УДК 004

Решение задачи алгоритмом косяка рыб

Потылицын Андрей Олегович
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В данной статье описывается метод решения задачи на нахождение оптимальной скорости грузовика, используя алгоритм поиска косяком рыб с помощью языка программирования C++.
Ключевые слова: C++, максимум функции, алгоритм поиска косяком рыб.

Solving the problem by the fish school algorithm

Potylitsyn Andrey Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the method of solving the problem of finding the maximum speed truck, using the search algorithm of a school of fish using the C++ programming language.
Keywords: C++, algorithm search for a school of fish, maximum of function.

Имеется большое количество способов к решению задач на оптимизацию. Симплекс, генетический и множество других, а так же имеются методы, основанные на популяционных алгоритмах. Один из них – алгоритм косяка рыб. Таких алгоритмов так же много, как и методов, они все моделируют решение задач, путем движения колон, которые находят лучшее значение.

Цель исследования заключается в решении задачи, где будет найден максимум функции. Необходимо используя алгоритм косяком рыб и язык программирования C++, выявить эффективность данного алгоритма. Описание задачи: «Найти оптимальную скорость Грузовика, который зависит от количества грузчиков и времени погрузки. Необходимо найти оптимально соотношение между количеством грузчиков и временем погрузки 1 ящика в грузовик. При этом дана математическая функция выражающая скорость (рис.1)

$$z = 3x^2 + xy + 2y^2 - x - 4y$$

Рисунок 1 – Функция задачи

Z – скорость грузовика в км/ч , У – Количество грузчиков в ед. , Х – Время погрузки 1 груза.»

А.Н.Куликова в своей статье подробно говорит об алгоритме и доказывает эффективность алгоритма косяка рыб в действии [1]. В.А.Частиковым и др. был разработан метод алгоритма и адаптировали метод для оптимального движения квадрокоптера, который имел возможность самостоятельного управления, без помощи пилота [2-3]. М.М.Егин рассматривает принцип работы алгоритма косяка рыб в многомерной глобальной оптимизации [4]. В статье Дружининой М.А. проведены исследования, показавшие особенности работы алгоритма движения косяка рыб, сформированы рекомендации по выбору значений, оптимальных параметров алгоритма движения косяка рыб, исходя из тонкостей решаемой задачи оптимизации [5]. В статье В.С. Васильева, генетические алгоритмы в сравнении с оптимизационными методиками, рассматриваются основные понятия генетических алгоритмов, применение генетических алгоритмов к анализу и прогнозированию социально-экономических систем [6].

Была поставлена задача, где необходима было найти оптимальную скорость движения Грузовика. Решение будет представлено как вручную, с помощью онлайн решения, а так же с помощью языка программирования C++ [7].

Для начала рассмотри решение задачи вручную (Рис.2). В этом варианте не будет применен алгоритм косяка рыб.

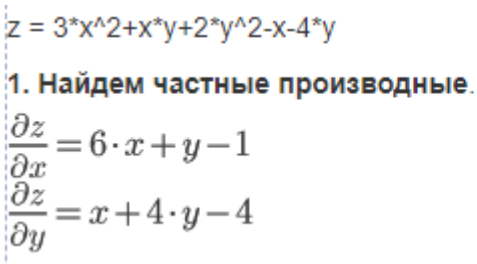

Рисунок 2 – Нахождение частных производных

После нахождения частных производных необходимо решить саму систему (Рис.3).

2. Решим систему уравнений.

6*x+y-1 = 0

x+4*y-4 = 0

Получим:

а) Из первого уравнения выражаем *x* и подставляем во второе уравнение:

x = -4*y+4

-23*y+23 = 0

Откуда y = 6

Данные значения *y* подставляем в выражение для *x*. Получаем: x = -2

Количество критических точек равно 1.

$M_1(-2;6)$

Рисунок 3 – Решение системы

Следующим этапом будет нахождение частных производных второго порядка (Рис.4) и вычисление их в критических точках (Рис.5).

3. Найдем частные производные второго порядка.

$$\frac{\partial^2 z}{\partial x \partial y} = 1$$
$$\frac{\partial^2 z}{\partial x^2} = 6$$
$$\frac{\partial^2 z}{\partial y^2} = 4$$

Рисунок 4 – Нахождение частных производных второго порядка

4. Вычислим значение этих частных производных второго порядка в критических точках $M(x_0;y_0)$.

Вычисляем значения для точки $M_1(6;2)$

$$A = \frac{\partial^2 z}{\partial x^2}_{(0;1)} = 1$$
$$C = \frac{\partial^2 z}{\partial y^2}_{(0;1)} = 10$$
$$B = \frac{\partial^2 z}{\partial x \partial y}_{(0;1)} = 11$$

$AC - B^2 = 23 > 0$ и $A > 0$, то в точке $M_1(6;2)$ имеется минимум $z(8;4) = 55$

Вывод: В точке $M_1(6;2)$ имеется минимум $z(8;4) = 55$;

Рисунок 5 – Вычисление и вывод

При решении задачи ручным способом был получен ответ 55, следовательно , оптимальная скорость будет 55км/ч. Теперь представим это же решение, но уже с помощью метода анализа косяка рыб на языка C++(рис.6).

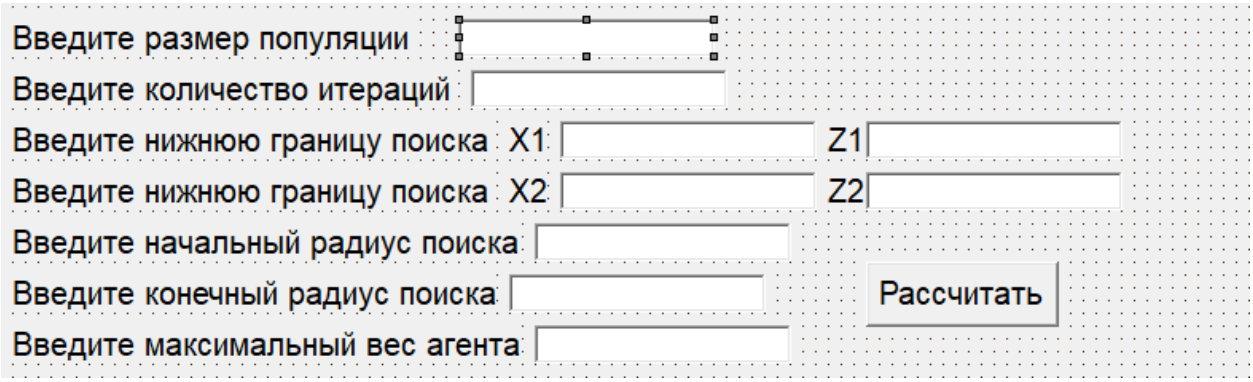

Рисунок 6 – Окно ввода

Далее необходимо представить массивы (рис.7).

```
array_fill(array, raz_pop, (max_vec / 2));
array_fill(array2, raz_pop, 0);
array_fill(array3, raz_pop, 0);
array_fill(pos2, raz_pop, 0);
array_fill(pos3, raz_pop, 0);
array_fill(pos4, raz_pop, 0);
array_fill(ran1, raz_pop, 0);
array_fill(ran2, raz_pop, 0);
array_fill(ran3, raz_pop, 0);
```

Рисунок 7 – Объявленные массивы

Необходимо так же выбрать положение агента, оно будет случайным (рис.8).

```
for (int n = 0; n < raz_pop; n++) {
        pos2[n] =  verx + rand() % verxl;
        pos3[n] = niz + rand() % nizl;
        pos4[n] = niz + rand() % nizl;
        ran1[n] = pos2[n];
        ran2[n] = pos3[n];
        ran3[n] = pos4[n];
```

Рисунок 8 – Выбор случайного положения

Следующим этапом будет цикл. Он будет вычислять значение функции агента. Так же будет просчет значения со сдвигом, сравнение значений и при условии, что новое – лучше, то оставляем и наоборот (рис.9).

```
for (int j = 0; j < raz_pop; j++) {
        if (ran1[j] < 0) ran1[j] = 0;
        array2[j] = 3 * pow(ran1[j], 2) + (ran2[j] * ran1[j]) + 2 * pow(ran2[j], 2) - ran1[j] - 4 * ran2[j];
        pos2[j] = pos2[j] + random(-1, 1) * kk;
        pos3[j] = pos3[j] + random(-1, 1) * kk;
        if (pos2[j] < 0) pos2[j] = 0;
        array2[j] = 3 * pow(ran1[j], 2) + (ran2[j] * ran1[j]) + 2 * pow(ran2[j], 2) - ran1[j] - 4 * ran2[j];
```

Рисунок 9 – Нахождение значения

Следующим пунктом вычислим «Вес», после чего будет высчитываться величина «Общий шаг миграции», добавим смещение косяка рыб. (рис. 10)

```
            array3[j] = array2[j] - array1[j];
            m += (pos2[j] - ran1[j]) * array1[j];
            m3 += (pos3[j] - ran2[j]) * array3[j];
            m1 += array3[j];
            if (m1 == 0) m1 = 1;
            if (m == 0) m1 = 1;
            if (m3 == 0) m3 = 1;
            if (array3[j] == 0) array3[j] = 1;
    }

    m2 = m / m1;
    m4 = m3 / m1;

    for (int l = 0; l < raz_pop; l++) {
            if ((array3[l] > max) && (array3[l] != 0)) max = array3[l];
    }

    float t = array_sum(array);

    for (int k = 0; k < raz_pop; k++) {
            array[k] = (array3[k] / max);
            if (array[k] > max_vec) array[k] = max_vec;
            if (array[k] < 0) array[k] = 1;
    }
```

Рисунок 10 – Вес рыбы и смещение

А теперь финальная стадия, осталось просчитать последнее нахождение агентов после итерации, учтя вес, сдвиг. Если общий вес растет, то делается все верно и круг можно сузить и наоборот, а так же сделает подсчет коэффициентов и найдем максимум (Рис.11).

```
for (int f = 0; f < raz_pop; f++) {
        pos2[f] = pos2[f] + m2;
        pos3[f] = pos3[f] + m4;
        bary4 += pos3[f] * array[f];
        bary += pos2[f] * array[f];
        bary1 += array[f];
}

bary3 = bary / bary1;
bary2 = bary4 / bary1;

for (n = 0; n < raz_pop; n++) {
        if ((pos2[n] - bary3) == 0) bary3 = bary3 - 1;
        if ((pos3[n] - bary2) == 0) bary2 = bary2 - 1;

        if (array_sum(array) > t) {
                pos2[n] = pos2[n] + ((kk * 2) *  random(0, 1)  * (pos2[n] - bary3) / (sqrt(pow(pos2[n] - bary3, 2))));
                pos3[n] = pos3[n] + ((kk * 2) *  random(0, 1)  * (pos3[n] - bary2) / (sqrt(pow(pos3[n] - bary2, 2))));
        } else {
                pos2[n] = pos2[n] - ((kk * 2) *  random(0, 1)  * (pos2[n] - bary3) / (sqrt(pow(pos2[n] - bary3, 2))));
                pos3[n] = pos3[n] - ((kk * 2) *  random(0, 1)  * (pos3[n] - bary2) / (sqrt(pow(pos3[n] - bary2, 2))));
        }

        if (pos2[n] > verx1) pos2[n] = verx1;
        if (pos2[n] < verx) pos2[n] = verx;
        if (pos3[n] > niz1) pos3[n] = niz1;
        if (pos3[n] < niz) pos3[n] = niz;
```

Рисунок 11 – Смещение и нахождение максимума

Теперь необходимо все проверить и сравнить полученный алгоритм решения из онлайн сервиса и написанный самим, если ответ будет верен, то следовательно написанный код верен и имеет возможность решения методом алгоритма косяка рыб (рис.12).

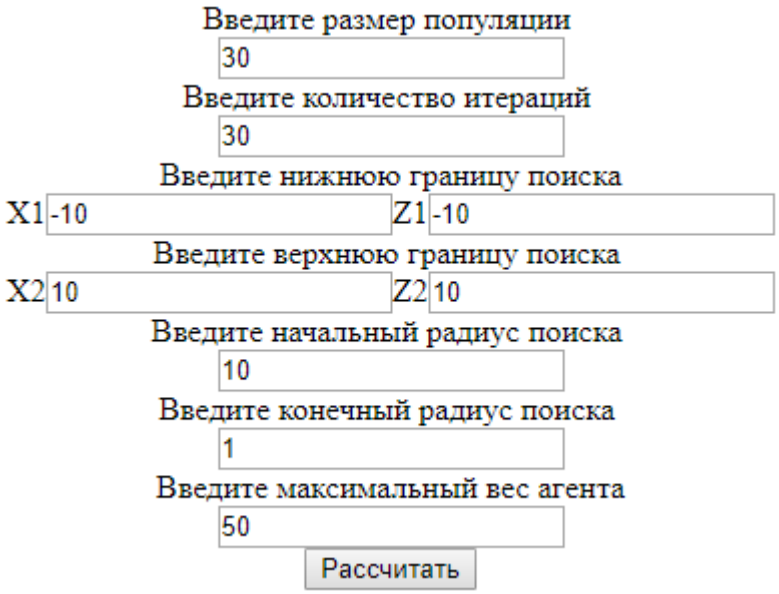

Рисунок 12 – Введенные параметры

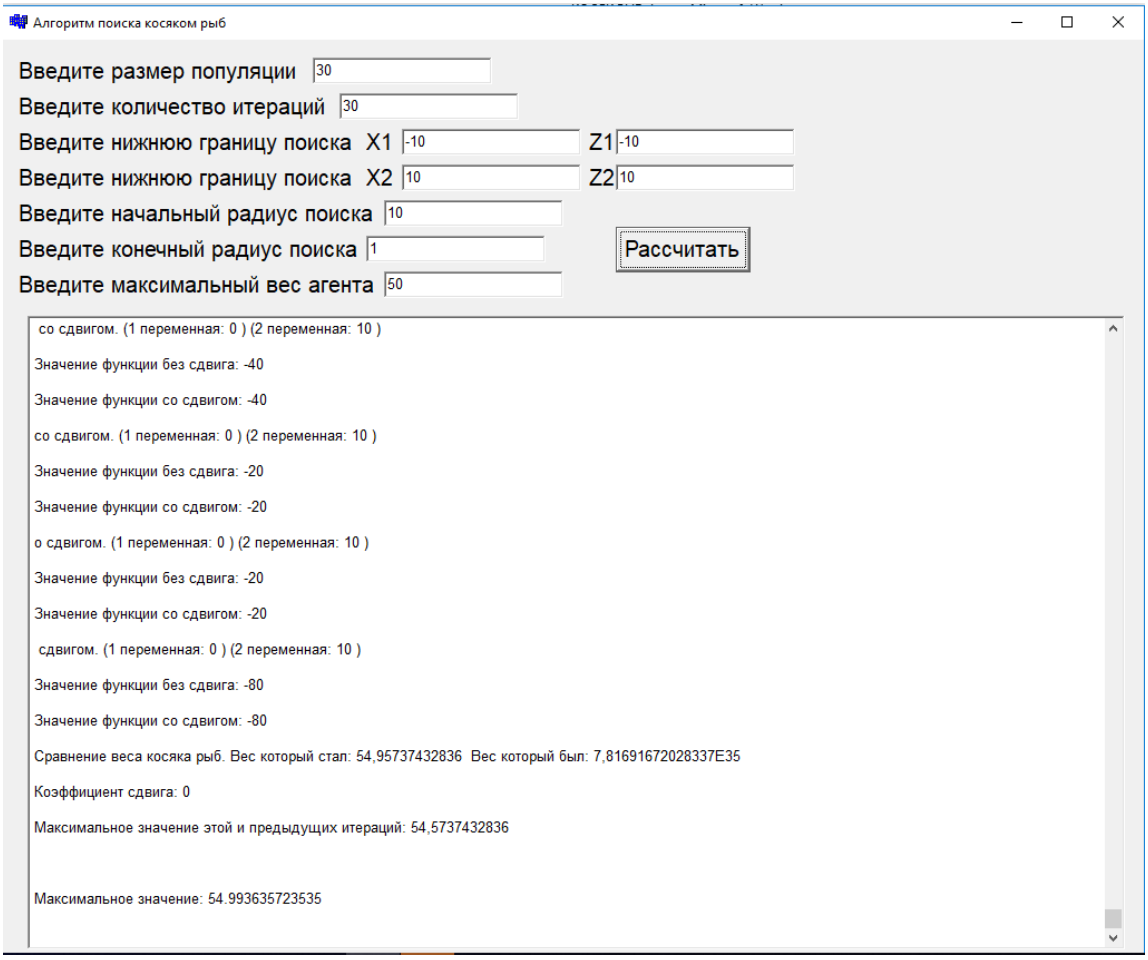

Рисунок 13 – Полученный результат

Как видно на рисунке (рис.13) результаты совпадают практически полностью.

Следует отметить, что при решении задачи через язык программирования имеется недочет, т.к. ответ получается приближенным к точному. Но зато имеется быстрая программа считывающая и выдающая ответ в разы быстрее, чем решение вручную.

В данной статье была решена задача с функцией у которой 2 неизвестных, а так же была написана программа на языке программирования С++ и проверена этой же задачей.

Библиографический список

1. Куликов А.Н. Программа оптимизации, инспирированная поведением косяка рыб // Инноватика. 2014. №1. С. 33-42.
2. Частикова В.А., Дружинина М.А., Кекало А.С. Адаптация алгоритма поиска косяком рыб для оптимизации движения квадрокоптеров // Научные труды Кубанского государственного технологического университета. 2014. №S6. С. 97-100.
3. Частикова В.А., Дружинина М.А., Кекало А.С. Исследование эффективности алгоритма поиска косяком рыб в задаче глобальной оптимизации // Современные проблемы науки и образования. 2014. №4 URL: https://www.science-education.ru/ru/article/view?id=14142.
4. Васильев В.С. Генетические алгоритмы в сравнении с оптимизационными методиками // Студенческая наука для развития информационного общества, сборник материалов 6 всероссийской научно-технической конференции. Ставрополь: Северо-Кавказский федеральный университет, 2017. С. 330-331.
5. Егин М.М. Алгоритм косяка рыб в многомерной оптимизации // Прикладные исследования и технологии ART 2016. Москва: Негосударственное образовательное учреждение высшего образования Московский технологический институт, 2016. С. 96-99
6. Дружинина М.А. Особенности работы алгоритма движения косяка рыб // Современное состояние и перспективы развития технических наук. Уфа: Башкирский государственный университет, 2014. С. 23-25.
7. Поиск экстремумов функции URL: https://math.semestr.ru/math/extremum.php (дата обращения: 14.11.19).