**по лабораторной работе № 7**

**Дисциплина: Оптимизация вычислительных процессов**

# Тема: Алгоритм бактериальной оптимизации

Выполнил:

Студент 46/2 группы

Ролдугин А.С.

Преподаватель:

Климец А.А.

**Цель работы:** разработать алгоритм бактериальной оптимизации функций.

**1. Алгоритм бактериальной оптимизации функций.**

Алгоритм бактериальной оптимизации (Bacterial Foraging Оptimization, BFO) основан на использовании трёх следующих основных механизмов: хемотаксис, репродукция, ликвидация и рассеивание.

Программную реализацию метода имитации поведения бактерий начинаем с инициализации начальных данных:

- определяем размер популяции;

- задаем число бактерий в популяции;

- максимальное число шагов хемотаксиса;

- максимальное число шагов репродукции;

- максимальное число шагов ликвидации и распределения;

- вероятность того, что бактерия будет уничтожена или ее положение будет изменено.

На втором шаге генерируем начальные позиций бактерий.

На каждом этапе сохраняем текущее значение положения бактерий для последующих сравнений.

Вносим элемент случайности с помощью вектора единичной длины.

После чего делаем шаг вдоль случайного направления, определяемого вектором. После каждой итерации считаем значение целевой функции для новой позиции и добавляем «роевой» эффект притяжения.

Процедура хемотаксиса реализует в алгоритме BFO локальную оптимизацию. Значение шага хемотаксиса может меняться в процессе поиска, уменьшаясь по некоторому закону с ростом числа итераций.

Механизм репродукции позволяет ускорить сходимость алгоритма .

В результате выполнения процедуры репродукции результирующее число бактерий в популяции остаётся низменным.

Процедуры хемотаксиса и репродукции недостаточны для отыскания глобального максимума целевой функции, поскольку эти процедуры не позволяют бактериям покидать найденные ими локальные максимумы этой функции. С помощью Процедуры ликвидации и рассеивания решаем проблему.

Механизм ликвидации и рассеивания включается после выполнения определённого числа процедур репродукции.

С заданной вероятностью случайным образом выбираем бактерии и уничтожаем их. Вместо каждой из уничтоженных бактерий в случайно выбранной точке пространства поиска создаём новую бактерию с тем же номером. В результате выполнения операции ликвидации и рассевания число бактерий в колонии также остаётся постоянным.

Процедура хемотаксиса реализует в алгоритме BFO локальную оптимизацию. Значение шага хемотаксиса может меняться в процессе поиска, уменьшаясь по некоторому закону с ростом числа итераций.

Механизм репродукции имеет своей целью ускорение сходимости алгоритма (за счёт сужения области поиска). В результате выполнения процедуры репродукции результирующее число бактерий в популяции остаётся низменным.

Процедуры хемотаксиса и репродукции недостаточны для отыскания глобального максимума целевой функции, поскольку эти процедуры не позволяют бактериям покидать найденные ими локальные максимумы этой функции. Процедура ликвидации и рассеивания призвана преодолеть этот недостаток. Механизм ликвидации и рассеивания включается после выполнения определённого числа процедур репродукции и состоит в следующем. С заданной вероятностью случайным образом выбираем некое количество бактерий и уничтожаем их. Вместо каждой из уничтоженных бактерий в случайно выбранной точке пространства поиска создаём новую бактерию с тем же номером. В результате выполнения операции ликвидации и рассевания число бактерий в колонии также остаётся постоянным и равным.

**2. Программная реализация алгоритма BFO:**

**def** Rosenbrock**(**x**,** y**):**

**return** **(**1 **-** x**)** **\*\*** 2 **+** 100 **\*** **(**y **-** x **\*\*** 2**)** **\*\*** 2

**def** Rastrigin**(**x**,** y**):**

**return** 20 **+** x **\*\*** 2 **+** y **\*\*** 2 **-** 10 **\*** **(**cos**(**2 **\*** pi **\*** x**)** **+** cos**(**2 **\*** pi **\*** y**))**

**def** Himmelblau**(**x**,** y**):**

**return** **(**x**\*\***2**+**y**-**11**)\*\***2 **+** **(**x**+**y**\*\***2**-**7**)\*\***2

**def** draw\_Rosenbrock**():**

**return** **(**1 **-** X**)** **\*\*** 2 **+** 100 **\*** **(**Y **-** X **\*\*** 2**)** **\*\*** 2

**def** draw\_Rastrigin**():**

**return** 20 **+** X **\*\*** 2 **+** Y **\*\*** 2 **-** 10 **\*** **(**np**.**cos**(**2 **\*** pi **\*** X**)** **+** np**.**cos**(**2 **\*** pi **\*** Y**))**

**def** draw\_Himmelblau**():**

**return** **(**X**\*\***2**+**Y**-**11**)\*\***2 **+** **(**X**+**Y**\*\***2**-**7**)\*\***2

**def** select\_func**(**name\_func**)** **:**

**if** name\_func **==** "Функция Розенброка" **:**

f **=** Rosenbrock

Z **=** draw\_Rosenbrock

**elif** name\_func **==** "Функция Растригина" **:**

f **=** Rastrigin

Z **=** draw\_Rastrigin

**else** **:**

f **=** Himmelblau

Z **=** draw\_Himmelblau

**return** f**,** Z

**class** **Bacterium:**

**def** \_\_init\_\_**(**self**):**

self**.**x **=** 0

self**.**y **=** 0

self**.**dir **=** **[]**

self**.**norm **=** 0

self**.**fitness **=** 0

self**.**health **=** 0

self**.**f **=** select\_func**(**selected\_func**.**get**())[**0**]**

**def** diffusion**(**self**)://перемещение**

self**.**x **=** uniform**(-**5**,** 5**)**

self**.**y **=** uniform**(-**5**,** 5**)**

**def** jump**(**self**)://кувырок**

self**.**dir **=** **[**uniform**(-**0.5**,** 0.5**),** uniform**(-**0.5**,** 0.5**)]**

self**.**norm **=** sqrt**(**self**.**dir**[**0**]** **\*\*** 2 **+** self**.**dir**[**1**]** **\*\*** 2**)**

**def** swim**(**self**)://плавает**

**while** self**.**can\_swim**():**

self**.**x **+=** float**(**step**.**get**())** **\*** self**.**dir**[**0**]** **/** self**.**norm

self**.**y **+=** float**(**step**.**get**())** **\*** self**.**dir**[**1**]** **/** self**.**norm

**def** can\_swim**(**self**)://может плпавать**

x **=** self**.**x **+** float**(**step**.**get**())** **\*** self**.**dir**[**0**]** **/** self**.**norm

y **=** self**.**y **+** float**(**step**.**get**())** **\*** self**.**dir**[**1**]** **/** self**.**norm

**return** self**.**fitness **>** self**.**f**(**x**,** y**)**

**def** calc\_fitness**(**self**):**

self**.**fitness **=** self**.**f**(**self**.**x**,** self**.**y**)**

self**.**health **+=** self**.**fitness

**def** clone**(**self**):**

newBacterium **=** Bacterium**()**

newBacterium**.**x **=** self**.**x

newBacterium**.**y **=** self**.**y

newBacterium**.**dir **=** self**.**dir

newBacterium**.**norm **=** self**.**norm

newBacterium**.**fitness **=** self**.**fitness

newBacterium**.**health **=** self**.**health

**return** newBacterium

**def** rotate**():**

plt**.**show**()**

**def** calc**(**populationSize**,** numberToDiffuse**,** maxChemotaxis**,** maxReproduction**,** maxDiffusion**,** diffusionChance**,** step**)** **:**

f**,** Z **=** select\_func**(**selected\_func**.**get**())**

**if** populationSize **%** 2 **!=** 0 **:**

showerror**(**title**=**"Error"**,** message**=**"Введите размер популяции ТОЛЬКО четное число!"**)**

**return**

result **=** **[**0**,** 0**,** 0**]**

colony **=** **[]**

chemotaxis **=** 0

rep **=** 0

diff **=** 0

**for** i **in** range**(**populationSize**):**

bacterium **=** Bacterium**()**

bacterium**.**diffusion**()**

bacterium**.**calc\_fitness**()**

colony**.**append**(**bacterium**)**

**while** chemotaxis **<** maxChemotaxis**:**

chemotaxis **+=** 1

step **=** 1 **/** **(**chemotaxis **/** 10**)**

**for** b **in** colony**:**

b**.**jump**()**

**for** b **in** colony**:**

b**.**swim**()**

b**.**calc\_fitness**()**

**if** rep **<** maxReproduction**:**

part **=** int**(**len**(**colony**)** **/** 2**)**

colony **=** sorted**(**colony**,** key**=lambda** item**:** item**.**fitness**)[:**part**]**

**for** i **in** range**(**part**):**

colony**.**append**(**colony**[**i**].**clone**())**

rep **+=** 1

**if** diff **<** maxDiffusion**:**

current **=** 0

**while** current **<** numberToDiffuse**:**

i **=** randint**(**1**,** populationSize **-** 1**)**

chance **=** uniform**(**0**,** 1**)**

**if** chance **<** diffusionChance**:**

colony**[**i**].**diffusion**()**

current **+=** 1

diff **+=** 1

**for** b **in** colony**:**

b**.**calc\_fitness**()**

colony **=** sorted**(**colony**,** key**=lambda** item**:** item**.**fitness**)**

best **=** colony**[**0**]**

result **=** **[**best**.**x**,** best**.**y**,** best**.**fitness**]**

**print(**"Result:"**,** best**.**x**,** best**.**y**,** f**(**best**.**x**,** best**.**y**))**

draw**(**best**)**

**def** draw**(**best**)** **:**

f**,** Z **=** select\_func**(**selected\_func**.**get**())**

ax**.**scatter**(**best**.**x**,** best**.**y**,** f**(**best**.**x**,** best**.**y**),** color**=**'red'**,** s**=**50**,** marker**=**'o'**)**

lbl **=** "Найденный минимум\nx: " **+** str**(**best**.**x**)** **+** "\ny: " **+** str**(**best**.**y**)** **+** "\nz: " **+** str**(**f**(**best**.**x**,** best**.**y**))**

tk**.**Label**(**root**,**text**=**lbl**).**grid**(**row**=**16**,** column**=**20**,** columnspan **=** 2**)**

**def** draw\_graphic**(**name\_func**)** **:**

f**,** Z **=** select\_func**(**name\_func**)**

plt**.**cla**()**

ax**.**plot\_surface**(**X**,** Y**,** Z**(),** color **=** "green"**)**

canvas**.**draw**()**

root **=** tk**.**Tk**()**

fig **=** plt**.**figure**(**dpi**=**100**)**

ax **=** fig**.**add\_subplot**(**111**,** projection**=**'3d'**)**

canvas **=** FigureCanvasTkAgg**(**fig**,** master**=**root**)**

canvas**.**mpl\_connect**(**

"key\_press\_event"**,** **lambda** event**:** **print(**f"you pressed {event.key}"**))**

canvas**.**mpl\_connect**(**"key\_press\_event"**,** key\_press\_handler**)**

canvas**.**get\_tk\_widget**().**grid**(**row **=** 0**,** column **=** 0**,**rowspan**=**20**,** columnspan **=** 20**)**

toolbar **=** NavigationToolbar2Tk**(**canvas**,** root**,** pack\_toolbar**=False)**

toolbar**.**update**()**

toolbar**.**grid**(**row**=**21**,** column**=**0**,** columnspan **=** 20**,** rowspan **=** 1**)**

tk**.**Label**(**root**,** text**=**"Размер популяции:"**).**grid**(**row **=** 2**,** column**=**20**,** columnspan **=** 2**)**

populationSize **=** tk**.**Entry**(**root**)**

populationSize**.**grid**(**row **=** 3**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Число особей для ликвидации и распределения:"**).**grid**(**row **=** 4**,** column**=**20**,** columnspan **=** 2**)**

numberToDiffuse **=** tk**.**Entry**(**root**)**

numberToDiffuse**.**grid**(**row **=** 5**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Число шагов хемотаксиса:"**).**grid**(**row **=** 6**,** column**=**20**,** columnspan **=** 2**)**

maxChemotaxis **=** tk**.**Entry**(**root**)**

maxChemotaxis**.**grid**(**row **=** 7**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Число шагов репродукции:"**).**grid**(**row **=** 8**,** column**=**20**,** columnspan **=** 2**)**

maxReproduction **=** tk**.**Entry**(**root**)**

maxReproduction**.**grid**(**row **=** 9**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Максимальное число шагов ликвидации и распределения:"**).**grid**(**row **=** 10**,** column**=**20**,** columnspan **=** 2**)**

maxDiffusion **=** tk**.**Entry**(**root**)**

maxDiffusion**.**grid**(**row **=** 11**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Шанс ликвидации и распределения:"**).**grid**(**row **=** 12**,** column**=**20**,** columnspan **=** 2**)**

diffusionChance **=** tk**.**Entry**(**root**)**

diffusionChance**.**grid**(**row **=** 13**,** column **=** 20**,** columnspan **=** 2**)**

tk**.**Label**(**root**,** text**=**"Шаг:"**).**grid**(**row **=** 14**,** column**=**20**,** columnspan **=** 2**)**

step **=** tk**.**Entry**(**root**)**

step**.**grid**(**row **=** 15**,** column **=** 20**,** columnspan **=** 2**)**

rotateBtn **=** tk**.**Button**(**master**=**root**,** text**=**"Обзор"**,** command**=**rotate**,** width **=** 10**)**

rotateBtn**.**grid**(**row **=** 21**,** column **=** 20**,** columnspan **=** 1**)**

calcBtn **=** tk**.**Button**(**master**=**root**,** text **=** "Старт"**,** width **=** 10**,** command **=**

**lambda:** calc**(**int**(**populationSize**.**get**()),** int**(**numberToDiffuse**.**get**()),** int**(**maxChemotaxis**.**get**()),**

int**(**maxReproduction**.**get**()),** int**(**maxDiffusion**.**get**()),** float**(**diffusionChance**.**get**()),** float**(**step**.**get**())))**

calcBtn**.**grid**(**row**=**21**,** column **=** 21**,** columnspan **=** 1**)**

root**.**mainloop**()**

Рисунок 1 – Программная реализация алгоритма BFO

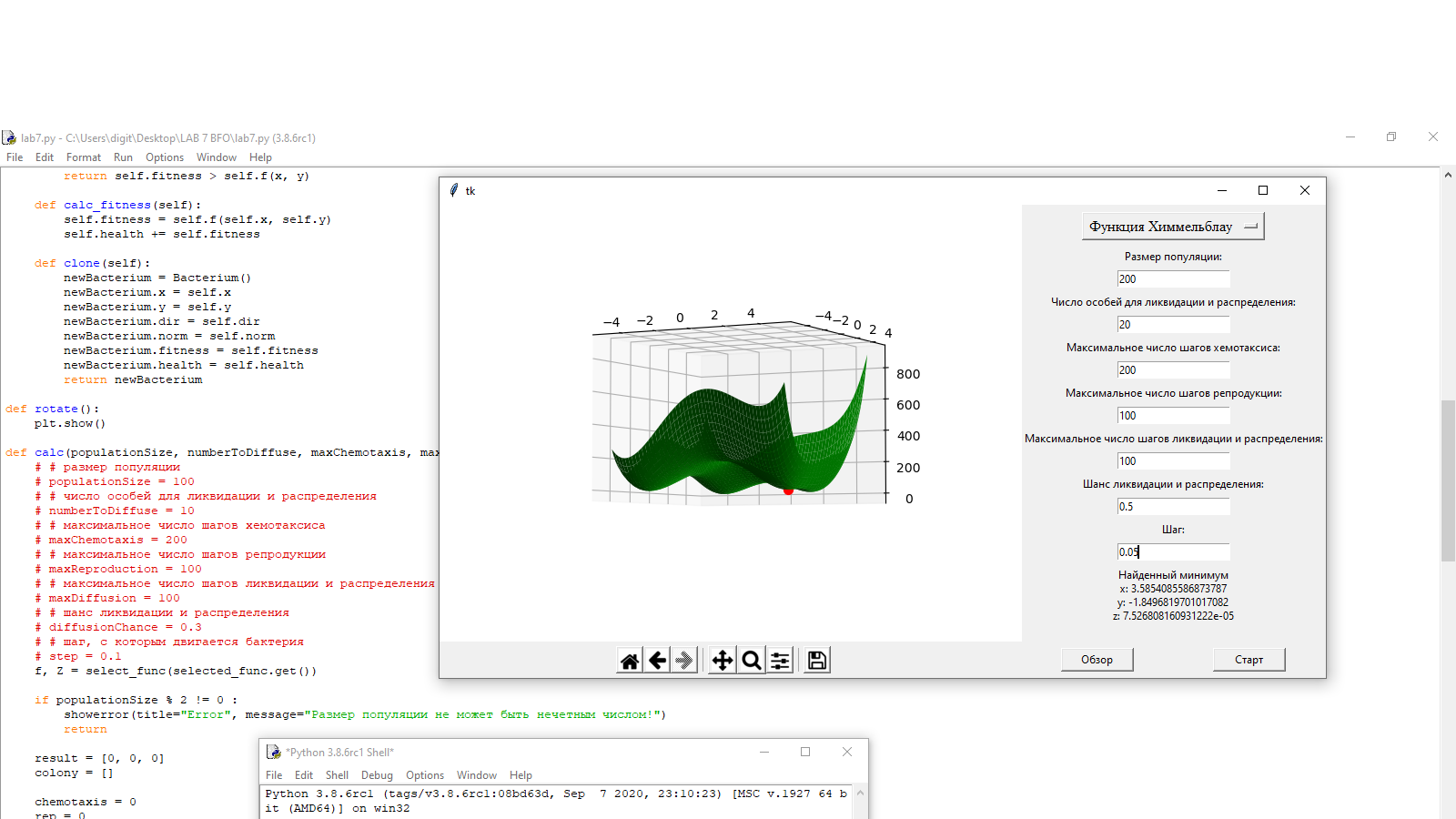


Рисунок 2 – Результат работы BFO на функции Химмельблау

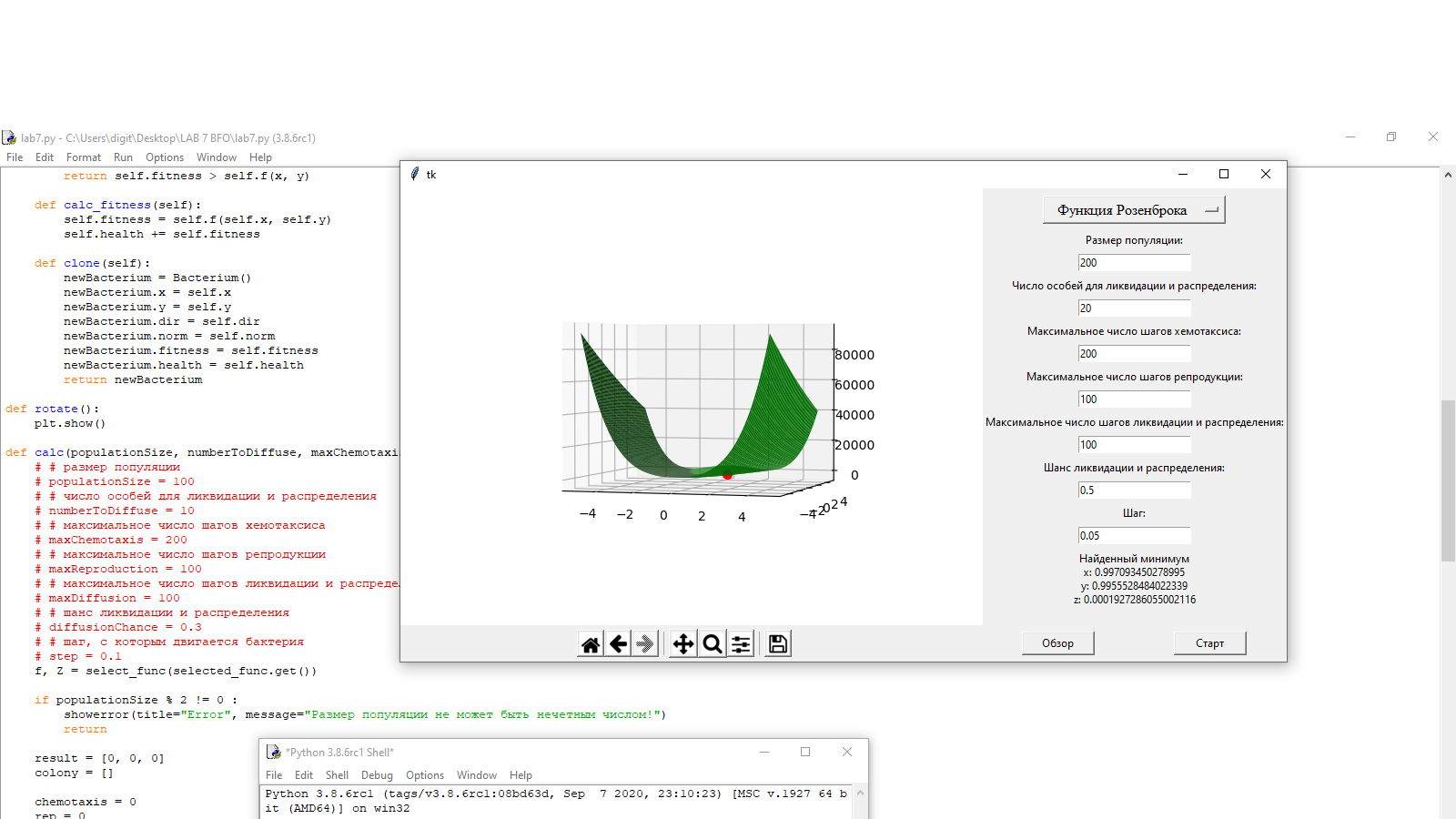


Рисунок 3 – Результат работы BFO на функции Розенброка

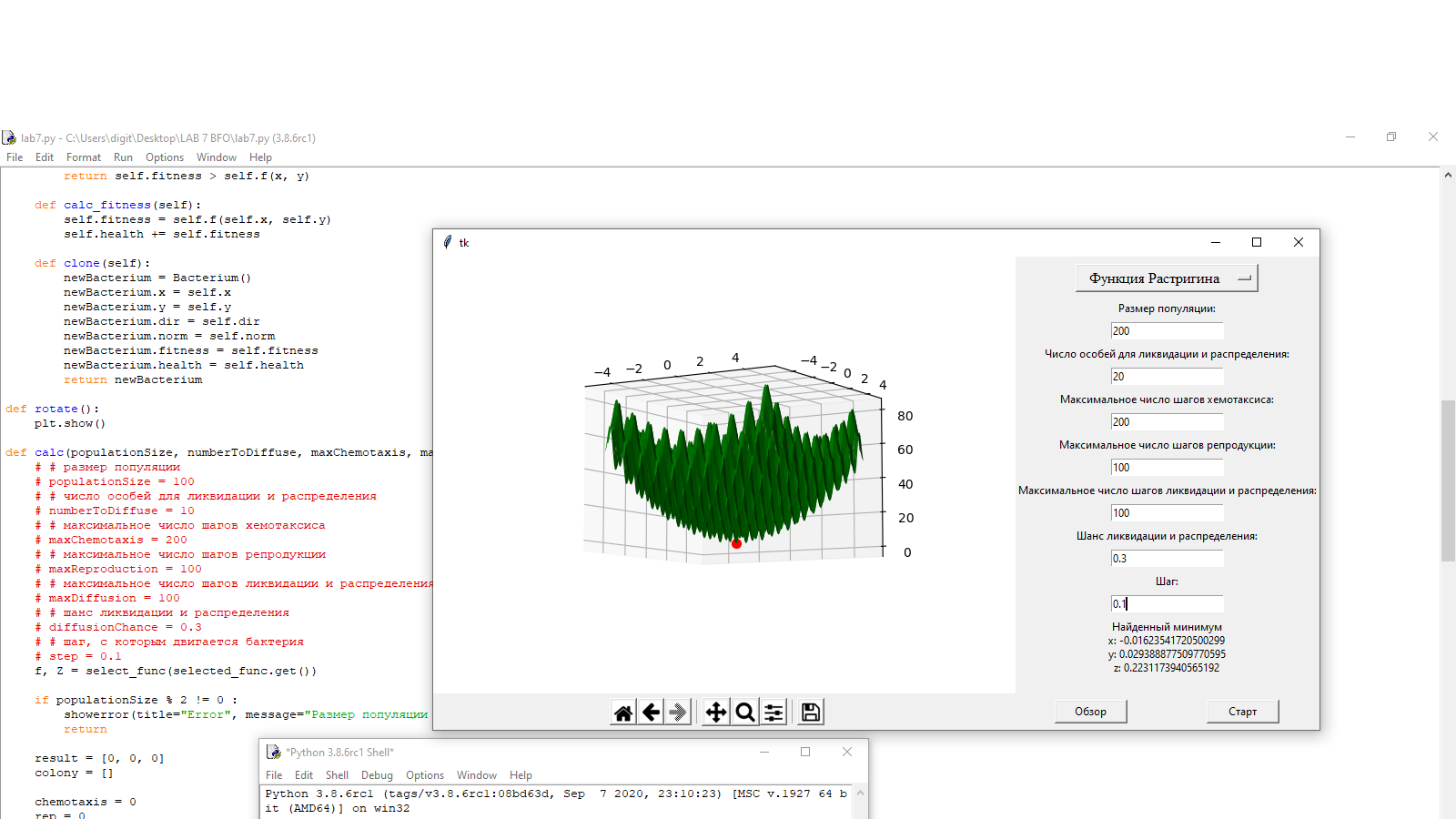


Рисунок 4 – Результат работы BFO на функции Растригина

**Вывод:**

В рассмотренном алгоритме бактериальной оптимизации (BFO)

основан на использовании трёх основных механизмов: хемотаксис, репродукция, ликвидация и рассеивание. Процедура хемотаксиса реализует в алгоритме BFO локальную оптимизацию. Механизм репродукции имеет своей целью ускорение сходимости алгоритма (за счёт сужения области поиска). Процедура ликвидации и рассеивания позволяет бактериям покидать найденные ими локальные максимумы для продолжения поиска глобального максимума.