ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”**

Факультет Информационных технологий и программирования

Направление подготовки 010400.68.04 Специализация Технологии проектирования и разработки программного обеспечения

Квалификация (степень) Магистр

Специальное звание

Кафедра Компьютерных технологий Группа 6538

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

**на тему**

## Адаптивная настройка параметров эволюционных алгоритмов с помощью обучения с подкреплением

Автор магистерской диссертации Рост А. Ю. (подпись)

Научный руководитель Шалыто А. А. (подпись)

Руководитель магистерской программы (подпись)

**К з а щ и т е д о п у с т и т ь**
Зав. кафедрой Васильев В. Н. (подпись)

“____”_____2015 г.

Санкт-Петербург, 2015 г.

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ ..... 4
1. Обзор предметной области ..... 5
1.1. Эволюционные алгоритмы ..... 5
1.2. Обзор существующих методов настройки параметров ЭА ..... 6
1.2.1. Метод EARPC ..... 6
1.2.2. Обучение с подкреплением ..... 7
1.2.3. Настройка параметров ЭА как задача для обучения с подкреплением ..... 10
1.2.4. Метод, предложенный Karafotias et al. ..... 11
1.3. Выводы по главе 1 ..... 16
2. Предлагаемые методы настройки параметров ЭА ..... 17
2.1. Цель работы ..... 17
2.2. Метод на основе алгоритмов EARPC и UTree ..... 17
2.3. Метод с адаптивным выделением множества действий ..... 18
2.4. Выводы по главе 2 ..... 19
3. Результаты экспериментов ..... 20
3.1. Описание экспериментов ..... 20
3.1.1. Значения параметров ..... 21
3.1.2. Описание результатов ..... 22
3.2. Сфера ..... 22
3.3. Функция Розенброка ..... 26
3.4. Функция Леви ..... 27
3.5. Функция Растригина ..... 31
3.6. Выводы по главе 3 ..... 34
ЗАКЛЮЧЕНИЕ ..... 39

# ВВЕДЕНИЕ

Эффективность работы эволюционного алгоритма зависит от выбора значений его параметров. Подбор параметров может осуществляться до запуска эволюционного алгоритма. Однако оптимальные значения параметров могут изменяться в ходе работы алгоритма. Поэтому необходим метод адаптивной настройки параметров в процессе оптимизации.

Значения параметров эволюционного алгоритма лежат в заданном интервале значений. Задачу выбора значений параметров дискретизируют, разделяя диапазон допустимых значений параметра на интервалы. Разбиение на интервалы может производиться до запуска алгоритма и не меняться в процессе его работы. Однако изменение разбиения во время работы способствует улучшению работы алгоритма.

Существуют алгоритмы адаптивной настройки параметров эволюционного алгоритма, в которых вероятность выбора значения параметра пропорциональна эффективности его применения. Одним из таких алгоритмов является *EARPC*. В данном методе интервал допустимых значений разбивается в ходе работы алгоритма. Также существует метод настройки параметров эволюционного алгоритма с помощью обучения с подкреплением. Однако в данном подходе разбиение диапазона допустимых значений производится до запуска алгоритма.

В данной работе предлагаются два метода адаптивной настройки параметров эволюционного алгоритма. Один из них является улучшением существующего метода настройки параметров с помощью обучения с подкреплением за счет разбиения диапазона допустимых значений параметра в ходе работы алгоритма при помощи алгоритма *EARPC*. Второй метод основан на применении $Q$-обучения с адаптивным выделением множества действий агента.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Эволюционные алгоритмы

Эволюционный алгоритм(ЭА) [1, 2] часто применяются для решения задач оптимизации. Данный подход основан на идеях, заимствованных из биологической эволюции: естественный отбор, мутация, скрещивание и наследование признаков. Каждая итерация алгоритма характеризуется набором особей, называемым поколением. Начальное поколение обычно формируется случайным образом. На множестве особей вводят функции приспособленности, чтобы количественно оценивать, насколько данная особь близка к верному решению. Наиболее приспособленные особи имеют большую вероятность быть выбранными для создания нового поколения. При помощи оператора скрещивания (кроссовера) по двум особям текущего поколения создается новая особь для следующего поколения. Оператор мутации вносит в особь малые случайные изменения. Общая схема эволюционного алгоритма представлена на листинге 1.

Листинг 1 – Общая схема эволюционного алгоритма

```
1: Создать начальное поколение
2: Вычислить значение функции приспособленности для каждой особи
3: while (условие останова эволюционного алгоритма не выполнено) do
4:     Выбирается подмножество особей текущего поколения
5:     Применяя операторы мутации и кроссовера к выбранным особям, создаются новые особи
6:     Вычисляется значение функции приспособленности для созданных особей
7:     Путем замены новыми особями наименее приспособленных в текущем поколении формируется новое поколение
8: end while
```

В качестве критерия останова часто используют следующие условия [3]:

- найдено верное решение;
- достигнуто заданное количество поколений;
- превышено заданное время работы;
- превышено заданное число вычислений функции приспособленности;
- за заданное число поколений не произошло улучшение решения.

Эволюционные алгоритмы применяются для решения задач оптимизации, к которым точные алгоритмы не применимы. Стоит отметить, что эф-

фективность работы эволюционного алгоритма сильно зависит от выбора значений его параметров, таких как вероятность мутации, вероятность скрещивания, число особей в поколении. Значения параметров зависят не только от эволюционного алгоритма, но и от решаемой задачи оптимизации. Подбор параметров может осуществляться до запуска эволюционного алгоритма. Однако оптимальные значения параметров могут изменяться в ходе работы алгоритма. Поэтому необходим метод адаптивной настройки параметров в процессе оптимизации.

## 1.2. Обзор существующих методов настройки параметров ЭА

Рассмотрим формальную постановку задачи адаптивной настройки параметров ЭА. Имеется набор $\{v_1, ..., v_n\}$ из $n$ параметров ЭА, каждый из которых может принимать значения из некоторого дискретного набора или из непрерывного интервала значений. Целью алгоритма является выбор таких значений параметров $v_i$, чтобы ЭА работал наиболее эффективно.

Большинство алгоритмов адаптивной настройки параметров ЭА можно отнести к классу методов сопоставления вероятности (probability matching techniques), в которых вероятность выбора значения параметра пропорциональна эффективности его применения [4]. В данной работе использовался один из наиболее эффективных алгоритмов данного класса, называемый *EARPC*.

### 1.2.1. Метод EARPC

В методе *EARPC* [5] выбор значений параметров происходит независимо друг от друга. В данном методе для выбора значения параметра диапазон допустимых значений делится во время работы алгоритма на два подынтервала.

Схема работы алгоритма *EARPC* представлена на листине 2. Будем называть назначением параметров набор $(v)$ подобранных значений параметров $(v_1, .., v_n)$. В ходе работы алгоритма каждому назначению параметров $(v)$ соответствует некоторый функционал качества $q((v))$. Выбор новых значений параметров осуществляется следующим образом. Ранее используемые назначения параметров разбиваются на два кластера $c_1$ и $c_2$, например, с помощью алгоритма *k-means*. Затем для каждого параметра $v_i$ интервал его допустимых значений разбивается на два подынтервала. Для этого необходимо выбрать подходящую точку разбиения. Для этого все ранее используемые назначения параметра $v_i$ сортируются по возрастанию. В качестве кандидатов на

точку разбиения рассматриваются средние значения между двумя соседними назначениями в полученной упорядоченной последовательности. Для каждого кандидата $s$ на точку разбиения множество ранее используемых назначений разбивается в соответствии с $s$ на два множества $p_1$ и $p_2$. Во множестве $p_1$ находятся все ранее используемые назначения параметра $v_i$, меньшие или равные $s$, а в множестве $p_2$ находятся все ранее используемые назначения параметра $v_i$, большие $s$. Обозначим $c_i(p_j)$ – подмножество $p_j$, где $i, j \in \{1, 2\}$, соответствующее кластеру $c_i$. Для каждого разбиения по формуле (1) считается *энтропия*. В качестве итоговой точки разбиения $s$ выбирается та, при которой полученная энтропия минимальна. Для множеств $p_1$ и $p_2$ считается среднее качество $Q_1$ и $Q_2$ соответственно. Множеству $p_1$ соответствует интервал $[v_{min}, s]$, а множеству $p_2$ – интервал $(s, v_{max}]$, где $v_{min}$ и $v_{max}$ нижняя и верхняя границы диапазона допустимых значений параметра $v_i$ соответственно. Из двух данных интервалов значений случайным образом выбирается один, при этом вероятность выбора первого интервала пропорциональна $Q_1$, а второго – $Q_2$. Затем значение параметра $v_i$ случайным образом выбирается из соответствующего множества.

$$
\begin{aligned}
e_{p_1} &= -\frac{|c_1(p_1)|}{|p_1|} ln(\frac{|c_1(p_1)|}{|p_1|}) - \frac{|c_2(p_1)|}{|p_1|} ln(\frac{|c_2(p_1)|}{|p_1|}), \\
e_{p_2} &= -\frac{|c_1(p_2)|}{|p_2|} ln(\frac{|c_1(p_2)|}{|p_2|}) - \frac{|c_2(p_2)|}{|p_2|} ln(\frac{|c_2(p_2)|}{|p_2|}), \\
H &= \frac{|p_1|}{|c_1|} e_{p_1} + \frac{|p_2|}{|c_2|} e_{p_2}
\end{aligned}
\tag{1}
$$

Недавно был предложен эффективный метод настройки параметров ЭА с помощью обучения с подкреплением. Однако сравнение эффективности применения данного метода и алгоритма *EARPC* не проводилось. В данной работе проводится сравнение данных подходов, а также предлагается новый метод адаптивного выбора параметров ЭА. Далее рассматриваются основные принципы работы алгоритмов обучения с подкреплением и метод, предложенный в работе [6].

### 1.2.2. Обучение с подкреплением

Алгоритмы обучения с подкреплением [7, 8] часто используются для выбора стратегий поведения в интерактивной среде. Большинство таких алгорит-

Листинг 2 – Алгоритм *EARPC* в случае деления на два подынтервала.

1: Ранее выбранные назначения параметров $\{(v_1, .., v_n)\}$ разбиваются на два кластера $c_1$ и $c_2$ с помощью алгоритма *k-means*.
2: **for** параметр $v_i$ **do**
3: Отсортировать назначения параметров по значению *i*-ого
4: $H_{best} \leftarrow \infty$
5: **for** точка разбиения $s = \frac{v_{ij}+v_{i(j+1)}}{2}$ **do**
6: Разбить назначения в соответствии с точкой разбиения *s* на множества $p_1$ и $p_2$
7: Рассчитать энтропию $H$ разбиения по точке *s* по формуле (1)
8: **if** $H_{best} < H$ **then**
9: $H_{best} \leftarrow H$
10: Запомнить множества $p_1$ и $p_2$
11: **end if**
12: **end for**
13: $Q_1 = \frac{1}{|p_1|} \sum\limits_{\mathbf{v} \in p_1} q(\mathbf{v})$, $Q_2 = \frac{1}{|p_2|} \sum\limits_{\mathbf{v} \in p_2} q(\mathbf{v})$
14: Случайным образом выбрать интервал значений, при этом вероятность выбора первого пропорциональна $Q_1$, а второго – $Q_2$
15: Случайным образом выбрать значение параметра $v_i$ из выбранного интервала
16: **end for**

мов не требуют заранее подобранных тестовых примеров, так как их обучение происходит одновременно с применением накопленного опыта.

Принцип работы алгоритма обучения с подкреплением представлен на схеме 1. У агента есть некоторый набор возможных действий. На каждом шаге алгоритма агент воздействует на среду, которая находится в некотором состоянии, выбирая одно из возможных действий и применяя его к среде. В следствие этого среда может перейти в новое состояние. За выбор действия агент получает численную награду. Задачей агента является максимизация суммарной награды. Действие, выбранное агентом, определяет не только полученную награду, но и состояние, в которое перейдет среда после его применения.

Задачу обучения с подкреплением в большинстве случаев можно описать как *марковский процесс принятия решений*. Для этого необходимо определить:

— дискретное множество состояний среды $S$;
— дискретное множество действий агента $A$;
— функцию награды $R : S \times A \to \mathbb{R}$;

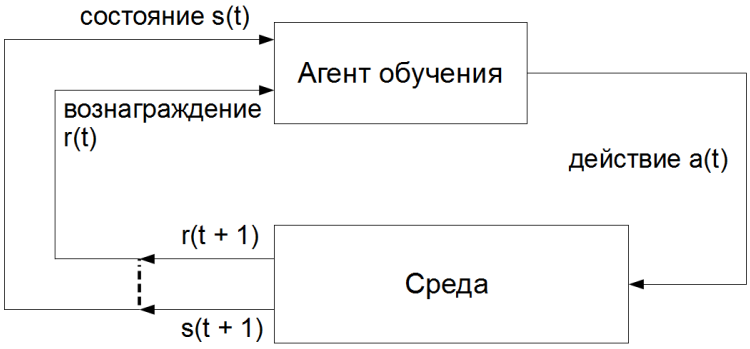



Рисунок 1 – Схема алгоритма обучения с подкреплением.

— функцию переходов $T : S \times A \times S \to \mathbb{R}$ При этом $T(s, a, s')$ определяет вероятность перехода из состояния $s$ в состояние $s'$ после применения действия $a$.

Выделяют класс алгоритмов обучения с подкреплением, строящих модель среды, которые используют функцию награды $R$ и функцию переходов $T$ для определения стратегии поведения. В частности, возможны стратегии в которых алгоритм будет получать незначительную награду в течение некоторого времени, чтобы достичь некоторого состояния среды, которому соответствует большая ожидаемая награда. В рамках данной работы такие алгоритмы не рассматривались.

#### 1.2.2.1. Q-обучение

Алгоритм $Q$-обучения [9, 10] относится к классу алгоритмов обучения с подкреплением не строящих модель среды. Псевдокод алгоритма представлен на листинге 3. Во время работы алгоритма аппроксимируется функция полезности $Q : S \times A \to \mathbb{R}$, которая описывает ожидаемую награду за действие $a$ в состоянии $s$. Для расчета значений $Q$ обычно используют *TD*-обучение(temporal difference learning). При этом значения $Q$ изменяются по формуле $Q(s, a) = Q(s, a) + \alpha(\gamma Q(s', a') - Q(s, a))$, где $\alpha$ – скорость обучения, $\gamma$ – дисконтный фактор.

Выбор действия определяется стратегией исследования среды. Одна из самых простых стратегий - *жадная* заключается в том, чтобы выбирать действие, за которое самое большое ожидаемое вознаграждение, т.е. $\arg\max_a \{Q(s, a)\}$. Однако в таком случае агент склонен выбирать локально максимальное значение награды, недостаточно исследовав среду. Для улучшения жадной стратегии можно выбирать с вероятностью $\varepsilon$ случайное действие,

иначе – действие с максимальной ожидаемой наградой. Такая стратегия называется *$\varepsilon$-жадной*. При этом значение $\varepsilon$ может меняться во время работы алгоритма, что позволяет перейти от исследования среды к применению накопленного опыта.

Листинг 3 – Алгоритм Q-обучения с $\varepsilon$-жадной стратегией исследования среды.

**Require:** $\varepsilon$ — вероятность выбора случайного действия; $\alpha$ — скорость обучения; $\gamma$ — дисконтный фактор.

Инициализировать $Q(s, a)$ для всех $s \in S$, $a \in A$
**while** (не достигнуто условие останова) **do**
    Получить состояние среды $s$
    $p \leftarrow$ случайное вещественное число $\in [0, 1]$
    **if** ($p \leqslant \varepsilon$) **then**
        $a \leftarrow \arg\max_a Q(s, a)$
    **else**
        $a \leftarrow$ случайное действие $\in A$
    **end if**
    Применить действие $a$ к среде
    Получить от среды награду $r$ и состояние $s'$
    $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
**end while**

### 1.2.3. Настройка параметров ЭА как задача для обучения с подкреплением

Рассмотрим задачу выбора параметров эволюционного алгоритма, как задачу, решаемую при помощи обучения с подкреплением [11]. В качестве среды выступает эволюционный алгоритм. Агент совершает действие – выбор значений настраиваемых параметров, таких как вероятность мутации или кроссовера. Затем среда генерирует следующее поколение особей, используя выбранные значения параметров ЭА, и переходит в новое состояние. Награда, возвращаемая агенту средой, является некоторой функцией от значений оптимизируемой функции – функции приспособленности, вычисленных на особях текущего и предыдущего поколений.

Значения параметров ЭА лежат в заданном интервале значений. Таким образом, чтобы установить значение параметра ЭА, агент должен выбрать некоторое значение из заданного интервала. Обычно эту задачу дискретизируют, разделяя диапазон допустимых значений параметра на подынтервалы.

Каждый из подынтервалов соответствует действию агента. Совершив действие – выбор подынтервала, агент в качестве значения параметра устанавливает случайное значение из выбранного подынтервала. Разбиение на интервалы можно делать априорно, то есть разбиение диапазона значений происходит до запуска алгоритма и не меняется в процессе его работы. Однако для некоторых методов адаптивной настройки параметров было показано, что изменение разбиения во время работы способствует улучшению работы алгоритма [5, 12]. В частности, значение параметра можно подобрать тем точнее, чем меньше шаг разбиения. В то же время это усложняет задачу выбора оптимального подынтервала. Существующие методы настройки параметров ЭА с помощью обучения с подкреплением используют априорное разбиение.

### 1.2.4. Метод, предложенный Karafotias et al.

На конференции *GECCO 2014* был предложен метод [6] подбора параметров ЭА на основе обучения с подкреплением, в котором множество состояний среды формируется во время работы алгоритма, а множество действий задается до запуска алгоритма.

В качестве алгоритма обучения с подкреплением используется $Q$-обучение с $\varepsilon$-жадной стратегией исследования среды. Предположим, что число настраиваемых параметров равно $k$. Диапазон допустимых значений настраиваемого параметра $v_i$ делится на $m_i$ интервалов до начала работы алгоритма. Действием является выбор интервалов, из которых будет случайно выбрано значение параметра, для всех настраиваемых параметров и обозначается как $b_1, b_2, \ldots, b_k$, где $b_i, 0 < b_i <= m_i$ – номер интервала для параметра $v_i$. Таким образом число допустимых действий агента в каждом состоянии равно $\prod_{i=1}^{k} m_i$. Отметим, что действием агента осуществляется одновременный выбор значений для всех параметров.

Функции награды для ЭА, максимизирующего функцию приспособленности, задается формулой (2), где $f_t$ – лучшее значение функции приспособленности, полученное на *t*-ой итерации.

$$R = c(\frac{f_{t+1}}{f_t} - 1) \tag{2}$$

Для ЭА, которые не ухудшают лучшее известное решение, функция награды неотрицательна. Стоит отметить, что значение функции приспособленно-

сти зачастую остается неизменным несколько итераций подряд. В самом деле, чтобы награда была положительная, необходимо улучшить лучшую известное значение функции приспособленности. Чтобы замедлить скорость обучения при нулевой награде менялся коэффициент скорости обучения $\alpha$. А именно:

$$Q(s,a) = Q(s,a) + \alpha(r)(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

$$\alpha(r) = \begin{cases} \alpha, & \text{при } r > 0 \\ \alpha_0, & \text{иначе} \end{cases}$$

Стоит отметить, что $\alpha_0 \ll \alpha$.

Для выделения состояний среды используются следующие наблюдаемые характеристики ЭА:

- генетическое разнообразие;
- разнообразие значений функции приспособленности (среднеквадратичное отклонение);
- стагнация (число итераций без улучшения текущего значения функции приспособленности);
- прирост среднего в поколении значения функции приспособленности.

Множество состояний среды строится при помощи *UTree*, описанного далее.

#### 1.2.4.1. Алгоритм UTree

Эффективность применения алгоритмов обучения с подкреплением экспоненциально уменьшается с увеличением числа возможных состояний среды. Однако в большинстве задач не все из них являются существенными. Одним из подходов, позволяющих уменьшить размерность множества состояний среды, является объединение нескольких несущественных состояний. Таким образом, множество состояний может быть разбито на несколько значимых состояний.

Одним из алгоритмов объединения состояний является алгоритм *UTree* [13]. В алгоритме *UTree* строится дерево, листьями которого являются полученные в результате объединения состояния. Также существует вариант алгоритма *UTree*, применимый в случае непрерывного множества состояний. Отличие от алгоритма *UTree* для дискретного случая заключается в том, что он не требует изначального выделения состояний среды. Состояния среды выделяются автоматически в ходе работы алгоритма *UTree*.

Схема работы алгоритма представлена на листинге 4. Состояния среды выделяются на основании наблюдаемых параметров среды. По своей структуре алгоритм *UTree* представляет собой дерево решений в узлах которого стоят условия на параметры среды. Каждому листу соответствует состояние s среды алгоритма обучения с подкреплением, ожидаемое значение награды в котором обозначается как $V(s)$, где $V(s) = \max\limits_{a} Q(s,a)$. Изначально в дереве существует лишь один лист, и таким образом у среды есть единственное возможное состояние $s$, для которого $V(s) = 0$. Алгоритм состоит из двух циклично повторяемых этапов: этапа сбора данных и этапа их обработки.

На этапе сбора данных при помощи текущего построенного дерева по параметрам среды $I$ определяется состояние среды алгоритма обучения с подкреплением $s$. Затем агент *жадно* выбирает действие $a$ и сохраняет полученный кортеж $(I, a, I', r)$,где $I$ – исходные параметры среды, $a$ – выбранное действие, $I'$ – параметры среды после применения действия $a$, $r$ – награда, полученная агентом. Затем на основе полученного опыта агент обновляет значение $Q(s,a)$.

На этапе обработки для каждого сохраненного кортежа вычисляется значение $q(I,a) = r + \gamma V(s')$ – значение ожидаемой награды после применения действия $a$ к среде с параметрами $I$, где $s'$ – состояние среды обучения с подкреплением, соответствующее параметрам среды $I'$. Для каждого состояния $s$ при помощи критерия разбиения ищется точка разбиения. Алгоритм поиска точки разбиения представлен на листинге 5. Если такая точка найдена, то состояние $s$ разбивается на два новых состояния. Множество сохраненных кортежей $(I, a, I', r)$ распределяется по новым состояниям. Затем рассчитываются $Q(s_1, a)$, $Q(s_2, a)$, $V(s_1)$ и $V(s_2)$. Отметим, что в методе, предложенном *Karafotias et. al*, значения $Q(s,a)$ копировались в состояния $Q(s_1, a)$, $Q(s_2, a)$, то есть пересчета значений ожидаемой награды не производилось.

Листинг 4 – Алгоритм *UTree*.

**Фаза обучения**

1: По параметрам среды $I$ найти соответствующее состояние $s$, являющиеся листом дерева решений.
2: Выбрать действие (интервал значений параметра): $a = \arg\max_{a'} Q(s, a')$.
3: Применить выбранное действие к среде, получив награду $r$.
4: Сохранить переход $(I, a, I', r)$ в состоянии $s$.
5: Обновить значение $Q(s, a)$ и $V(s) = \max_{a} Q(s, a)$.

**Фаза разбиения**

1: **for** состояния $s$ **do**
2: **for** переход $(I, a, I', r)$ в состоянии $s$ **do**
3: $q(I, a) = r + \gamma V(s')$
4: **end for**
5: С помощью *критерия разбиения* определить параметр среды и его значение, по которому лучше разделить состояние.
6: **if** найдена точка разбиения **then**
7: Создать два новых состояния $s_1$ и $s_2$.
8: Распределить переходы состояния $s$ по состояниям $s_1$ и $s_2$ в соответствии с разбиением.
9: Рассчитать $Q(s_1, a)$ и $Q(s_2, a)$ по сохраненным переходам.
10: Заменить состояние $s$ в дереве решений, на вершину с детьми $s_1$ и $s_2$ и условием выбора, соответствующим точке разбиения.
11: **end if**
12: **end for**

#### 1.2.4.2. Критерий Колмогорова-Смирнова

В статистическом анализе используют различные критерии однородности для проверки гипотезы о принадлежности двух независимых выборок одному закону распределения. Одним из наиболее используемых непараметрических критериев о проверке однородности двух эмпирических законов распределения является критерий однородности Смирнова [14].

Эмпирическая функция распределения является приближением теоретической функции распределения, построенное с помощью выборки из него. Пусть $\{X_i\}_{i=1}^n$ выборка объема $n$ из случайной величины $X$. Эмпирической функцией распределения случайной величины $X$ называется случайная величина $F(x) = \frac{1}{n}\sum_{i=1}^{n} H(x - X_i)$, где $H$ – функция Хевисайда. По сути заданная

Листинг 5 – Алгоритм поиска точки разбиения

**Require:** $s$ – разбиваемое состояние
1: $best \leftarrow +\infty$
2: **for** параметра среды $p_i$ **do**
3: Отсортировать $I$ в состоянии $s$ по значению параметра $p_i$.
4: $gap \leftarrow 0.2|I|$
5: **for** $j \in [gap, |I| - gap]$ **do**
6: $candidate \leftarrow \frac{p_{i,j}+p_{i,j+1}}{2}$
7: $x \leftarrow \{q(I_{p_{i,j}}, a) | p_{i,j} < candidate\}$
8: $y \leftarrow \{q(I_{p_{i,j}}, a) | p_{i,j} > candidate\}$
9: С помощью критерия Колмогорова-Смирнова считается $p$-value для множеств $x$ и $y$.
10: **if** $p$-value$< best$ **then**
11: $best \leftarrow p$-value
12: $result \leftarrow (i, candidate)$
13: **end if**
14: **end for**
15: **if** $best < 0.05$ **then return** $result$
16: **end if**
17: **end for**

таким образом функция распределения в точке $x$ равна частоте элементов выборки, не превосходящих $x$.

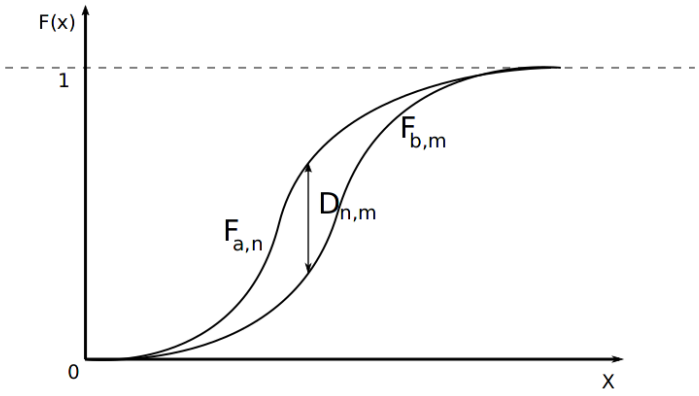



Рисунок 2 – График $F_{a,n}$ и $F_{b,m}$.

Критерий позволяет найти точку, в которой сумма накопленных частот расхождений наибольшая, и оценить достоверность этого расхождения. В каче-

стве нулевой гипотезы $H_0$ принимается, что две исследуемые выборки подчиняются одному закону распределения случайной величины. Для двух независимых выборок $a$ и $b$, объемами $n$ и $m$ соответственно, строятся эмпирические функции распределения $F_{a,n}$ и $F_{b,m}$. Затем считается значение $\sqrt{\frac{nm}{n+m}}D_{n,m}$, где $D_{n,m} = \sup\limits_x |F_{a,n}(x) - F_{b,m}(x)|$. Если рассчитанное значение превышает квантиль распределения Колмогорова $K_\alpha$ для заданного уровня значимости $\alpha$, то нулевая гипотеза $H_0$ отвергается.

## 1.3. Выводы по главе 1

Описаны основные принципы работы алгоритмов обучения с подкреплением, применяющихся в данной работе для настройки параметров ЭА. Приведен обзор существующих методов адаптивной настройки параметров ЭА. Описан способ настройки параметров ЭА как задачи оптимизации, решаемой с помощью алгоритма обучения с подкреплением.

# ГЛАВА 2. ПРЕДЛАГАЕМЫЕ МЕТОДЫ НАСТРОЙКИ ПАРАМЕТРОВ ЭА

## 2.1. Цель работы

Существует метод адаптивной настройки параметров ЭА с помощью обучения с подкреплением, позволяющий адаптивно выделять состояния среды. Множество действий агента определяется разбиением диапазона допустимых значений параметра, которое задается до начала выполнения алгоритма. Также существуют методы настройки параметров ЭА, позволяющие адаптивно разбивать диапазон допустимых значений параметра.

В данной работе предлагается исследовать эффективность адаптивного выделения множества действий агента за счет разбиения диапазона допустимых значений параметра в ходе работы алгоритма. Целью исследований являлась разработка метода адаптивной настройки параметров эволюционного алгоритма с помощью обучения с подкреплением. Предлагаемый алгоритм на основе обучения с подкреплением должен формировать множество действий агента во время работы, адаптивно разбивая диапазон допустимых значений параметра.

## 2.2. Метод на основе алгоритмов EARPC и UTree

Данный метод является объединением методов *EARPC* (1.2.1) и *UTree* (1.2.4.1). В отличие от метода, предложенного Karafotias et al. (1.2.4), выбор значений параметров происходит с помощью алгоритма *EARPC*. В процессе работы по наблюдаемым характеристикам ЭА строится дерево решений *UTree*. Алгоритм *EARPC* выбирает значения параметров на основе сохраненных в листе переходов $(I, a, I', r)$. Для оценки качества выбранного назначения параметров для алгоритма *EARPC* используется награда, получаемая агентом. Таким образом, необходимо по наблюдаемым значениям ЭА найти лист дерева *UTree*. Затем значения параметров выбираются при помощи алгоритма *EARPC*, используя переходы, хранящиеся в найденном листе.

При построении дерева *UTree* необходимо определить способ разбиения листа на два состояния. Для применения критерия разбиения Колмогорова-Смирнова, для каждого перехода $(I, a, I', r)$, сохраненного в листе, вычисляется значение $q(I, a) = r + \gamma V(s')$, где $s'$ – лист, соответствующий характеристикам ЭА $I'$. При этом необходимо посчитать ожидаемую награду $V(s')$. В качестве $V(s')$ предлагается использовать математическое ожидание награды

в листе $s'$. Алгоритм *EARPC* разбивает диапазон значений параметра на два подынтервала, один из которых выбирается с вероятностью пропорциональной средней награде на подынтервале. Таким образом $V(s')$ вычисляется по формуле $V(s') = \sum_{i=1}^{2} \frac{Q_i^2}{Q_1+Q_2}$, где $Q_1$ и $Q_2$ средние значения награды на первом и втором подынтервале соответственно.

Кроме того, после выделения новых состояний среды необходимо пересчитать значения ожидаемой награды для полученных состояний. В методе, предложенном *Karafotias et al.*, значения $Q(s, a)$, где $s$ – разбиваемое состояние, копировались в новые состояния, то есть значения ожидаемой награды для полученных состояний не пересчитывались. В предлагаемом подходе при выделении нового состояния в соответствии с алгоритмом *UTree* множество переходов перераспределяется между полученными состояниями. Таким образом значения ожидаемой награды пересчитываются автоматически при следующей итерации алгоритма *EARPC*.

### 2.3. Метод с адаптивным выделением множества действий

Также в данной работе предлагается метод адаптивной настройки параметров ЭА с помощью $Q$-обучения с адаптивным выделением множества действий. В данном подходе действие определяется аналогично методу, предложенному *Karafotias et al*. Однако разбиение диапазона допустимых значений параметров меняется в ходе работы алгоритма.

Агент выбирает действие на основе алгоритма $Q$-обучения с $\varepsilon$-жадной стратегией исследования среды. В случае когда значения ожидаемой награды примерно одинаковы для всех воможных действий, агент не может выбрать какое из действий наиболее эффективно. Поэтому в данном случае текущее разбиение диапазонов значения параметров пересчитывается. При этом в следствие изменения разбиения меняется множество допустимых действий агента.

В процессе работы алгоритма сохраняются выбранные назначения параметров и полученные за эти назначения награды. Сохраненные данные используются при переразбиении диапазона значений для каждого из настраиваемых параметров. Опишем процедуру переразбиения диапазона. Сначала диапазон делится на два подынтервала при помощи критерия Колмогорова-Смирнова. На каждой следующей итерации разбиения диапазона, полученные на текущей итерации подынтервалы при помощи критерия Колмогорова-Смирнова разбиваются на два подынтервала. В случае, если точка разбиения подынтервала не

найдена, разбиение интервала не происходит. Таким образом, максимальное число подынтервалов на которые разбивается диапазон допустимых значений параметра равен $2^i$, где $i$ – число итераций разбиения диапазона. На листинге 6 представлен алгоритм разбиения диапазона для $i = 2$.

Листинг 6 – Алгоритм разбиения диапазона с двумя итерациями в методе с адаптивным выделением множества действий.

**Require:** $V$ — множество назначений параметров;
1: **for** параметр *v* **do**
2: Разбиение $P \leftarrow \emptyset$
3: Отсортировать множество назначений по параметру *v*
4: С помощью критерия Колмогорова-Смирнова найти точку разбиения *s* множества *V*
5: **if** Точка разбиения *s* не найдена **then**
6: $P \leftarrow \{[v_{min}, v_{max}]\}$
7: **else**
8: Разбить множество *V* на *L* и *R* в соответствии с *s*
9: Найти точку разбиения $s_l$ для множества *L*
10: Найти точку разбиения $s_r$ для множества *R*
11: **if** Точки разбиения $s_l$ и $s_r$ не найдены **then**
12: $P \leftarrow \{[v_{min}, s], (s, v_{max}]\}$
13: **else if** Точка разбиения $s_l$ не найдена **then**
14: $P \leftarrow \{[v_{min}, s], (s, s_r], (s_r, v_{max}]\}$
15: **else if** Точка разбиения $s_r$ не найдена **then**
16: $P \leftarrow \{[v_{min}, s_l], (s_l, s], (s, v_{max}]\}$
17: **else**
18: $P \leftarrow \{[v_{min}, s_l], (s_l, s], (s, s_r], (s_r, v_{max}]\}$
19: **end if**
20: **end if**
21: **end for**

## 2.4. Выводы по главе 2

Предложено два метода адаптивной настройки параметров эволюционных алгоритмов с помощью обучения с подкреплением. Оба метода формируют можество действий агента в ходе работы, адаптивно разбивая диапазон допустимых значений настраеваемого параметра. Предложенные подходы являются достаточно общими и не накладывают ограничения на настраиваемый эволюционный алгоритм.

# ГЛАВА 3. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

В данной главе приводятся результаты тестирования разработанных методов адаптивного выбора параметров ЭА на модельных задачах. Приводятся результаты сравнения существующих методов с существующими методами настройки параметров ЭА с помощью обучения с подкреплением. Также исследуется эффективность существующих методов выделения состояний среды.

## 3.1. Описание экспериментов

В качестве модельных задач используются известные задачи числовой оптимизации, а именно нахождение глобального минимума некоторой многомерной функции в ограниченной области с заданной точностью $\epsilon$. Пусть $F(x_1, \ldots, x_n)$ – оптимизируемая функция. При этом $x_i \in [min_i, max_i]$ для $1 \leqslant i \leqslant n$. В качестве функции приспособленности используется функция $F$. Поэтом более приспособленной считается особь с меньшим значением функции приспособленности.

В качестве особи ЭА, решающего заданную задачу, рассматривался набор из $n$ вещественных чисел. Пусть $x_1, \ldots, x_n$ – текущее решение задачи оптимизации. В качестве эволюционных операторов применялся только оператор мутации, определяемый следующим образом:

$$x_i = \begin{cases} max_i, & \text{при } x_i + \sigma dx_i > max_i \\ min_i, & \text{при } x_i + \sigma dx_i < min_i \\ x_i + \sigma dx_i, & \text{иначе} \end{cases}, \text{для } 1 \leqslant i \leqslant n \tag{3}$$

где $dx_i \sim \mathcal{N}(0, 1)$, а $\sigma$ – настраиваемый параметр, называемый *шагом*. Ожидаемым поведением методов адаптивного выбора параметров ЭА является уменьшение значения шага мутации при приближении к глобальному минимуму оптимизируемой функции. В проводимых экспериментах допустимые значения параметра $\sigma$ задаются интервалом $[0, k]$, где $k$ – некоторая константа. В рамках данной работы проводились эксперименты с различными значениями параметра $k$. При увеличении параметра $k$ увеличивается диапазон допустимых значений параметра $\sigma$, что усложняет задачу поиска его оптимального значения.

В качестве ЭА, решающего поставленную задачу, используется $(\mu + \lambda)$ эволюционная стратегия. При увеличении $\lambda$ увеличивается число использова-

ний каждого выбранного значения параметра. Это способствует более точной оценке эффективности выбранного значения, однако сказывается на производительности. Кроме того, при тестировании предложенных и существующих методов настройки параметров ЭА нельзя ограничиться рассмотрением только $(1 + \lambda)$ эволюционной стратегии. В самом деле, если в поколении всего одна особь, то из предложенных для построения дерева *UTree* характеристик ЭА остается только число итераций без улучшения текущего решения. Таким образом, в рамках данной работы рассматривались различные значения параметра $\mu$, а в качестве характеристик ЭА использовались среднеквадратичное отклонение значений функции приспособленности в поколении, уменьшение среднего значения функции приспособленности и число итераций без улучшения текущего решения.

Пусть $f_t$ – минимальное значение функции приспособленности на $t$-ом поколении ЭА. Награда вычисляется по формуле $R = c(\frac{f_t}{f_{t+1}} - 1)$. При решении задачи минимизации при помощи $(\mu + \lambda)$ эволюционной стратегии, $f_{t+1} \leqslant f_t$. Поэтому агент всегда получает неотрицательную награду.

### 3.1.1. Значения параметров

В рамках проводимых экспериментов большинство параметров были взяты из статьи karafotias. В алгоритме $Q$-обучения были использованы следующие значения параметров:

- $\alpha_0 = 0.2$ – коэффициент скорости обучения при нулевой награде;
- $\alpha = 0.9$ – коэффициент скорости обучения при положительной награде;
- $\gamma = 0.8$ – дисконтный фактор;
- $\varepsilon = 0.1$ – вероятность выбора случайного действия;
- $c = 100$ – коэффициент масштабирования награды.

Эксперименты проводились для следующих значений параметров ЭА:

- $\mu \in \{1, 5, 10\}$ – число особей в поколении;
- $\lambda \in \{1, 3, 7\}$ – число потомков, создаваемых для каждой особи в процессе формирования следующего поколения;
- $\sigma \in \{1, 2, 3\}$ – верхняя граница допустимых значений $\sigma$;
- $\epsilon = 10^{-5}$ – допустимое отклонение решения от минимального значения оптимизируемой функции.

### 3.1.2. Описание результатов

Введем следующие обозначения для исследуемых методов настройки параметров ЭА:

- *karafotias* – метод, предложенный *Karafotias et al*. (1.2.4);
- *q-learn* – метод настройки параметров ЭА с помощью $Q$-обучения с одним состоянием среды и множеством действий, заданным до начала работы алгоритма;
- *earpc* – метод *EARPC* (1.2.1);
- *uearpc* – метод на основе *EARPC* и *UTree* (2.2);
- *adaptive* – метод с адаптивным выделением множества действий (2.3).

Для каждого метода проводилось 50 запусков ЭА для каждой комбинации рассматриваемых значений параметров $k$, $\mu$, $\lambda$. Для каждой задачи было составлено три таблицы, в которых представлены усредненные по 50 запускам значения числа поколений, потребовавшихся для решения задачи, при различных значениях параметров ЭА. В первой из них находятся результаты работы метода adaptive. Остальные таблицы содержат сравнение пар методов *karafotias* и *q-learn*, *earpc* и *uearpc* при разных значениях $\mu$, $\sigma$ и $k$. Зеленым цветом выделяется лучшее значение среди всех исследуемых методов при заданных значениях параметров ЭА.

Далее подробно описываются модельные задачи, на которых проводилось экспериментальное исследование методов настройки параметров ЭА. Для каждой задачи приводятся результаты ее решения.

## 3.2. Сфера

Необходимо с точностью $\epsilon$ найти минимум унимодальной сферической функции (4).

$$f(x_1, .., x_n) = \sum_{i=1}^{n} x_i^2 \tag{4}$$

График функции для двух переменных представлен на рис. 3. При $x_i \in [-10; 15]$ глобальный минимум достигается в точке $(0, .., 0)$.

Результаты применения исследуемых методов к данной задаче приведены в таблицах 1, 2 и 3. Результаты подтверждают интуитивные предположения: для всех рассмотренных методов при увеличении параметров $\mu$ или $\lambda$, уменьшается среднее число поколений, потребовавшихся для нахождения минимума сферической функции, а при увеличении $k$ оно увеличивается. В

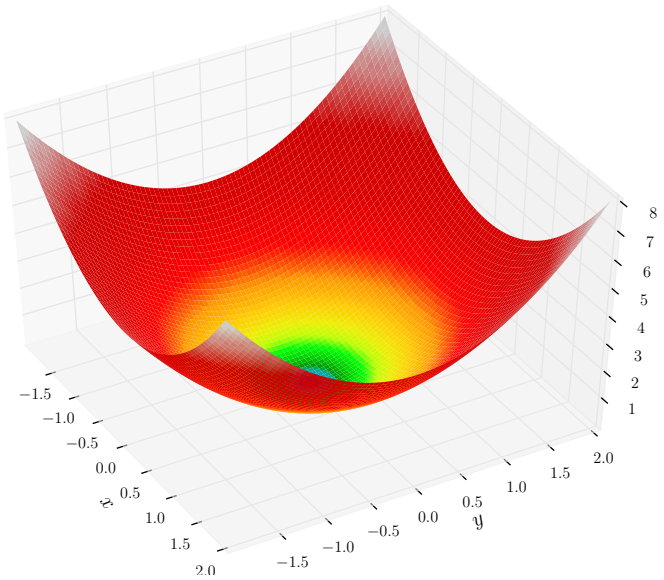



Рисунок 3 – График сферической функции для двух переменных.

самом деле, при увеличении параметров $\mu$ или $\lambda$ увеличивается число особей, рассматриваемых при формировании следующего поколения. Увеличение параметра $k$ увеличивает диапазон допустимых значений $\sigma$, что затрудняет выбор оптимального значения.

Предложенный метод *adaptive* превосходит остальные рассматриваемые методы на большинстве конфигураций ЭА. Отметим, что при значении параметра $\mu$ равном единице, то есть в случае когда поколение состоит из одной особи, число итераций ЭА, понадобившихся для решения задачи, с помощью метода *adaptive* оказалось в разы меньше, чем при использовании остальных методов. Это можно объяснить тем, что в методе предложенном *Karafotias et al*. награда, полученная за использование того или иного значения параметра, перестает учитываться со временем работы алгоритма. В методе *adaptive* отсутствует данный недостаток, так как полученный опыт сохраняется не только в значениях ожидаемой награды, но и в разбиении диапазона значений оптимизируемого параметра. Это происходит в случае продолжительного выбора неэффективного значения параметра. В методе *uearpc* разбиение диапазона значений настраеваемого параметра происходит на основе кластеризации значений и при этом не учитываются значения награды. Данные предположения подтверждаются графиками распределения выбранных значений параметра $\sigma$ в предложенных методах, приведенными на рис. 4, 5 и 6. На графиках представлены выбираемые значения параметра $\sigma$ для каждого поколения ЭА. Так-

| k = 1 | | | |
|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 2434 | 2207 | 878 |
| 5 | 1450 | 569 | 368 |
| 10 | 703 | 331 | 186 |
| k = 2 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 4342 | 2333 | 1464 |
| 5 | 1891 | 974 | 616 |
| 10 | 1164 | 459 | 188 |
| k = 3 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 8055 | 2826 | 1427 |
| 5 | 2447 | 1445 | 896 |
| 10 | 1996 | 1053 | 409 |

Таблица 1 – Таблица с результатами применения метода adaptive для сферической функции.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 8769 | 8048 | 4221 | 2683 | 1085 | 2620 |
| 5 | 1664 | 2076 | 706 | 824 | 390 | 473 |
| 10 | 959 | 747 | 378 | 358 | 195 | 167 |
| k = 2 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 25192 | 28523 | 7681 | 6478 | 3360 | 3739 |
| 5 | 3814 | 3468 | 994 | 1163 | 716 | 756 |
| 10 | 2397 | 1833 | 445 | 465 | 320 | 252 |
| k = 3 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 36698 | 29112 | 7845 | 9115 | 3907 | 4813 |
| 5 | 8328 | 5886 | 2790 | 2222 | 919 | 777 |
| 10 | 2398 | 2531 | 1074 | 1206 | 391 | 427 |

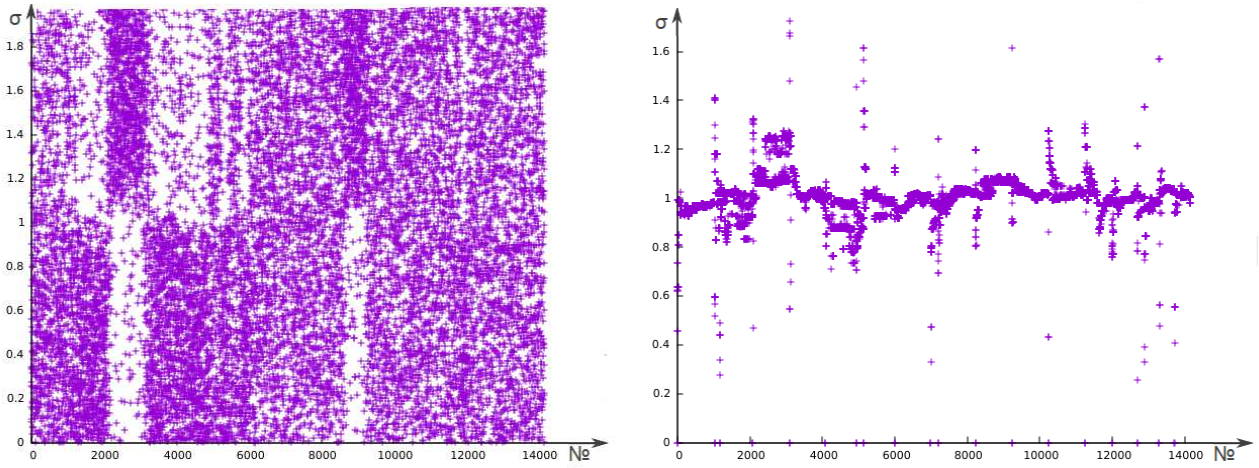
Таблица 2 – Таблица с результатами применения методов q-learn и karafotias для сферической функции.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 5258 | 4830 | 3942 | 3070 | 1653 | 2247 |
| 5 | 4472 | 3893 | 1589 | 845 | 642 | 568 |
| 10 | 1438 | 728 | 534 | 400 | 142 | 422 |
| k = 2 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 31738 | 16182 | 5152 | 4233 | 1688 | 2956 |
| 5 | 4908 | 5173 | 1631 | 946 | 1511 | 626 |
| 10 | 778 | 876 | 719 | 788 | 380 | 413 |
| k = 3 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 54868 | 30710 | 12446 | 11985 | 10118 | 9207 |
| 5 | 3348 | 12313 | 1379 | 1848 | 1424 | 1105 |
| 10 | 3173 | 3745 | 1114 | 1712 | 410 | 537 |

Таблица 3 – Таблица с результатами применения методов earpc и uearpc для сферической функции.

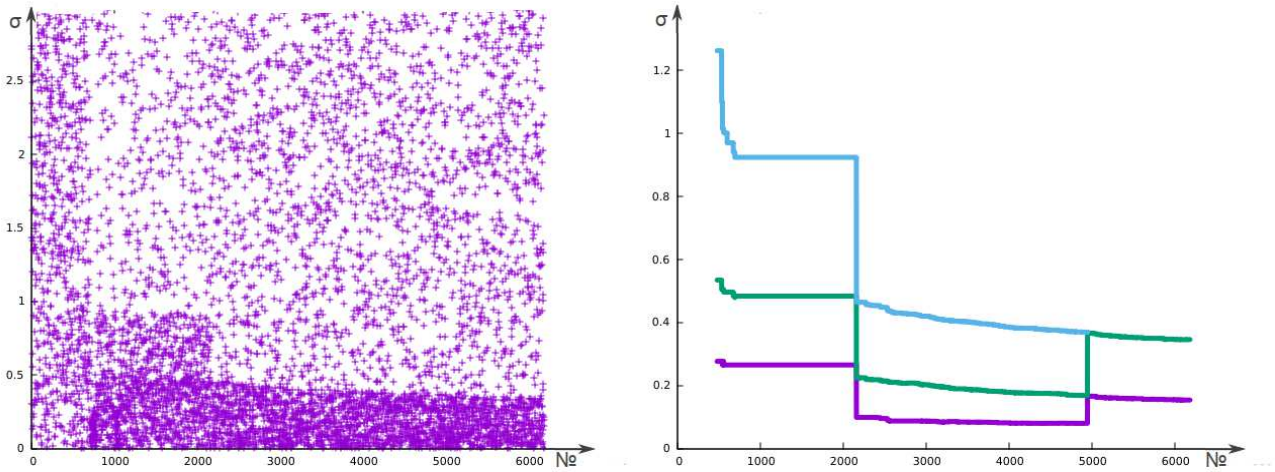
же на графиках для методов $uarpc$ и $adaptive$ представлены значения точек разбиения в зависимости от номера поколения ЭА.

На графике 4 видно, что в конце работы алгоритма *uearpc* значения настраеваемого параметра $\sigma$ выбираются случайно из всего диапазона допустимых значений. При этом диапазон разбивается на два интервала почти равного размера. На графике 6 видно, что при получении положительной награды за подбор эффективного значения параметра алгоритм *karafotias* продолжает выбирать значения параметра из того же интервала. Но если на протяжении нескольких итераций не была получена положительная награда, алгоритм теряет информацию о ранее выбранном эффективном значении. На графике 5 видно, что на протяжении всего процесса оптимизации алгоритм *adaptive* приближает значение настраеваемого параметра к оптимальному. При этом в начале работы алгоритма в течение некоторого числа итераций значения выбираются случайным образом, так как полученного опыта недостаточно для разбиения диапазона значений. В ходе работы пересчитываются размеры интервалов разбиения. А в конце работы алгоритм уменьшает необходимое число интервалов.




(a) График выбранных значений $\sigma$. (b) График значения точки разбиения.

Рисунок 4 – Графики работы метода *uarpc* для сферической функции.




(a) График выбранных значений $\sigma$. (b) График значения точек разбиения.

Рисунок 5 – Графики работы метода *adaptive* для сферической функции.

Отметим, что при применении методов *karafotias* и *q-learn*, *earpc* и *uearpc* были получены схожие результаты. Таким образом, выделение состояний среды слабо влияет на эффективность настройки значений параметров.

### 3.3. Функция Розенброка

Предложенная Ховардом Розенброком невыпуклая функция [15], задаваемая формулой (5), часто используется для оценки производительности алгоритмов оптимизации. Обычно в качестве значений $a$ и $b$ выбираются $a = 1$, $b = 100$. График функции для двух переменных представлен на рис. 7. Считается, что поиск глобального минимума данной функции при помощи алгоритмов оптимизации является нетривиальной задачей. Это объясняется тем,

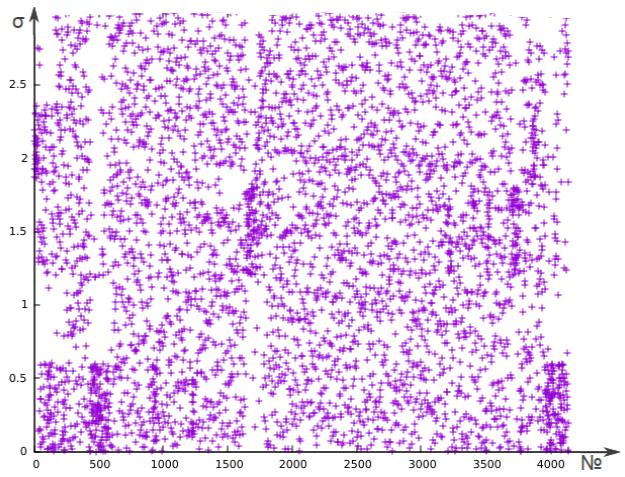



Рисунок 6 – График выбранных значений $\sigma$ в методе *karafotias* для сферической функции.

что глобальный минимум находится в длинной, узкой *впадине*, найти которую обычно не составляет труда. Трудность заключается в поиске минимума в этой впадине. При $x_i \in [-15; 10]$ он достигается в точке $(1, .., 1)$.

$$f(x_1, x_2) = (a - x_1^2)^2 + b(x_2 - x_1^2)^2 \tag{5}$$

Результаты применения исследуемых методов к данной задаче приведены в таблицах 4, 5 и 6. На основе посчитанных результатов можно сделать выводы аналогичные сделанным на основе задачи оптимизации сферической функции (3.2). Выводы подтверждаются графиками распределения выбранных значений параметра $\sigma$ в предложенных методах, приведенными на рис. 8, 9 и 10.

### 3.4. Функция Леви

Мультимодальная функция Леви для двух переменных задается формулой (6). График функции представлен на рис. 11. При $x, y \in [-10; 10]$ минимум достигается в точке $(1, 1)$.

$$\begin{aligned} f(x_1, x2) =& sin^2(3\pi x) + (x-1)^2(1 + sin^2(3\pi y)) + \\ &+ (y-1)^2(1 + sin^2(2\pi y)) \end{aligned} \tag{6}$$

Результаты применения исследуемых методов к данной задаче приведены в таблицах 7, 8 и 9. На основе посчитанных результатов можно сделать выводы аналогичные сделанным на основе задачи оптимизации сферической

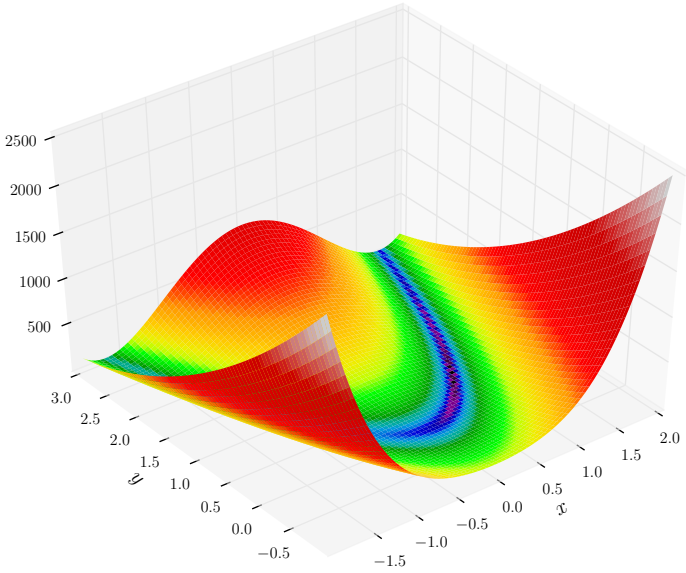



Рисунок 7 – График функции Розенброка для двух переменных.

| k = 1 | | | |
|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 3631 | 1776 | 1226 |
| 5 | 1666 | 918 | 502 |
| 10 | 1008 | 622 | 358 |
| k = 2 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 4744 | 1839 | 1160 |
| 5 | 2467 | 1128 | 967 |
| 10 | 1722 | 988 | 615 |
| k = 3 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 5159 | 2821 | 1517 |
| 5 | 2704 | 1544 | 902 |
| 10 | 2048 | 1296 | 955 |

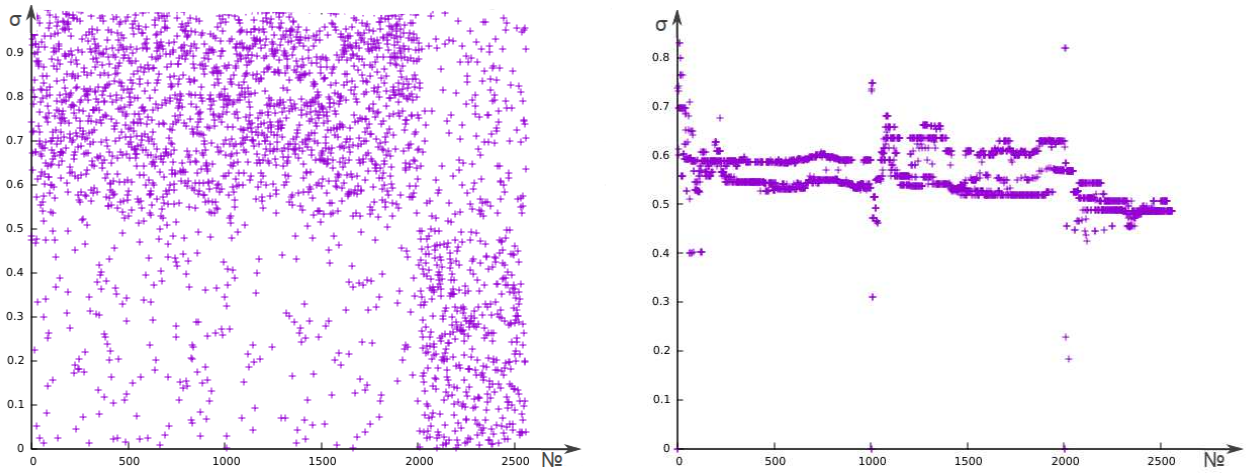
Таблица 4 – Таблица с результатами применения метода adaptive для функции Розенброка.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 9653 | 8385 | 2226 | 2069 | 1605 | 1757 |
| 5 | 1706 | 2281 | 894 | 942 | 570 | 675 |
| 10 | 1103 | 1105 | 604 | 665 | 489 | 473 |
| k = 2 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 23663 | 21043 | 6405 | 6825 | 2388 | 2183 |
| 5 | 3944 | 4029 | 1614 | 1780 | 1012 | 1461 |
| 10 | 2108 | 2255 | 1420 | 1116 | 856 | 712 |
| k = 3 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 29082 | 23305 | 7977 | 10726 | 4680 | 4266 |
| 5 | 6208 | 5419 | 2514 | 2096 | 1120 | 1629 |
| 10 | 3747 | 3258 | 1740 | 1523 | 995 | 1203 |

Таблица 5 – Таблица с результатами применения методов q-learn и karafotias для функции Розенброка.

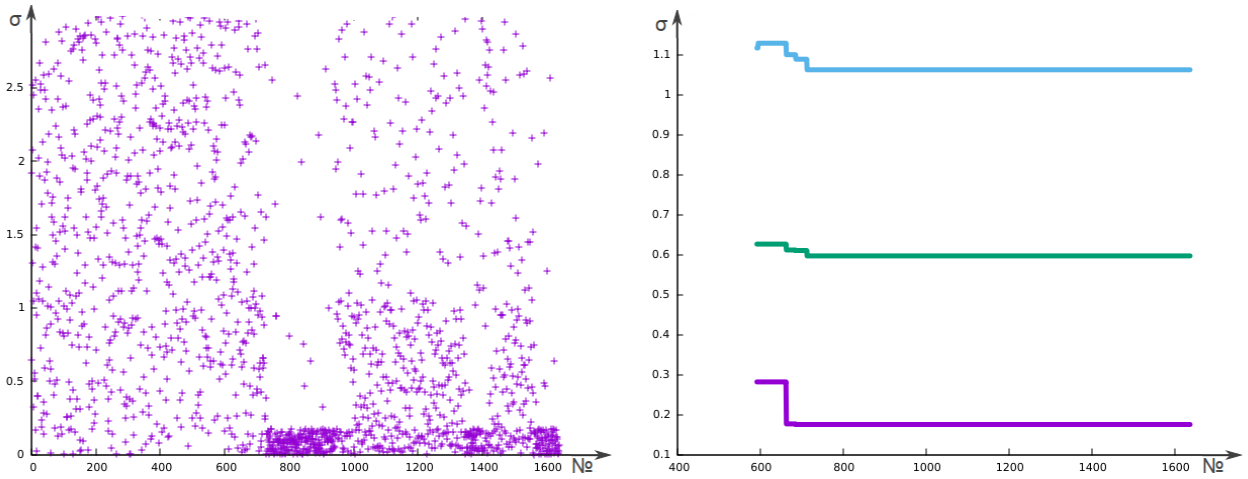
| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 7794 | 8689 | 3148 | 3610 | 1422 | 1422 |
| 5 | 4341 | 4530 | 1958 | 2197 | 1679 | 1329 |
| 10 | 2681 | 2926 | 1460 | 1485 | 1103 | 1858 |
| k = 2 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 27865 | 19142 | 9748 | 8997 | 3806 | 4085 |
| 5 | 5961 | 5750 | 2293 | 1720 | 933 | 1180 |
| 10 | 2411 | 2807 | 1486 | 1234 | 627 | 922 |
| k = 3 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 24249 | 27327 | 6541 | 16040 | 4144 | 5408 |
| 5 | 5953 | 7665 | 2823 | 2304 | 1033 | 1055 |
| 10 | 5095 | 3873 | 1013 | 1028 | 779 | 839 |

Таблица 6 – Таблица с результатами применения методов earpc и uearpc для функции Розенброка.




(a) График выбранных значений $\sigma$. (b) График значения точки разбиения.

Рисунок 8 – Графики работы метода *uarpc* для функции Розенброка.




(a) График выбранных значений $\sigma$. (b) График значения точек разбиения.

Рисунок 9 – Графики работы метода *adaptive* для функции Розенброка.

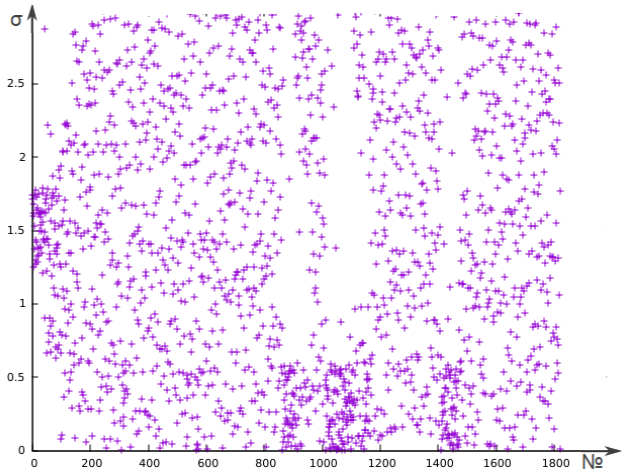



Рисунок 10 – График выбранных значений $\sigma$ в методе *karafotias* для функции Розенброка.

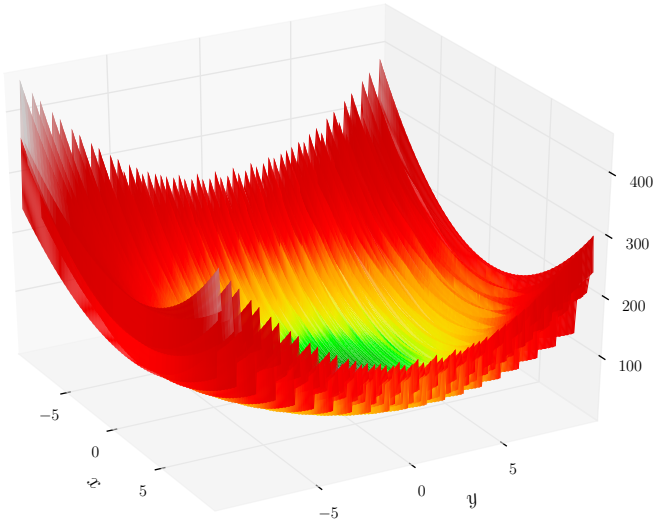
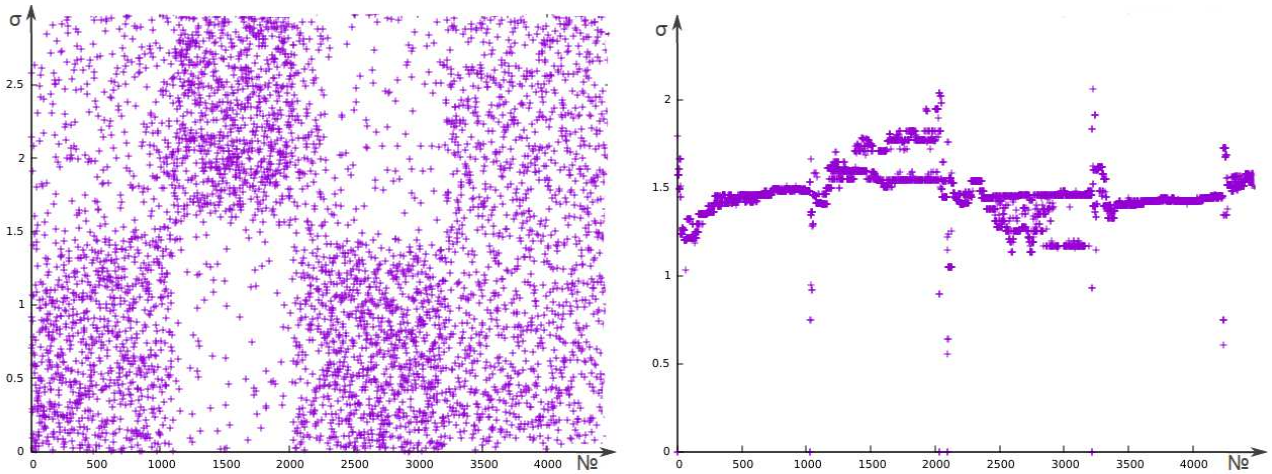



Рисунок 11 – График функции Леви для двух переменных.




(a) График выбранных значений $\sigma$. (b) График значения точки разбиения.

Рисунок 12 – Графики работы метода *uarpc* для функции Леви.

функции (3.2). Выводы подтверждаются графиками распределения выбранных значений параметра $\sigma$ в предложенных методах, приведенными на рис. 12, 13 и 14.

### 3.5. Функция Растригина

Леонард Растригин предложил [16] нелинейную мультимодальную функцию для тестирования эффективности алгоритмов оптимизации. Нахождение минимума этой функции затруднено большим количеством локальных минимумов в области поиска. Функция задается формулой (7). При $x_i \in [-5; 5]$ минимум достигается в точке $(0, 0)$.

| k = 1 | | | |
|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 3496 | 1980 | 1321 |
| 5 | 1778 | 855 | 356 |
| 10 | 884 | 533 | 308 |
| k = 2 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 4947 | 2020 | 1205 |
| 5 | 1935 | 1085 | 770 |
| 10 | 1803 | 887 | 618 |
| k = 3 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 5216 | 2900 | 1700 |
| 5 | 3105 | 1808 | 1017 |
| 10 | 2071 | 1330 | 667 |

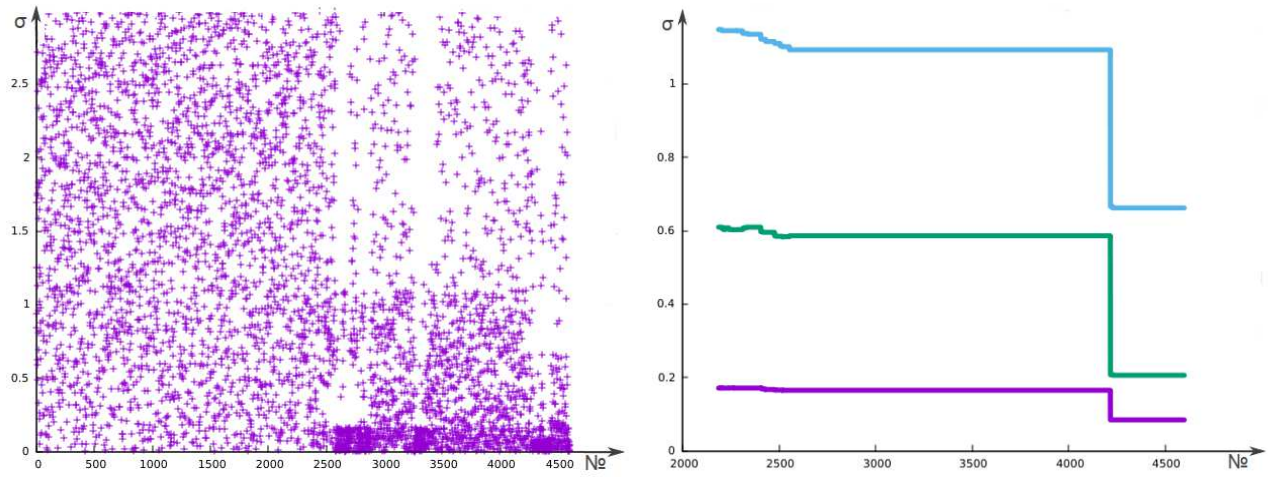
Таблица 7 – Таблица с результатами применения метода adaptive для функции Леви.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 7200 | 7265 | 3305 | 2903 | 1584 | 1600 |
| 5 | 1898 | 1865 | 862 | 794 | 606 | 444 |
| 10 | 840 | 804 | 502 | 624 | 331 | 294 |
| k = 2 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 13420 | 14789 | 6884 | 6601 | 3521 | 2370 |
| 5 | 3517 | 3369 | 1231 | 1326 | 792 | 1089 |
| 10 | 1884 | 2197 | 988 | 1006 | 731 | 564 |
| k = 3 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 21470 | 21953 | 9029 | 7071 | 4139 | 4019 |
| 5 | 6966 | 6742 | 2467 | 2664 | 1929 | 1788 |
| 10 | 2901 | 2943 | 1462 | 1832 | 1117 | 799 |

Таблица 8 – Таблица с результатами применения методов q-learn и karafotias для функции Леви.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 7986 | 14092 | 2789 | 3688 | 1923 | 3820 |
| 5 | 2372 | 2246 | 1632 | 2171 | 1426 | 1236 |
| 10 | 2353 | 1451 | 1491 | 1134 | 510 | 766 |
| k = 2 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 25721 | 30653 | 7477 | 4612 | 2533 | 2967 |
| 5 | 7222 | 5365 | 3039 | 2943 | 1153 | 2180 |
| 10 | 5139 | 3072 | 1045 | 2381 | 806 | 1064 |
| k = 3 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 28900 | 35119 | 6966 | 10883 | 3375 | 2062 |
| 5 | 10480 | 9059 | 4593 | 3714 | 594 | 1439 |
| 10 | 5210 | 4814 | 3140 | 2389 | 521 | 845 |

Таблица 9 – Таблица с результатами применения методов earpc и uearpc для функции Леви.




(a) График выбранных значений $\sigma$. (b) График значения точек разбиения.

Рисунок 13 – Графики работы метода *adaptive* для функции Леви.

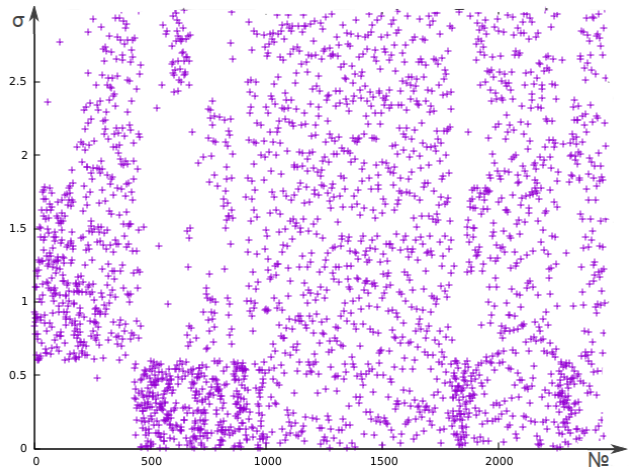



Рисунок 14 – График выбранных значений $\sigma$ в методе *karafotias* для функции Леви.

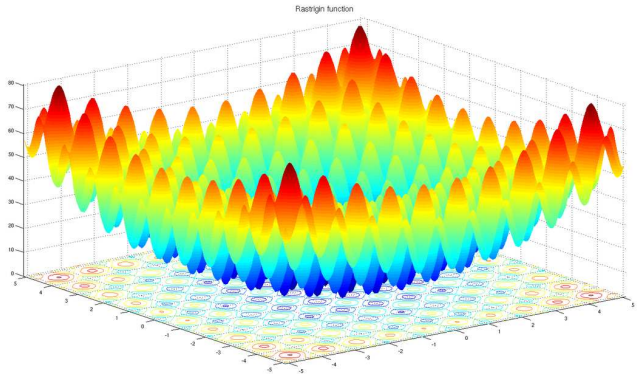



Рисунок 15 – График функции Растригина для двух переменных.

$$f(x_1, .., x_n) = An + \sum_{i=1}^{n} \left[x_i^2 - Acos\left(2\pi x_i\right)\right] \tag{7}$$

Результаты применения исследуемых методов к данной задаче приведены в таблицах 10, 11 и 12. На основе посчитанных результатов можно сделать выводы аналогичные сделанным на основе задачи оптимизации сферической функции (3.2). Выводы подтверждаются графиками распределения выбранных значений параметра $\sigma$ в предложенных методах, приведенными на рис. 16, 17 и 18.

### 3.6. Выводы по главе 3

Методы адаптивной настройки параметров ЭА, предложенные в главе 2, были протестированы на четырех модельных задачах. Было проведено сравнение предложенных методов с существующими алгоритмами настройки пара-

| k = 1 | | | |
|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 5124 | 2301 | 1411 |
| 5 | 1859 | 1053 | 401 |
| 10 | 1617 | 683 | 483 |
| k = 2 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 5165 | 3461 | 1753 |
| 5 | 1990 | 1396 | 934 |
| 10 | 2022 | 1181 | 707 |
| k = 3 | | | |
| $\mu$ \ $\lambda$ | 1 | 3 | 7 |
| 1 | 6535 | 3391 | 2573 |
| 5 | 3148 | 1721 | 1231 |
| 10 | 2352 | 1677 | 1101 |

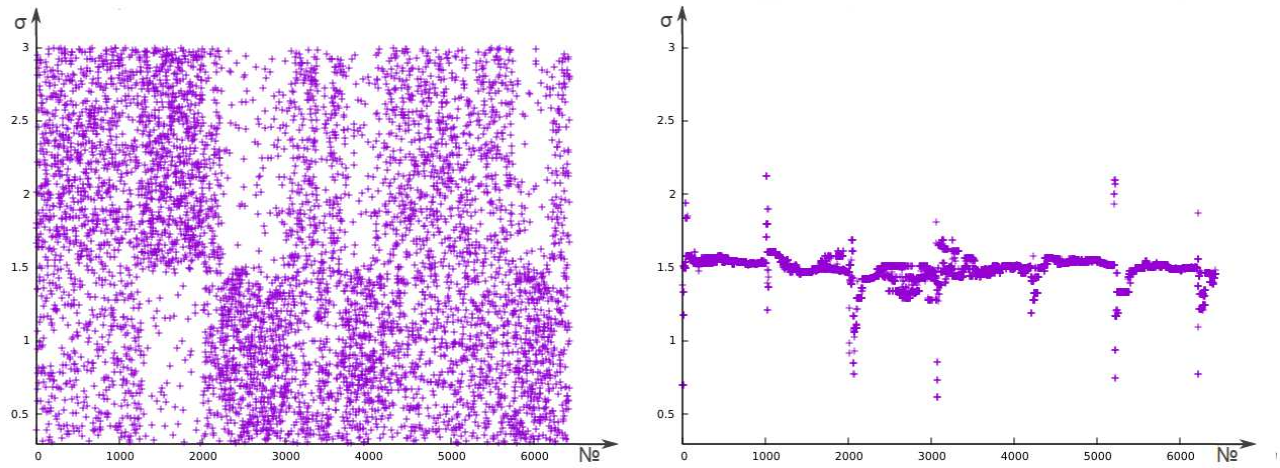
Таблица 10 – Таблица с результатами применения метода adaptive для функции Растригина.

| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 15058 | 13418 | 3914 | 4167 | 1791 | 2330 |
| 5 | 2311 | 2809 | 869 | 748 | 533 | 665 |
| 10 | 1497 | 1593 | 488 | 616 | 290 | 235 |
| k = 2 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 23371 | 27239 | 7295 | 7169 | 3488 | 2968 |
| 5 | 8076 | 5810 | 2116 | 2705 | 1001 | 1247 |
| 10 | 3415 | 2787 | 1037 | 1106 | 811 | 666 |
| k = 3 | | | | | | |
| $\mu$ \ $\lambda$ | 1 | | 3 | | 7 | |
| | q-learn | karafotias | q-learn | karafotias | q-learn | karafotias |
| 1 | 28702 | 36597 | 11427 | 9873 | 5820 | 6462 |
| 5 | 12438 | 6791 | 2289 | 3673 | 1626 | 1255 |
| 10 | 4473 | 6531 | 2000 | 1650 | 1137 | 1098 |

Таблица 11 – Таблица с результатами применения методов q-learn и karafotias для функции Растригина.

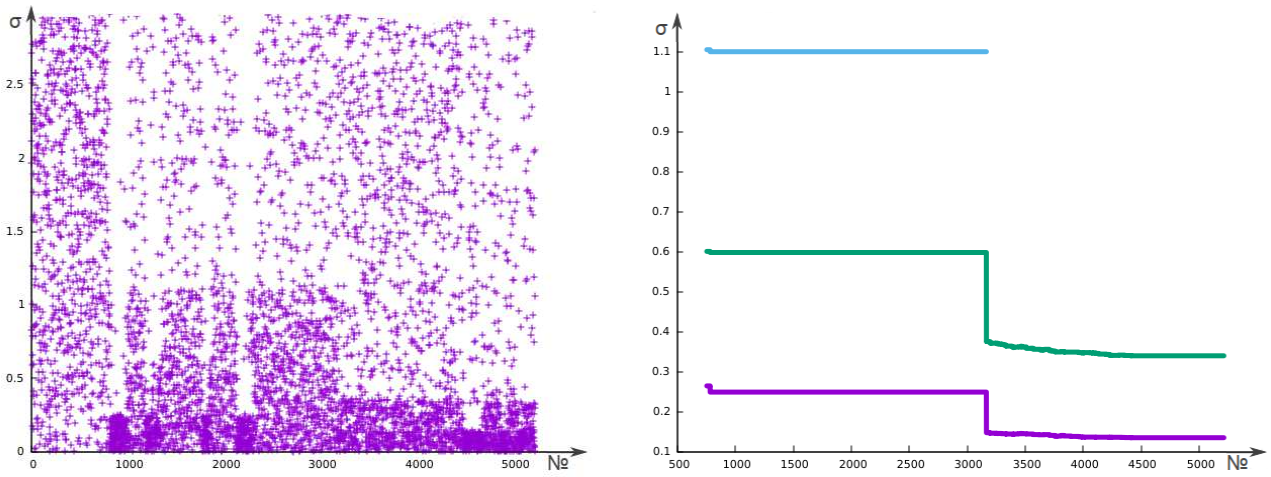
| k = 1 | | | | | | |
|---|---|---|---|---|---|---|
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 9003 | 12905 | 3553 | 2701 | 2296 | 1941 |
| 5 | 5730 | 4453 | 1393 | 2749 | 1130 | 1258 |
| 10 | 2213 | 3085 | 1161 | 1225 | 391 | 429 |
| k = 2 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 20005 | 20819 | 7166 | 13342 | 5874 | 7870 |
| 5 | 11153 | 11856 | 3775 | 4542 | 1775 | 2584 |
| 10 | 8768 | 4603 | 3719 | 1648 | 1740 | 2558 |
| k = 3 | | | | | | |
| $\lambda$ / $\mu$ | 1 | | 3 | | 7 | |
| | earpc | uearpc | earpc | uearpc | earpc | uearpc |
| 1 | 32533 | 43832 | 13061 | 10068 | 9797 | 9775 |
| 5 | 8952 | 11581 | 4434 | 6799 | 3951 | 4914 |
| 10 | 8320 | 10539 | 2611 | 2479 | 1742 | 1907 |

Таблица 12 – Таблица с результатами применения методов earpc и uearpc для функции Растригина.




(a) График выбранных значений $\sigma$. (b) График значения точки разбиения.

Рисунок 16 – Графики работы метода *uarpc* для функции Растригина.




(a) График выбранных значений $\sigma$. (b) График значения точек разбиения.

Рисунок 17 – Графики работы метода *adaptive* для функции Растригина.

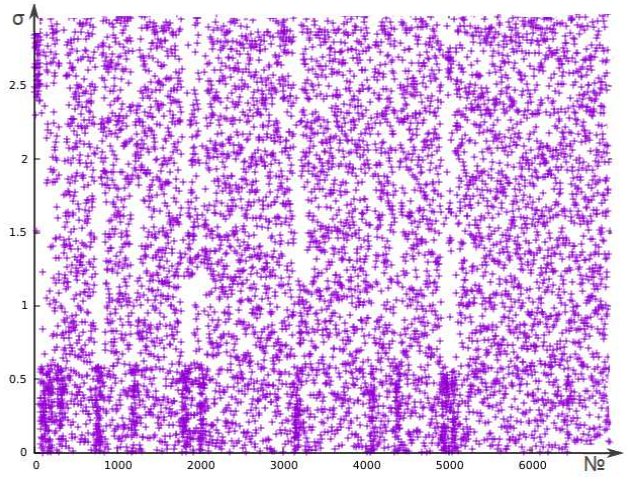



Рисунок 18 – График выбранных значений $\sigma$ в методе *karafotias* для функции Растригина.

метров ЭА. Наилучшие результаты были получены при использовании предложенного метода (2.3). Было продемонстрировано, что данный метод улучшает значения настраиваемых параметров на протяжении всего процесса оптимизации в отличие от остальных рассмотренных методов.

# ЗАКЛЮЧЕНИЕ

В работе предложены два метода адаптивной настройки параметров эволюционных алгоритмов с помощью обучения с подкреплением. В данных алгоритмах множество действий агента формировалось в процессе оптимизации, за счет разбиения диапазона допустимых значений параметра в ходе работы алгоритма. Один из предложенных алгоритмов был основан на двух существующих подходах к настройке параметров ЭА: алгоритме *EARPC* и методе, предложенном *Karafotias et al*. Второй алгоритм основан на том, что диапазон допустимых значений настраеваемого параметра переразбивается в случае, когда значения ожидаемой награды примерно равны для всех действий агента.

Предложенные методы адаптивной настройки параметров эволюционных алгоритмов были протестированы на четырех модельных задачах. Было проведено сравнение предложенных методов с существующими алгоритмами настройки параметров ЭА. Наилучшие результаты были получены при использовании метода с переразбиением диапазона значений настраеваемого параметра. Было продемонстрировано, что данный метод улучшает значения настраеваемых параметров на протяжении всего процесса оптимизации в отличие от остальных рассмотренных методов.

Таким образом была экспериментально проверена эффективность использования предложенного метода адаптивной настройки параметров эволюционных алгоритмов. Разработанный алгоритм удовлетворяет всем поставленным требованиям.

## СПИСОК ЛИТЕРАТУРЫ


1 *Скобцов Ю. А.* Основы эволюционных вычислений. — Донецк : ДонНТУ, 2008.

2 *Eiben A. E.*, *Smith J. E.* Introduction to Evolutionary Computing. — Berlin, Heidelberg, New York : Springer-Verlag, 2007.

3 *Mitchell M.* An Introduction to Genetic Algorithms. — Cambridge, MA : MIT Press, 1996. — 221 p.

4 *Eiben A.*, *Hinterding R.*, *Michalewicz Z.* Parameter control in evolutionary algorithms // Evolutionary Computation, IEEE Transactions on. — 1999. — Июль. — С. 124—141.

5 *Aleti A.*, *Moser I.* Entropy-based Adaptive Range Parameter Control for Evolutionary Algorithms // Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. — Amsterdam, The Netherlands, 2013. — С. 1501—1508. — (GECCO ’13).

6 *Karafotias G.*, *Eiben A. E.*, *Hoogendoorn M.* Generic Parameter Control with Reinforcement Learning // Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. — Vancouver, BC, Canada, 2014. — С. 1319—1326. — (GECCO ’14).

7 *Sutton R. S.*, *Barto A. G.* Reinforcement Learning: An Introduction. — Cambridge, MA, USA : MIT Press, 1998.

8 *Gosavi A.* Reinforcement Learning: A Tutorial Survey and Recent Advances // INFORMS Journal on Computing. — 2009. — Vol. 21, no. 2. — P. 178–192.

9 *Николенко С. И.*, *Тулупьев А. Л.* Самообучающиеся системы. — М., 2009.

10 *Watkins C. J. C. H.*, *Dayan P.* Q-learning // Machine Learning. — 1992. — С. 279—292.

11 Reinforcement Learning for Online Control of Evolutionary Algorithms / A. E. Eiben [и др.] // Proceedings of the 4th International Conference on Engineering Self-organising Systems. — Hakodate, Japan, 2007. — С. 151—160. — (ESOA’06).

12 *Aleti A.*, *Moser I.*, *Mostaghim S.* Adaptive Range Parameter Control // Evolutionary Computation (CEC), 2012 IEEE Congress on. — Июнь 2012. — С. 1—8.

13 *Uther W. T. B.*, *Veloso M. M.* Tree Based Discretization for Continuous State Space Reinforcement Learning // Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. — Madison, Wisconsin, USA, 1998. — С. 769—774. — (AAAI '98/IAAI '98).

14 *Stephens M. A.* Tests of fit for the logistic distribution based on the empirical distribution function // Biometrika. — 1979. — Т. 66, № 3. — С. 591—595.

15 *Rosenbrock H. H.* An Automatic Method for Finding the Greatest or Least Value of a Function // The Computer Journal. — 1960. — Т. 3, № 3. — С. 175—184.

16 *Rastrigin L.* Systems of extremal control // Theoretical Foundations of Engineering Cybernetics Series. — Moscow, 1974.