллективным плаванием. Коллективно-волевое плавание заключается в смещении всех агентов в направлении текущего центра массы популяции, если суммарный вес косяка в результате индивидуального и инстинктивно-коллективного плавания увеличился, и в противоположном направлении – если это вес уменьшился.

Ниже приведен псевдокод алгоритма:

**Алгоритм 1.** Алгоритм FSS.

```
Begin
    initialize_randomly_all_fish;
while (stop_criterion is not met)
        for (each_fish)
            individual_movement;
        evaluate_fitness_function;
        end for
        feeding_operator;
        for (each_fish)
        instinctive_movement;
        end for
    calculate_barycentre;
        for (each_fish)
         volitive_movement;
    evaluate_fitness_function;
        end for
update_individidual_step;
    end while
    end
```

Для решения задачи поиска минимума функции в алгоритме используется следующие модификации [2]:

**1. Инициализация аквариума**

После ввода параметров алгоритма происходит

заполнение аквариума рыбами, то есть каждому экземпляру присваиваются случайные координаты ($initPosition$), равномерно распределенные в пределах границ аквариума, и устанавливается вес, равный нулю.

**2. Критерий остановки**

В общем случае критерием остановки может быть достижение определенной точности результатов, время выполнения программы, количество используемой памяти и прочее. В текущей реализации алгоритма этим является количество итераций, которое задается до начала выполнения алгоритма (*iterationCount*).

**3. Индивидуальное плавание**

В этой стадии плавания рыбам дается шанс совершить индивидуальные перемещения, которые ограничиваются параметром «шаг индивидуального

плавания» (*individualStep*):

$$individualSwim = initPosition\ + rand(-1, 1) * individualStep, \quad (1)$$

Также на этом шаге высчитывать разница фитнесс-функции для начального положения и после индивидуального плавания, как указывалось ранее если агент не принял более выгодное положение или вышел за края диапазона, то считается что агент остался на месте, и разница фитнесс-функции обнуляется.

**4. Оператор кормления**

На этом шаге закрепляется успех в индивидуальной стадии плавания. Для этого используется характеристика «вес». Она равняется изменению функции приспособленности для данного агента до и после индивидуальной стадии, нормированного максимальным изменением функции среди популяции:

$$weight += \frac{\Delta f_i}{max\Delta f_i}, \quad (2)$$

Значение *weight* лежит в диапазоне [*weighInit*; *weightScale*] и при необходимости корректируется.

**5. Инстинктивно-коллективное плавание**

После этого рыбы совершают следующую стадию плавания — инстинктивно-коллективную. Для всего косяка рыб высчитывается величина «общий шаг миграции»:

$$migrateStep = \frac{\sum_{i=1}^{populstionSize}(individualSwim_i - initPosition_i) * \Delta f_i}{\sum_{i=1}^{populstionSize} \Delta f_i}, \quad (3)$$

С практической точки зрения это означает, что на каждого агента влияет вся популяция в целом, при этом влияние отдельного агента пропорционально его успехам в индивидуальной стадии плавания. После этого вся популяция смещается на подсчитанную величину *migrateStep*:

$$collInst = individualSwim + migrateStep\,, \quad (4)$$

**6. Центр тяжести косяка**

Перед следующей операцией плавания необходимо выполнить промежуточные действия, а именно: подсчитать центр тяжести всего косяка:

$$barycenter = \frac{\sum_{i=1}^{populstionSize}(collInst_i) * \mathrm{weight_i}}{\sum_{i=1}^{populstionSize} \mathrm{weight_i}}, \quad (5)$$

**7. Коллективно-волевое плавание**

На данном этапе плавания косяк рыб должен проявить интенсификационные и диверсификационные свойства (рисунок 1).

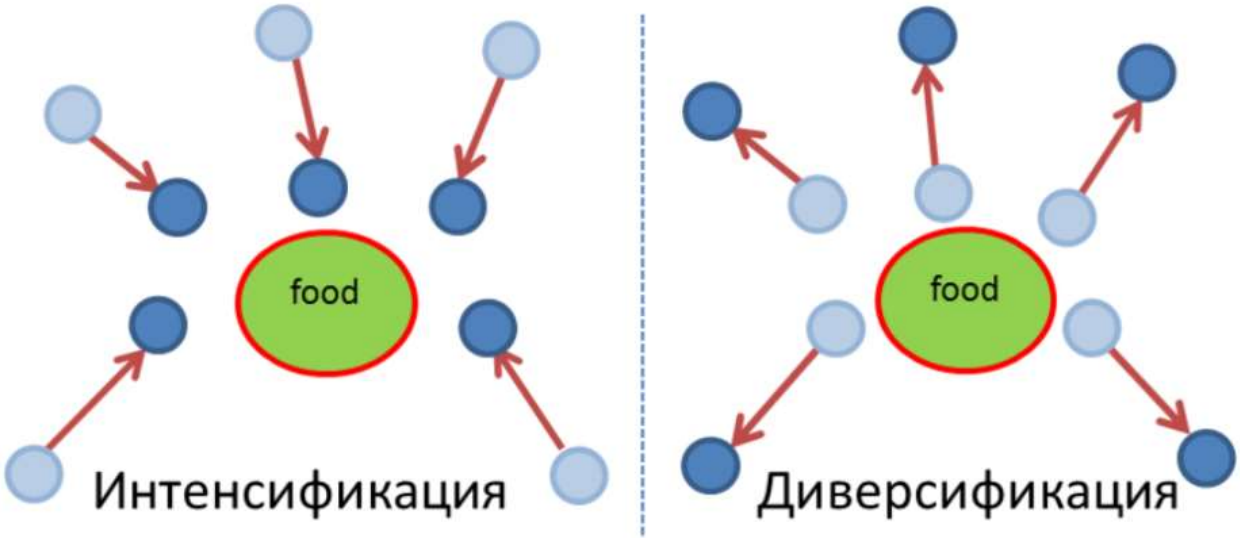


Рисунок 1 - Стадия коллективно-волевого плавания

Величина collStep в следующей формуле отвечает за шаг коллективного смещения. Рекомендуется использовать значение, в 2 раза большее индивидуального шага поиска. Оператор dist высчитывает расстояние между двумя точками в евклидовом пространстве:

$$collVolitiv = collInst \pm collStep * rand(0;\ 1) * \\ * \frac{collInst - barycenter}{\mathrm{d}ist(collInst, barycenter)}, \quad (6)$$

**8. Обновление позиций косяка**

Последним оператором в итерации является линейное уменьшение шага индивидуального поиска для следующей итерации. Также необходимо закрепить последние позиции агентов в начальные значения для продолжения итерации алгоритма, а также заново вычислить фитнесс-функции.

$$individStep -= \frac{InitIndividStep}{IterationCount}, \quad (7)$$

С практической точки зрения это объясняется тем, что чем больше итераций уже пройдено, тем ближе агенты находятся к минимуму функции, и, следовательно, имеет смысл ограничить область поиска на следующей итерации.

## 2. Исследование алгоритма

Для исследования эффективности работы алгоритма была выбрана задача глобальной оптимизации функций Химмельблау, Розенброка, Изома, "крест на подносе", Бута и сферы [3].

В результате того, что критерием остановки было выбрано количество итераций, в связи с этим количество вычислений функции равно:

$$neval = (iterationCount * 2 + 1) * populationSize, \quad (8)$$

График нахождения глобального минимума функции Химмельблау на диапазоне [-4, 4] представлен на рисунке 2.

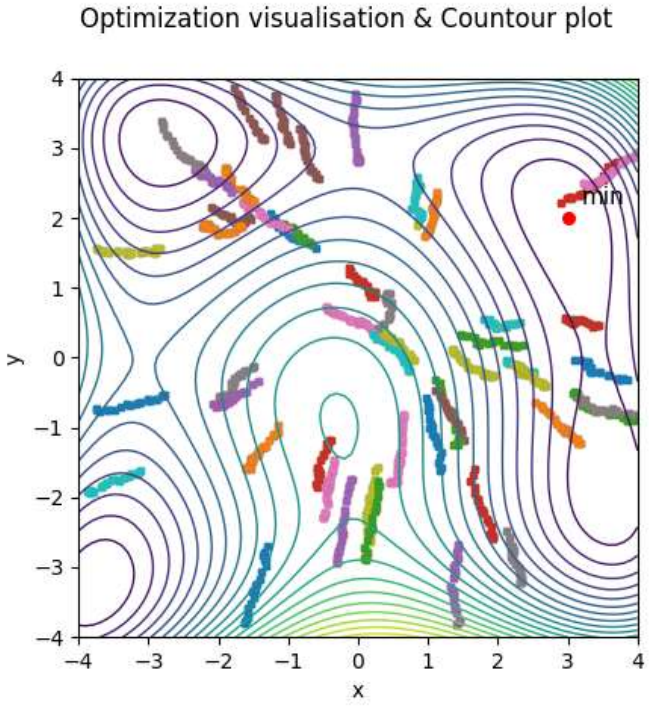


Рисунок 2 – Работа алгоритма на функции Химмильблау

В результате работы алгоритма получили минимум в точке *x_min* = [2.99999363, 2.00001501] и *F(x_min)* = 3.4180287285792345e-09.

У функции Химмильблау есть несколько локальных минимумов: значение 0 в точках (3.0, 2.0), (-2.805118, 3.131312), (-3.779310, -3.283186), (3.584428, -1.848126). В результате работы алгоритма был найден минимум в точке (3.0, 2.0).

График нахождения глобального минимума функции Розенброка на диапазоне [-4, 4] представлен на рисунке 3.

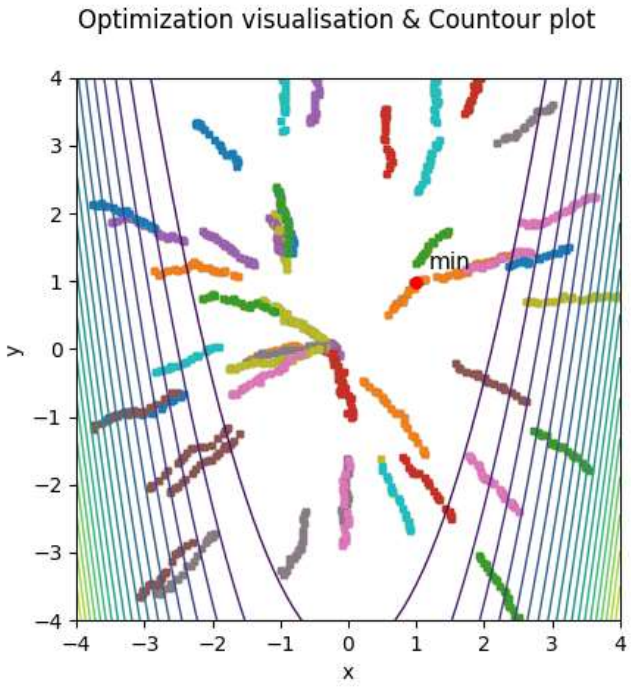


Рисунок 3 – Работа алгоритма на функции Розенброка

В результате работы алгоритма получили минимум в точке *x_min* = [0.9889603, 0.97799464] и *F(x_min)* = 0.00012210383140080444.

Глобальный минимум функции Розенброка находится в точке (1, 1) со значением 0.

В ходе работы алгоритма один агент точно приблизился к точке минимума, для выхода в более конкретное значение нуля надо уменьшать диапазон или увеличивать число итераций/агентов.

График нахождения глобального минимума функции Изома на диапазоне [-10, 10] представлен на рисунке 4.

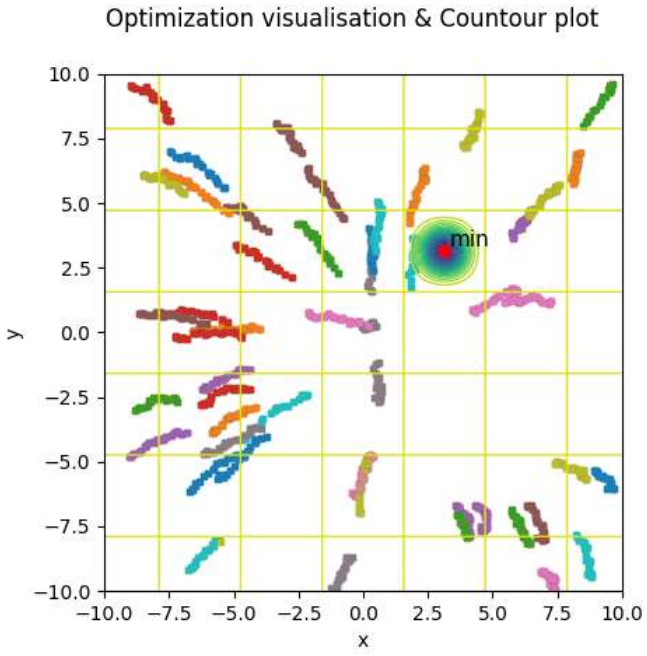


Рисунок 4 - Работа алгоритма на функции Изома

Глобальный минимум функции находится в точке (π, π) со значением f(x) = -1 (рисунок 5).

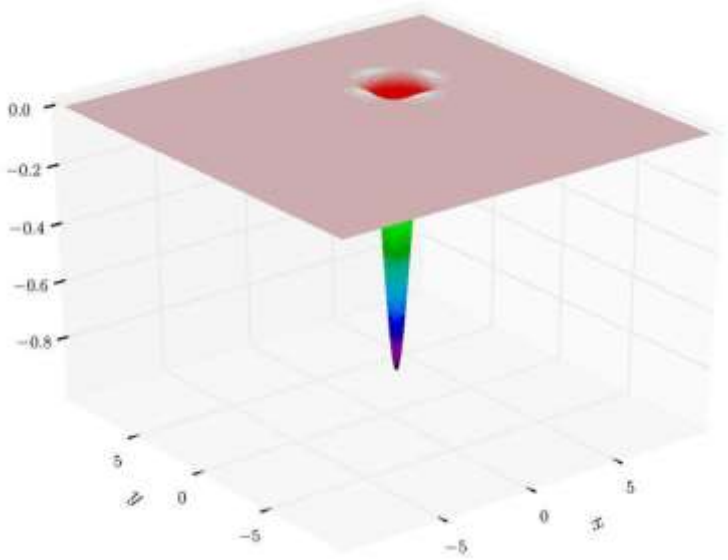


Рисунок 5 – Функция Изома

Алгоритм нашел точку минимума в x_min = [3.14159617, 3.141587] и f(x_min) = -0.9999999999334752.

График нахождения глобального минимума функции "крест на подносе" (Cross-In-Tray), на диапазоне [-10, 10] представлен на рисунке 6.

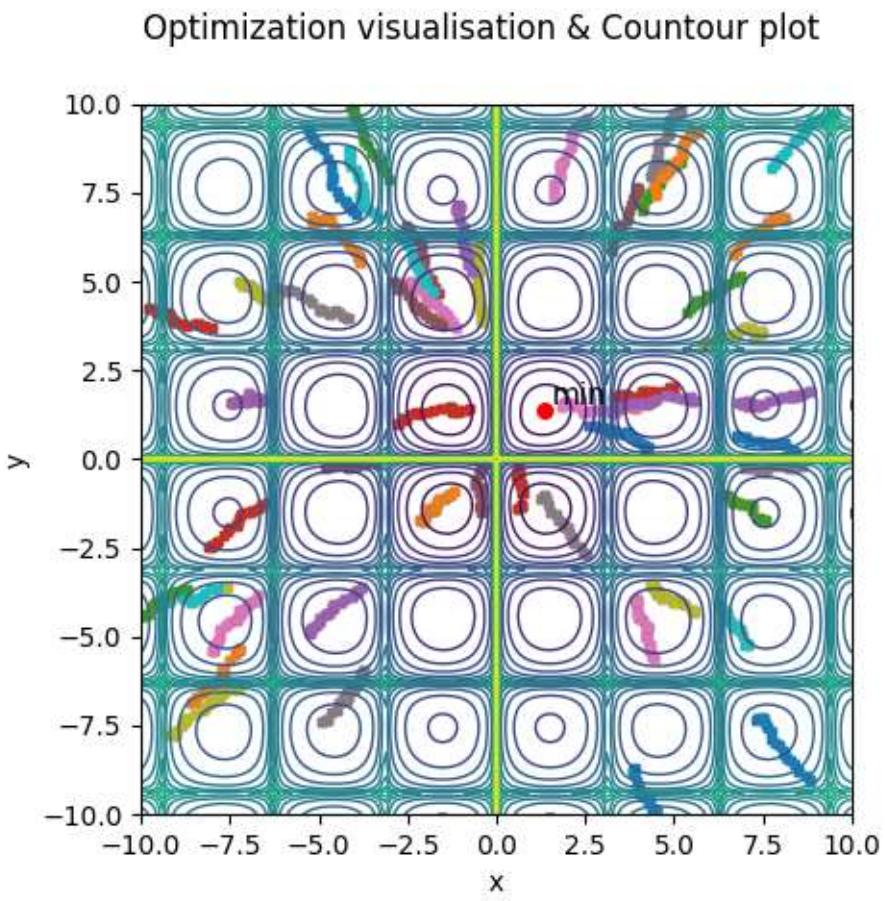


Рисунок 6 – Работа алгоритма на функции «Крест на подносе»

Алгоритм нашел точку минимума в x_min = [1.3494137, 1.34941557] и f(x_min) = -2.0626118708085803.

У функции "Крест на подносе" имеются следующие значения минимумов (рисунок 7):

$$\text{Min} = \begin{cases} f(1.34941, -1.34941) & = -2.06261 \\ f(1.34941, 1.34941) & = -2.06261 \\ f(-1.34941, 1.34941) & = -2.06261 \\ f(-1.34941, -1.34941) & = -2.06261 \end{cases}$$

Рисунок 7 – Значения минимума функции «Крест на подносе»

Алгоритм успешно нашел минимум в точке (1.349, 1.349).

График нахождения глобального минимума модифицированной функции Бута, на диапазоне [-10, 10] представлен на рисунке 8.

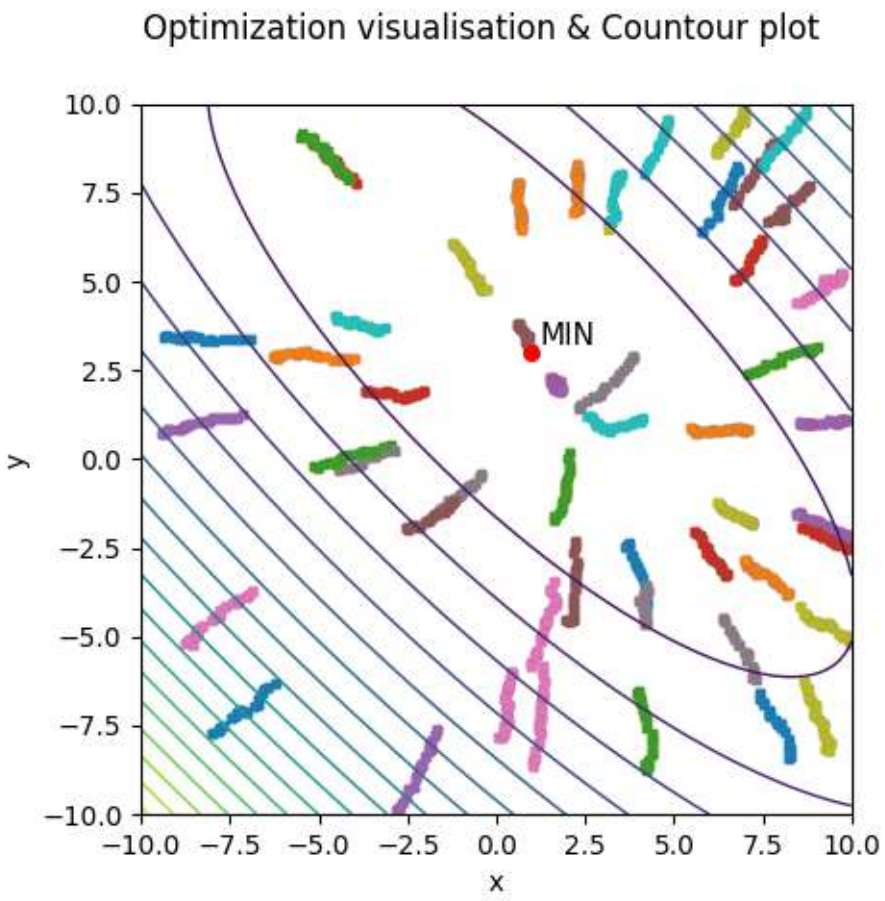


Рисунок 8 – Работа алгоритма на модифицированной функции Бута

Алгоритм нашел точку минимума в x_min = [1.00002314e+00 2.99998570e+00 1.05976183e-04] и f(x_min) = 1.2283706957558716e-08.

Функция Бута имеет локальный минимум в f(1, 3) = 0. Алгоритм точно нашел точку минимума.

График нахождения глобального минимума сферической функции, на диапазоне [-10, 10] представлен на рисунке 9.

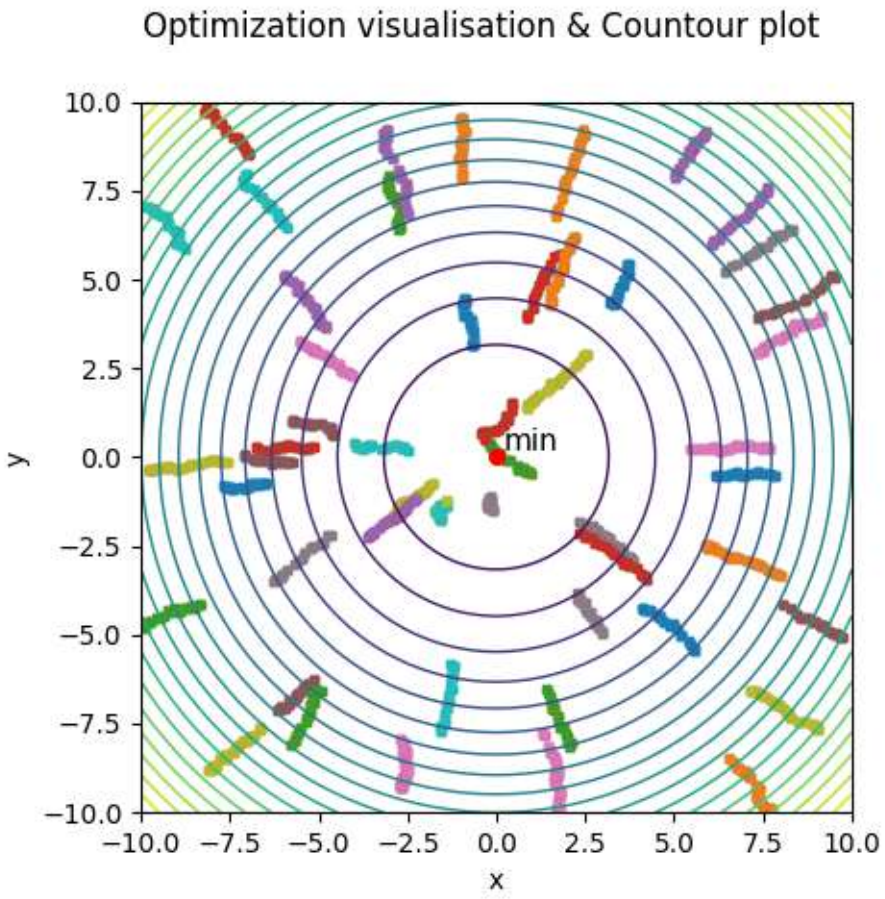


Рисунок 9 – Работа алгоритма на сферической функции

У сферической функции есть только один глобальный минимум – в начальной точке, то есть, когда все значения равны нулю. Алгоритм нашел минимум функции в x_min = [1.20548012e-06, 1.44814247e-05, 2.57766731e-06] и f(x_min) = 2.178092122545681e-10. Данные значения соответствуют точке минимума в x = (0, 0, 0) и f(x) = 0. Алгоритм корректно нашел минимумы для всех функций.

# ВЫВОД

В данной работе был успешно реализован алгоритм оптимизации поиска косяком рыб. Его модификация и особенности реализации подробно описаны в работе, а работоспособность проверена на функциях Химмельблау, Розенброка, Изома, "крест на подносе”, Бута и сферы. Алгоритм успешно находил точки минимума и справился с поставленной задачей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карпенко А.П. Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов // Информационные технологии. – 2012. – №7. – 32 с.

2. Куликов А.Н. Программна оптимизации, инспирированная поведением косяка рыб // Инноватика. Научный электронный журнал. 2014 № 1

3. Тестовые функции для оптимизации // Википедия [Интернет-ресурс] URL: https://inlnk.ru/Pmp66E