Глава 38. МЕТОД, ИМИТИРУЮЩИЙ ПОВЕДЕНИЕ СТАИ ОКУНЕЙ

38.1. Постановка задачи

Дана целевая функция $f(x) = f(x_1, x_2, ..., x_n)$, определенная на множестве допустимых решений $D \subseteq R^n$.

Требуется найти условный глобальный минимум функции $f(x)$ на множестве D, т.е. такую точку $x^* \in D$, что

$$f(x^*) = \min_{x \in D} f(x), \tag{38.1}$$

где $x = (x_1, x_2, ..., x_n)^T$, $D = \{x \mid x_i \in [a_i, b_i], i = 1,2,...,n\}$.

Задача поиска максимума функции $f(x)$ сводится к задаче поиска минимума путем замены знака перед функцией на противоположный: $f(x^*) = \max_{x \in D} f(x) = -\min_{x \in D} [-f(x)]$. Функция $f(x)$ может быть многоэкстремальной, поэтому искомое решение в общем случае неединственное.

38.2. Стратегия поиска решения

Метод (Perch School Search – PSS) имитирует поведение стай речного окуня. С целью добывания пищи окуни средних размеров сбиваются в стаи по 5–12 особей. Совсем мелкие окуни образуют стаи, содержащие около 100 особей. Как правило, они берут рыбу, на которую они охотятся, в кольцо (окуневый котел) и из него не выпускают. Окуни атакуют жертв, находящихся ближе к границе котла, постепенно продвигаясь к его центру. Для поиска новых источников пищи окуни используют механизм миграции. Более крупные окуни обычно держатся на глубине, в ямах и омутах и охотятся поодиночке. Они реже, чем мелкие, организуют стаи, но известно, что они объединяются для борьбы с другими хищными рыбами (щуками и судаками). Речной окунь использует весьма агрессивную модель охоты, он активно преследует жертву, иногда выскакивая за ней даже на поверхность воды. Самые крупные окуни, – одиночные, самостоятельные хищники. Это связанно с тем, что крупный окунь уже не нуждается в коллективной охоте и может самостоятельно охотиться на любую, доступную по размеру рыбу. При отсутствии кормовых объектов на постоянном месте обитания, окунь начинает перемещаться в поисках мест, изобилующих мелкой рыбешкой и другим кормом.

При решении задачи поиска глобального условного минимума функции используются конечные наборы $I = \{x^j = (x_1^j, x_2^j, ..., x_n^j)^T, j = 1,2,...,NP\} \subset D$ возможных решений, называемые популяциями, где x^j – особь (окунь) с номером j, NP – размер популяции.

В начале процесса метод порождает популяцию окуней с помощью равномерного на множестве допустимых решений распределения. При этом все решения упорядочиваются по возрастанию значения целевой функции. Затем популяция делится на стаи. Порядок формирования стай: наилучшее решение помещается в первую стаю, следующее – во вторую и т.д. M-е в M-ю стаю, $(M+1)$-е снова в первую стаю, $(M+2)$-е во вторую стаю и т.д. Описанный процесс

соответствует обмену информацией между стаями с целью эффективного приближения к глобальному экстремуму.

В каждой стае определяется лидер по величине целевой функции. Каждая стая организует окуневый котел, в котором происходит охота. При этом реализуется движение всех окуней стаи к своему лидеру, исследуя границы котла. Во время движения запоминается наилучшее положение (в этом положении охота считается удачной, происходит фаза питания окуней). В результате находятся новые лидеры каждой стаи и новые положения ее членов. Среди лидеров стай находится абсолютный лидер и наименее успешный.

В стае, соответствующей абсолютному лидеру, реализуется окуневый котел, в котором тщательно исследуется вся область, занимаемая стаей. В результате находится новый абсолютный лидер.

Затем среди стай выбирается стая с самым слабым лидером. Она перемещается в другую область множества допустимых решений (водоема). Для этого реализуется движение лидера стаи на основе распределения Леви с проверкой принадлежности множеству допустимых решений. Остальные члены стаи генерируются с помощью равномерного закона распределения на параллелепипедном множестве, размер которого по каждой координате определяется удвоенным расстоянием от лидера до ближайшей границы множества D. В полученной стае реализуется окуневый котел, и находится новый лидер.

Остальные стаи совершают плавание в направлении к текущему абсолютному лидеру всей популяции. При этом локальный лидер стаи двигается по прямой к абсолютному лидеру, а остальные окуни этой стаи двигаются параллельно ему. В этом движении все окуни запоминают свою наилучшую позицию и в ней остаются (питаются).

По окончании миграции всех стай популяции абсолютный лидер помещается в множество *Pool*. Происходит новое деление популяции на стаи, и начинается новая глобальная итерация до достижения заданного их числа.

На заключительном шаге в множестве *Pool* организуется взаимодействие лидеров, выявленных на каждой глобальной итерации алгоритма (охота крупных окуней за более серьезной добычей). Заданное число раз в множестве *Pool* выбирается тройка окуней и реализуется операция перекоммутации (path-relinking) [x], в результате которой множество пополняется еще одним решением.

После окончания процедуры перекоммутации среди элементов множества *Pool* находится наилучшее решение, которое считается приближенным решением поставленной задачи.

Предложенный метод является гибридным, так как содержит идеи, использованные в алгоритме лягушек (деление на стаи), алгоритме кукушек (перелет Леви), миграционном алгоритме (движение к лидеру), алгоритме перекоммутации (поиск в множестве *Pool*).

Общая схема работы метода изображена на рис. 38.1.

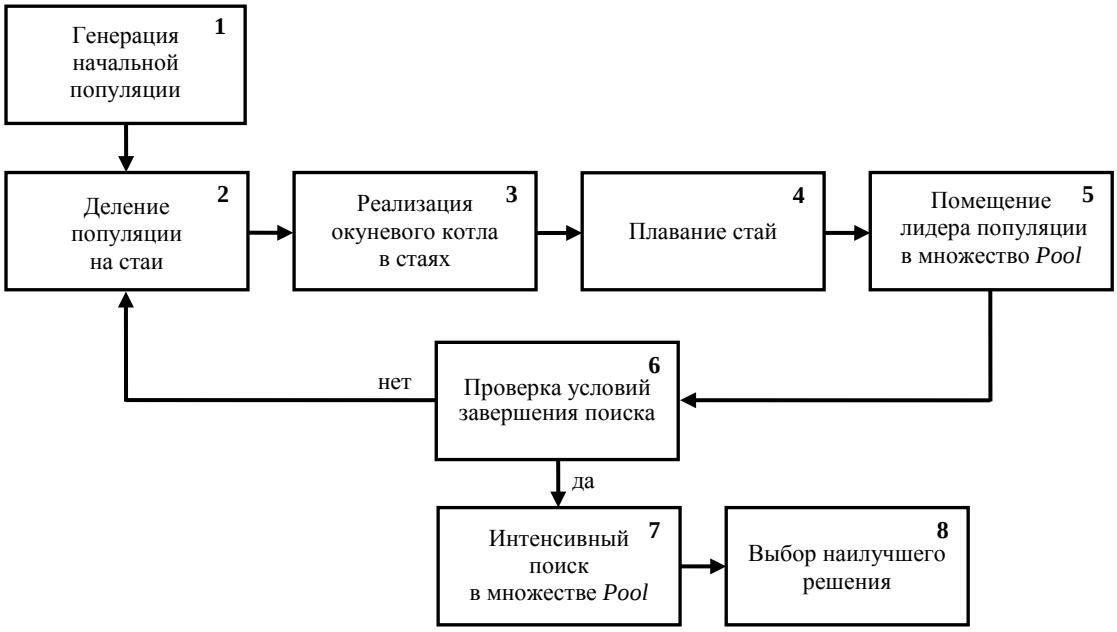


Рис. 38.1. Общая схема работы метода, имитирующего поведение стаи окуней

Рис. 38.2. Охота стаи окуней

Рис. 38.3. Окуневый котел

38.3. Алгоритм решения задачи

Шаг 1. *Задание параметров метода*:

- контролирующий параметр *NStep*, определяющий количество шагов до окончания движения;
- количество стай в популяции M;
- количество окуней в стае s;
- число членов популяции $NP = s \cdot M$;
- останавливающий параметр $Iter_{\max}$, определяющий максимальное количество итераций;
- параметр λ распределения Леви;
- величина шага α;
- максимальное число перекоммутаций $PR_{\max}$;
- число шагов в процедуре перекоммутации Δ_{pr}.

Шаг 2. *Создание начальной популяции окуней.*

Шаг 2.1. Создать популяцию $I = \{x^j = (x_1^j, x_2^j, ..., x_n^j)^T, j = 1, 2, ..., NP\} \subset D$ из NP решений (окуней) со случайно сгенерированными координатами x_i из промежутка $[a_i, b_i]$ с использованием равномерного закона распределения:

$$x_i^j = a_i + rand_i[0,1] \cdot (b_i - a_i),\ i = 1, \ldots, n;\ j = 1, \ldots, NP.$$

где $rand_i[0,1]$ – равномерный закон распределения на отрезке $[0;1]$.

Шаг 2.2. Для каждого решения (окуня) в популяции вычислить значения целевой функции.

Положить $iter = 1$ (счетчик числа глобальных итераций).

Шаг 3. *Деление популяции на стаи.*

Шаг 3.1. Упорядочить решения в популяции по возрастанию значений целевой функции.

Шаг 3.2. Сформировать M стай по s окуней в каждой: наилучшее (с наименьшим значением целевой функции) решение, поместить в первую стаю, следующее – во вторую и т.д., M -е поместить в M -ю стаю, $(M+1)$-е снова в первую стаю и т.д. Результатом являются M стай, содержащих по s окуней каждая, так что $NP = M \cdot s$. Первый помещенный в стаю окунь является ее лидером $x^{loc,m}, m = 1,...,M$. Лидер первой стаи одновременно является лидером всей популяции: $x^{loc,1} = x^{glob}$.

Шаг 4. *Реализация окуневого котла в каждой стае.*

Шаг 4.1. Для каждой стаи $m = 1,...,M$ выполнить следующие действия.

Передвинуть каждого окуня по направлению к лидеру стаи в окрестности ее границы:

$$x^{j,m,k} = x^{j,m} + k\frac{\left(x^{loc,m} - x^{j,m}\right)}{Nstep}, k = 0,1,...,\left[\sigma \cdot Nstep\right]; j = 1,...,s;$$

где $x^{j,m}$ – начальное положение окуня с номером j в стае с номером m; $x^{j,m,k}$ – положение окуня во время движения; $x^{loc,m}$ – положение лидера стаи с номером m; $[\cdot]$ – целая часть числа; значение параметра котла $\sigma \in [0,1;0,5]$ генерируется с помощью равномерного закона распределения на каждой итерации для каждой стаи независимо.

После всех выполненных шагов для каждого окуня найти наилучший шаг (такой шаг, на котором значение целевой функции было наименьшим), а окуню занять эту наилучшую позицию $x^{j,m,new}$:

$$x^{j,m,new} = \underset{k=0,1,...,[\sigma \cdot Nstep]}{\operatorname{argmin}} f\left(x^{j,m,k}\right), j = 1,...,s.$$

Шаг 4.2. В каждой стае определить нового лидера $x^{loc,m,new}, m = 1,...,M$.

Шаг 4.3. Упорядочить стаи по возрастанию значений целевой функции. Первой стае соответствует абсолютный лидер $x^{loc,1} = x^{glob}$, в остальных стаях находится локальный лидер $x^{loc,m}, m = 2,...,M$; стае с номером M соответствует наибольшее значение целевой функции среди лидеров.

Шаг 5. *Плавание стаи с абсолютным лидером.*

Шаг 5.1. Передвинуть каждого окуня по направлению к абсолютному лидеру стаи, двигаясь вдоль соединяющей их прямой (приближаясь, а затем удаляясь в том же направлении):

$$x^{j,1,k} = x^{j,1} + k\frac{\left(x^{glob} - x^{j,1}\right)}{Nstep}, k = 0,1,...,\left[\sigma_1 \cdot Nstep\right]; j = 1,...,s;$$

где значение параметра котла $\sigma_1 \in [1;1,5]$ генерируется с помощью равномерного закона распределения на каждой итерации.

Шаг 5.2. После всех выполненных шагов для каждого окуня стаи найти наилучший шаг (такой шаг, на котором значение целевой функции было наименьшим), а окуню занять эту наилучшую позицию $x^{j,1,new}$:

$$x^{j,1,new} = \underset{k=0,1,...,[\sigma_1 \cdot Nstep]}{\operatorname{argmin}} f\left(x^{j,1,k}\right), j = 1,...,s.$$

Шаг 5.3. В стае определить нового лидера $x^{loc,1,new} = x^{glob,new}$.

Шаг 6. *Плавание стаи с наихудшим лидером.*

Шаг 6.1. Плавание лидера. Новое положение лидера генерируется случайным образом с помощью распределения Леви:

$$x_i^{loc,M,new} = x_i^{loc,M} + \frac{\alpha}{iter} \cdot Levy_i(\lambda), \quad i = 1,\ldots,n,$$

где $x_i^{loc,M}$ – координата положения лидера стаи на текущей итерации, α – величина шага, $\lambda \in (1;3]$, а для генерации случайной величины согласно распределению Леви требуется:

- для каждой координаты $x_i = Levy_i(\lambda)$ с помощью равномерного закона распределения на множестве $[\varepsilon; b_i - a_i]$, где $\varepsilon = 10^{-7}$ – константа различимости, сгенерировать число R_i, $i = 1,...,n$:
  - найти числа $\theta_i = R_i \cdot 2\pi$ и $L_i = \left(R_i + \varepsilon\right)^{-\frac{1}{\lambda}}$, $i = 1,...,n$, где λ – параметр распределения;
  - вычислить значения координат по формулам:

$$Levy_i(\lambda) = L_i \sin\theta_i,\ i = 1,...,\left[\frac{n}{2}\right];\ Levy_i(\lambda) = L_i \cos\theta_i,\ i = \left[\frac{n}{2}\right] + 1,...,n.$$

Если полученное значение координаты $x_i^{loc,M,new}$ не принадлежит множеству допустимых решений, т.е. $x_i^{loc,M,new} \notin [a_i; b_i]$, то процесс его генерации повторяется.

Шаг 6.2. Генерировать новые позиции членов стаи $x^{j,M}$ с помощью равномерного распределения на параллелепипеде, образованном прямым произведением отрезков $[x_i^{loc,M} - \hat{x}_i, x_i^{loc,M} + \hat{x}_i]$, где $\hat{x}_i = \min\{(x_i^{loc,M} - a_i), (b_i - x_i^{loc,M})\}$.

Шаг 6.3. Реализовать окуневый котел в полученной стае.

Передвинуть каждого окуня по направлению к лидеру стаи в окрестности ее границы:

$$x^{j,M,k} = x^{j,M} + k\frac{\left(x^{loc,M,new} - x^{j,M}\right)}{Nstep}, k = 0,1,...,\left[\sigma_3 \cdot Nstep\right];\ j = 1,...,s;$$

где значение параметра котла $\sigma_3 \in [0{,}1; 0{,}5]$ генерируется с помощью равномерного закона распределения на каждой итерации.

После всех выполненных шагов для каждого окуня найти наилучший шаг (такой шаг, на котором значение целевой функции было наименьшим), а окуню занять эту наилучшую позицию $x^{j,M,new}$:

$$x^{j,M,new} = \underset{k=0,1,...,[\sigma_3 \cdot Nstep]}{\operatorname{argmin}} f\left(x^{j,M,k}\right),\ j = 1,...,s.$$

Шаг 6.4. Определить нового лидера стаи $x^{loc,M}$.

Шаг 7. *Плавание остальных стай.*

Для всех стай с номерами $m \in \{2,...,M-1\}$ выполнить следующие действия.

Шаг 7.1. Передвинуть лидера стаи по направлению к абсолютному текущему лидеру:

$$x^{loc,m,k} = x^{loc,m} + k\frac{\left(x^{glob,new} - x^{loc,m}\right)}{Nstep}, k = 0,1,...,\left[\sigma_2 \cdot Nstep\right];$$

где значение параметра $\sigma_2 \in [0,6;0,8]$ генерируется с помощью равномерного закона распределения на каждой итерации для каждой стаи независимо.

Шаг 7.2. Организовать движение остальных членов стаи параллельно лидеру:

$$x^{j,m,k} = x^{j,m} + k\frac{\left(x^{glob,new} - x^{loc,m}\right)}{Nstep}, k = 0,1,...,\left[\sigma_2 \cdot Nstep\right];\ j = 1,...,s.$$

После всех выполненных шагов для каждого окуня найти наилучший шаг (такой шаг, на котором значение целевой функции было наименьшим), а окуню занять эту наилучшую позицию $x^{j,\mathrm{m},new}$:

$$x^{j,\mathrm{m},new} = \underset{k=0,1,...,[\sigma_2 \cdot Nstep]}{\operatorname{argmin}} f\left(x^{j,\mathrm{m},k}\right), j = 1,...,s.$$

Шаг 7.3. Каждому из членов стаи занять наилучшую позицию, достигнутую в процессе плавания. Найти нового лидера стаи $x^{loc,m}, m \in \{2,...,M-1\}$.

Шаг 8. *Нахождение нового абсолютного лидера популяции среди лидеров стай.*

Среди решений, соответствующих лидерам стай, выбрать наилучшее и поместить его в множество $Pool$.

Шаг 9. *Проверка условий завершения поиска.*

Если $iter = Iter_{\max}$, то перейти к шагу 10. Иначе положить $iter = iter + 1$ и перейти к шагу 3.

Шаг 10. *Интенсивный поиск в множестве* $Pool$.

Шаг 10.1. Положить $pr = 1$.

Шаг 10.2. Выбрать три различных случайных решения $x^p_{pool}, x^q_{pool}, x^r_{pool}$ из множества $Pool$.

Шаг 10.3. Найти решение

$$x^{pq}_{pool} = \arg \min_{j=1,...,\Delta_{pr}-1} f\left(x^p_{pool} + j\ (x^q_{pool} - x^p_{pool}) / \Delta_{pr}\right)$$

Шаг 10.4. Добавить решение

$$x^{new} = \arg \min_{j=1,...,\Delta_{pr}-1} f\left(x^{pq}_{pool} + j\,(x^r_{pool} - x^{pq}_{pool}) / \Delta_{pr}\right)$$

в множество $Pool$. Положить $pr = pr + 1$.

Шаг 10.5. Если $pr > PR_{\max}$, то перейти к шагу 11. Иначе – к шагу 10.2.

Шаг 11. *Выбор наилучшего решения*. Среди решений в множестве $Pool$ выбрать наилучшее. Считать его приближенным решением поставленной задачи.

38.4. Программное обеспечение

На основе изложенного алгоритма сформирована программа поиска глобального минимума функций многих переменных. Среда разработки Microsoft Visual Studio 2019, язык программирования C# .

Возможности программы позволяют изучить алгоритм метода, а также влияние

параметров метода на результат его работы. В список выбираемых функций включены стандартные многоэкстремальные тестовые функции двух переменных, для которых известно точное решение – положение точки (точек) максимума и соответствующее значение целевой функции (табл. П.1). Поскольку сформированная программа решает задачу поиска минимума, то перед исследуемыми тестовыми функциями поставлен знак минус.

Программа состоит из главного окна (рис. 38.4) и окна пошаговой работы метода (рис. 38.5).

В главном окне метода пользователь выбирает оптимизируемую функцию, задает множество допустимых решений и параметры метода. Также в этом окне отображаются результаты работы метода. При необходимости можно создать отчет – текстовый файл с десятью сериями решений одной и той же задачи с одними и теми же значениями параметров. Внизу окна расположены кнопки «Описание алгоритма» и «Выход».

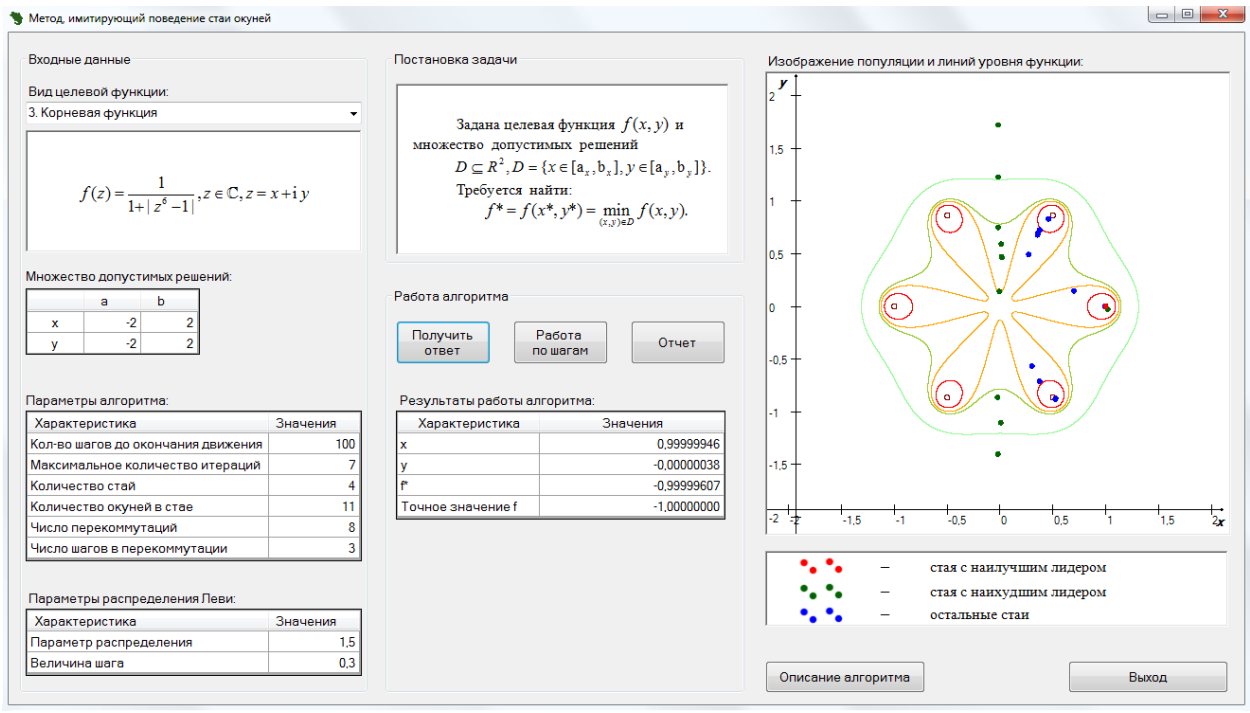


Рис. 38.4. Главное окно программы метода стаи окуней

Во время пошаговой работы отображаются схема работы метода, результаты выполнения каждого шага и окончательный результат работы метода. В центральной части окна расположен график изменения средней и наилучшей популяций.

В данном окне предусмотрена возможность выполнить N итераций по нажатию кнопки «Выполнить N итераций». Число N задается пользователем.

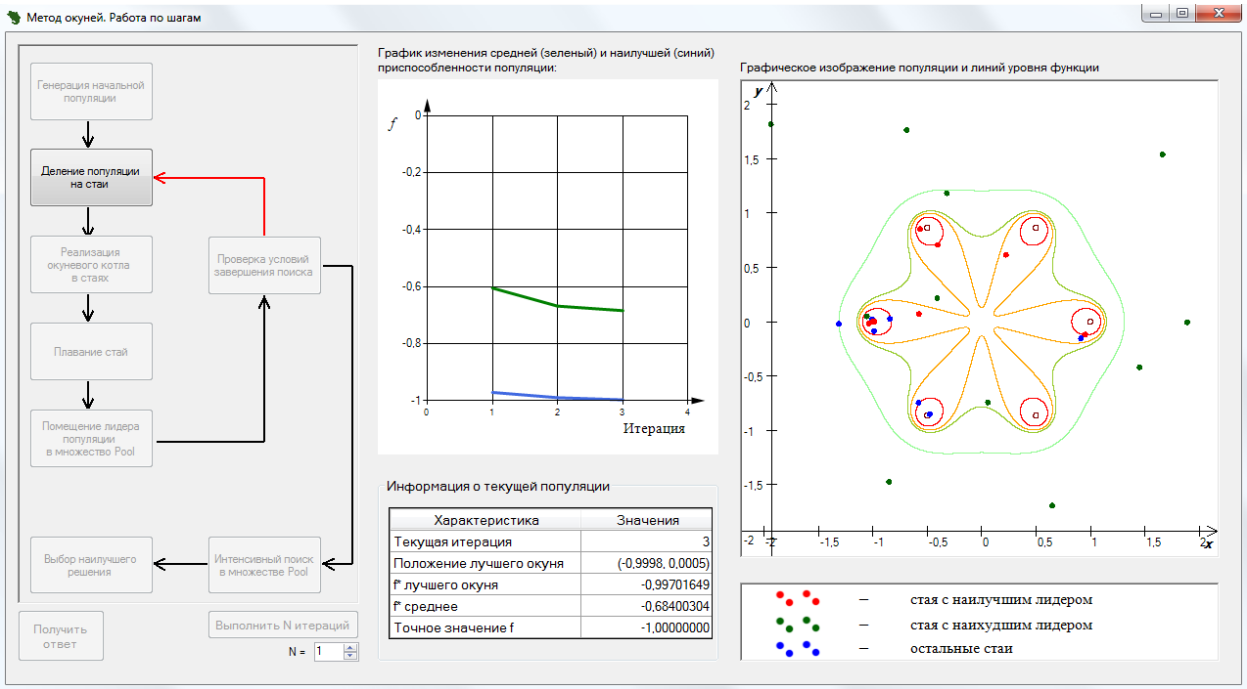


Рис. 38.5. Окно пошаговой работы метода стаи окуней

38.5. Тестовые примеры

Пример 38.1. Рассмотрим измененную функцию «Кожа». Зададим множество допустимых решений $x \in [-5;5],\ y \in [-5;5]$.

Выберем следующие параметры метода:

- контролирующий параметр $NStep = 100$, определяющий количество шагов до окончания движения;
- количество стай в популяции $M = 4$;
- количество окуней в стае $s = 15$ (число членов популяции $NP = s \cdot M$);
- останавливающий параметр $Iter_{\max} = 12$, определяющий максимальное количество итераций;
- параметр $\lambda = 1,5$ распределения Леви;
- величина шага $\alpha = 0,3$;
- максимальное число перекоммутаций $PR_{\max} = 10$;
- число шагов в процедуре перекоммутации $\Delta_{pr} = 5$.

На рис. 38.6. представлена популяция на начальной ($iter = 1$), промежуточных ($iter = 5$, $iter = 9$) и конечной ($iter = 12$) итерациях.

Результаты работы метода:

- решение с наилучшим положением $(x^*; y^*) = (-3,31581608;\ -3,07450023)$;
- значение целевой функции $f(x^*; y^*) = -14,06053734$;
- отклонение от точного решения $\Delta = 0$.

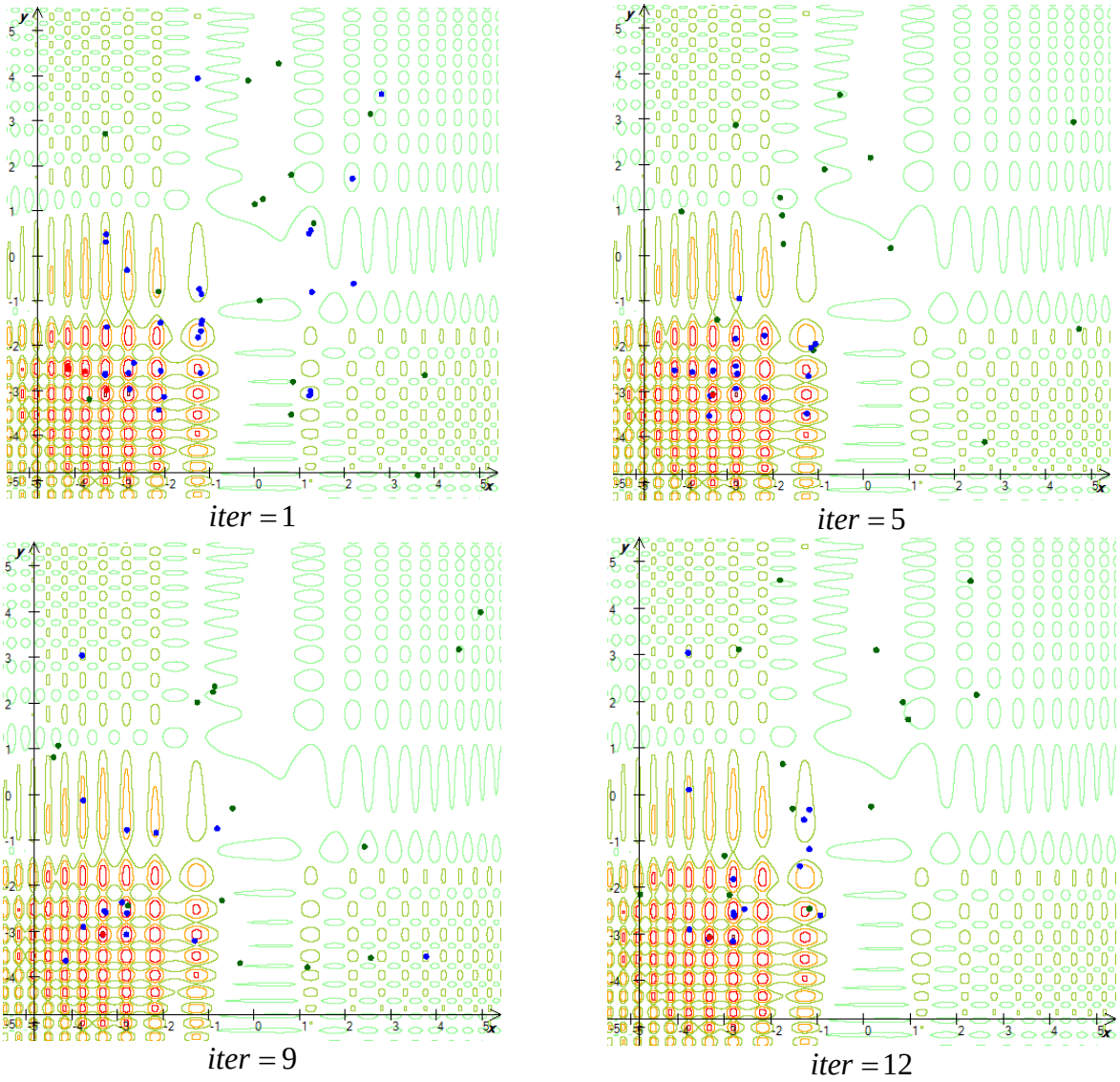


Рис. 38.6. Начальная, промежуточные и конечная популяции

Изменение наилучшего значения целевой функции при переходе от одной итерации поиска к другой представлено на рис. 38.7.

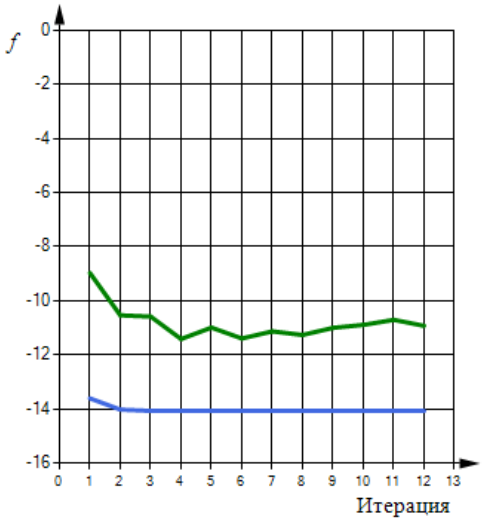


Рис. 38.7. Графики изменения среднего (зеленый) и наилучшего (синий) значения целевой функции

Пример 38.2. Рассмотрим измененную функцию Экли. Зададим множество допустимых решений $x \in [-10;10],\ y \in [-10;10]$.

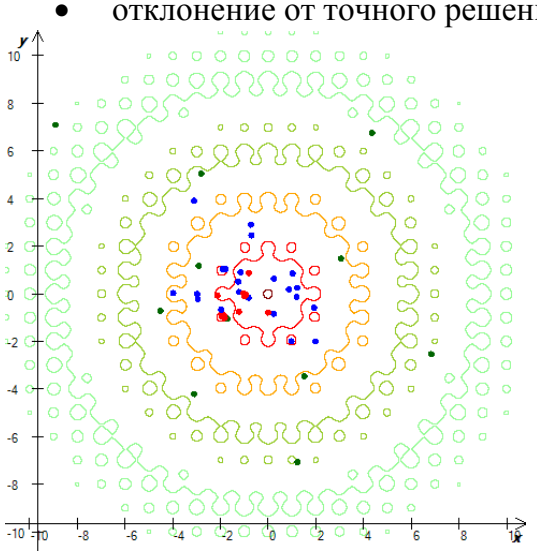
Выберем следующие параметры метода:

- контролирующий параметр $NStep = 100$, определяющий количество шагов до окончания движения;
- количество стай в популяции $M = 4$;
- количество окуней в стае $s = 11$ (число членов популяции $NP = s \cdot M$);
- останавливающий параметр $Iter_{\max} = 7$, определяющий максимальное количество итераций;
- параметр $\lambda = 1{,}5$ распределения Леви;
- величина шага $\alpha = 0{,}6$;
- максимальное число перекоммутаций $PR_{\max} = 8$;
- число шагов в процедуре перекоммутации $\Delta_{pr} = 3$.

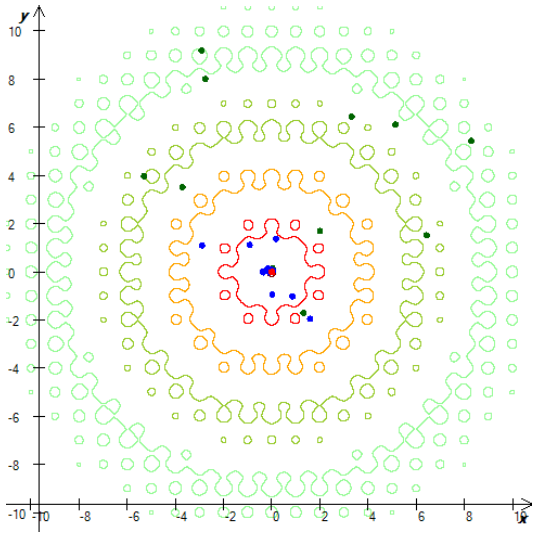
На рис. 38.8. представлена популяция на начальной ($iter = 1$), промежуточных ($iter = 3$, $iter = 5$) и конечной ($iter = 7$) итерациях.

Результаты работы метода:

- решение с наилучшим положением $(x^*; y^*) = (0{,}00002203;\ -0{,}00001490)$;
- значение целевой функции $f(x^*; y^*) = -19{,}99992561$;
- отклонение от точного решения $\Delta = 0$.



$iter = 1$



$iter = 3$

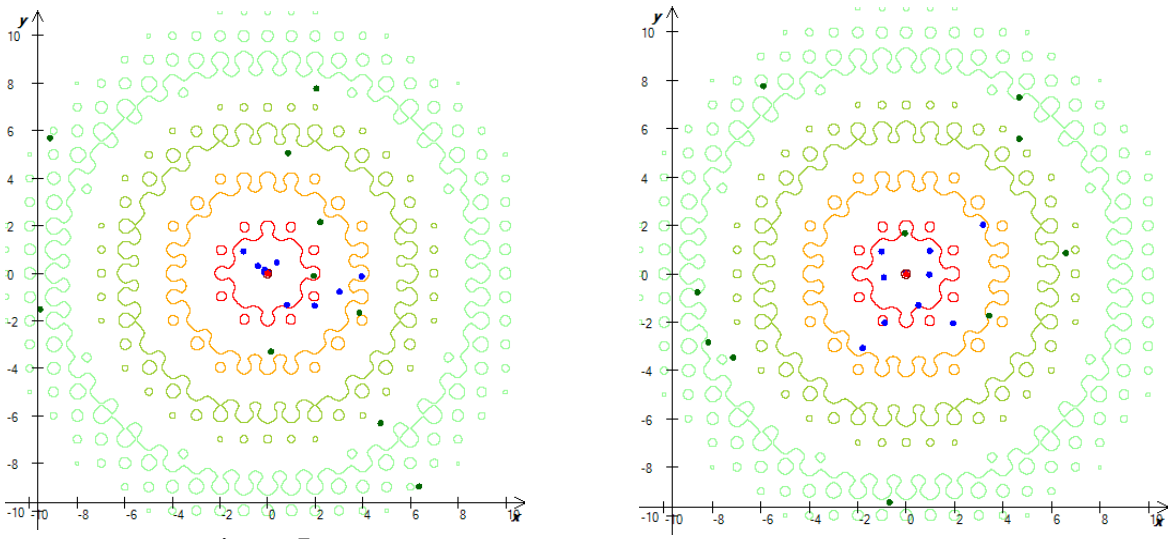


Рис. 38.8. Начальная, промежуточные и конечная популяции

Изменение наилучшего значения целевой функции при переходе от одной итерации поиска к другой представлено на рис. 38.9.

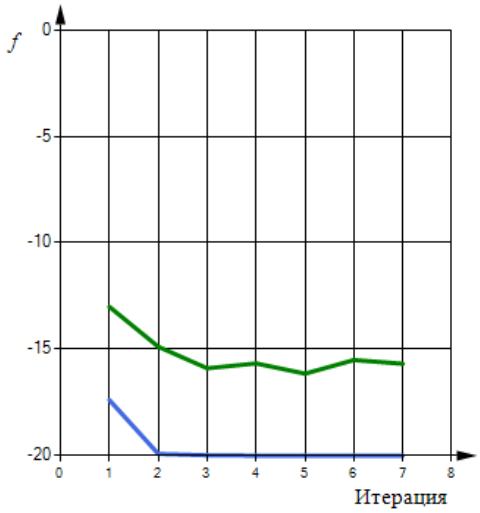


Рис. 38.9. Графики изменения среднего (зеленый) и наилучшего (синий) значения целевой функции

38.6. Анализ эффективности метода

38.6.1. Рекомендации по выбору параметров

Контролирующий параметр *NStep*, определяющий количество шагов до окончания движения. Для многоэкстремальных функций с большим расстоянием между экстремумами следует полагать $NStep = 100, ..., 200$ при условии $NP \geq 100$. Для функций с малым количеством локальных минимумов достаточно $NStep = 100$.

Максимальное число итераций $Iter_{\max}$, определяющий максимальное количество итераций. Учитывая суть алгоритма, необходимо определять $Iter_{\max} \geq 3$ различно, в зависимости от количества минимумов функции в области допустимых решений.

Для многоэкстремальных функций рекомендуется использовать:

- для малых популяций ($NP = 3 \le s \cdot M < 20$) $Iter_{\max} = 4,...,20$;
- для средних популяций ($NP = 20 \le s \cdot M \le 40$) $Iter_{\max} = 4,...,15$;
- для больших популяций ($NP = s \cdot M > 40$) $Iter_{\max} = 4,...,12$.

Для функций с малым количеством локальных минимумов рекомендуется использовать:

- для малых популяций $NP = s \cdot M \le 20$ $Iter_{\max} = 4,...,10$;
- для средних популяций $NP = 20 \le s \cdot M \le 40$ $Iter_{\max} = 3,...,8$;
- для больших популяций $NP = s \cdot M > 20$ $Iter_{\max} = 3,...,7$.

Количество стай в популяции M. По сути алгоритма следует выбирать $M \ge 3$.

Количество окуней в стае s следует выбирать в зависимости от количества минимумов функции и максимального числа итераций $Iter_{\max}$.

Максимальное число перекоммутаций $PR_{\max}$ рекомендуется использовать $PR_{\max} = 5,...,20$.

Число шагов в процедуре перекоммутации $\Delta_{pr} = 5,...,30$.

Параметр распределения Леви λ **рекомендуется использовать** $\lambda \in (1;3]$.

Величину шагов α для стаи с наихудшим лидером рекомендуется использовать $\alpha \in [0,3;0,7]$.

38.6.2. Анализ работы метода при различных значениях параметров

В данном разделе приводится статистический анализ и сравнение работы метода имитации поведения стаи окуней при различных значениях его параметров. Исследуемый метод применялся к некоторым функциям из набора тестовых функций (табл. П.1) с учетом замечаний из пункта 38.4. Для каждой функции проводились серии из 100 решений одной и той же задачи с одними и теми же значениями параметров. Для полученной выборки $\left\{ f^1, f^2, ..., f^{100} \right\}$ вычислялись:

- среднее значение отклонения полученного решения от точного $\overline{\Delta f} = \frac{1}{100} \sum_{i=1}^{100} \Delta f_i$, где $\Delta f_i = | f(x^*) - f^i |$;
- наименьшее значение отклонения $\Delta f_{best} = \min_i \Delta f_i$;
- среднеквадратическое отклонение $\overline{\sigma}_f = \sqrt{\overline{S}_{100}}$, где $\overline{S}_{100} = \frac{1}{99} \sum_{i=1}^{100} (\Delta f_i - \overline{\Delta f})^2$;
- количество успехов $n_{\text{усп}}$ (попаданий лучшей точки в ε-окрестность точного решения, $\varepsilon = \frac{1}{1000} \max_{i=1,...,n} | b_i - a_i |$).

Результаты, полученные для каждой функции, представлены в табл. 38.1-38.3.

Таблица 38.1. Влияние параметров метода. Параболическая функция

Параметры метода								$\overline{\Delta f}$	Δf_{best}	$\overline{\sigma}_f$	$n_{усп}$
NStep	$Iter_{\max}$	M	s	$PR_{\max}$	Δ_{pr}	α	λ				
100	4	4	3	8	3	0,6	1,5	0,000253	0	0,000383	72
100	6	4	3	8	3	0,6	1,5	0,000209	0	0,000520	82
100	10	4	3	8	3	0,6	1,5	0,000045	0	0,000206	96

100	4	4	7	8	3	0,6	1,5	0,000003	0	0,000007	100
100	3	4	1	8	3	0,6	1,5	0,000018	0	0,000027	100
100	4	4	11	8	3	0,6	1,5	0	0	0,000001	100

Таблица 38.2. Влияние параметров метода. Функция «Кожа»

Параметры метода								$\overline{\Delta f}$	Δf_{best}	$\overline{\sigma}_f$	n_{ycn}
NStep	$Iter_{\max}$	M	s	$PR_{\max}$	Δ_{pr}	α	λ				
100	20	4	3	10	5	0,6	1,5	0,076972	0	0,125540	47
100	10	4	3	10	5	0,6	1,5	0,292875	0,000002	0,586289	30
100	12	4	5	10	5	0,3	1,5	0,033539	0	0,056396	56
100	10	4	15	10	5	0,6	1,5	0,004537	0	0,022553	92
100	10	4	15	10	5	0,3	1,5	0,028184	0,000001	0,068515	66
100	12	4	15	10	5	0,3	1,5	0,001513	0	0,007861	97
100	5	4	15	10	5	0,3	1,5	0,005969	0	0,014755	86
100	5	4	15	10	5	0,3	1,1	0,004423	0	0,012657	89
100	5	4	15	20	5	0,3	1,5	0,006161	0	0,015456	87

Таблица 38.3. Влияние параметров метода. Функция Экли

Параметры метода								$\overline{\Delta f}$	Δf_{best}	$\overline{\sigma}_f$	n_{ycn}
NStep	$Iter_{\max}$	M	s	$PR_{\max}$	Δ_{pr}	α	λ				
100	4	4	3	8	3	0,6	1,5	0,235162	0,000021	0,608738	70
100	10	3	3	8	3	0,6	1,5	0,194856	0,000256	0,617575	89
100	4	4	5	8	3	0,6	1,5	0,023385	0,000256	0,031809	93
100	4	4	11	8	3	0,6	1,5	0,002806	0,000065	0,003627	100
100	7	4	11	8	3	0,6	1,5	0,000106	0	0,000197	100

Анализ работы метода, имитирующего поведение стаи окуней, показал, что результат работы алгоритма зависит от размера популяции.

При большой популяции за малое количество итераций находится глобальный экстремум, причем, не все окуни оказываются в глобальном минимуме: часть популяции остается в локальных минимумах.

Таким образом, данный метод позволяет находить не только глобальный минимум, но и некоторые локальные минимумы.

Алгоритм имеет несколько недостатков. На функциях по типу «Кожа» производится большое количество пересчета значений целевой функции, что влияет на скорость выполнения итераций. Также при малых популяциях на функциях, у которых глобальные минимумы находятся на большом расстоянии друг от друга, есть риск того, что популяции «застрянут» в локальных минимумах.