МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему Кластеризация данных с использованием моделирования эволюционных вычислений

Студент Н.Ю. Емельяненко _____
(И.О. Фамилия) (личная подпись)

Научный руководитель В.С. Климов _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д. ф.-м. н., доцент, С.В. Талалов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность исследования определена необходимостью совершенствования алгоритмов кластеризации данных для повышения их точности работы.

Использование алгоритмов кластеризации данных связано с решением различных практических задач, таких, например, как сегментация изображения, которая связана с кластеризацией его составных элементов – пикселей. Другой пример – подготовка обучающей выборки при создании классификатора, которая связана с предварительной кластеризацией данных для оценки их состава.

Алгоритмы машинного обучения в настоящее время активно развиваются. К ним относятся генетические алгоритмы (подобласть – «обучение с подкреплением»). Однако в настоящее время не изучена возможность применения генетических алгоритмов для решения задач кластеризации данных.

Цель исследования – разработка способа кластеризации данных, основанного на применении эволюционных вычислений.

Объектом исследования является кластеризация данных, **предметом** исследования – разработка методики использования генетического алгоритма для кластеризации данных.

Гипотезой исследования является предположение, что генетические алгоритмы, предназначенные для решения задач оптимизации, можно адаптировать для выполнения кластеризации данных.

Поставленная цель достигалась путем последовательного решения следующих **задач**:

1. Проведение анализа состояния вопроса по теме исследования.

2. Разработка методики по использования генетического алгоритма для кластеризации данных.

3. Разработка программной реализации для моделирования работы генетического алгоритма, основанного на предложенных подходах.

4. Проведение вычислительных экспериментов для оценки эффективности применения генетического алгоритма при кластеризации данных.

Исследованиями в области совершенствования алгоритмов кластеризации занимаются такие современники как: Kanzawa Y., Bao L., Oshio S., Chaimontree S., Yan Q., Ryazanov V., LaPlante F., Handl J., Yang X., Araújo D., Aszalós L., Almutairi N., Zhang Y., Zhu Q., Jin H., Vega-Pons S., Bhattacharyya D., Daliri M., Castellani U., Boryczka U. и др.

В ходе выполнения работы применялись такие **методы** теоретического исследования, как изучение и анализ научной литературы по проблемам кластеризации данных и вопросам практического использования генетических алгоритмов.

Также в ходе выполнения работы применялись практические **методы** исследования, такие как проведение вычислительных экспериментов, обработка статистических данных, программное моделирование работы генетических алгоритмов.

Научная новизна исследования – доказано, что генетические алгоритмы можно адаптировать под решение задачи кластеризации данных. При этом возможность генетического алгоритма выходить из локальных решений позволяет получать кластерные структуры лучше (с точки зрения плотности), чем при использовании алгоритма k-means.

Практическая значимость работы заключается в разработке методики применения генетических алгоритмов для решения задач кластеризации данных.

Апробация результатов исследования проходила на базе V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук».

На защиту выносятся:

1. Методика применения генетического алгоритма для кластеризации данных.
2. Результаты апробации предложенной методики.

В рамках написания магистерской диссертации были опубликованы следующие статьи:

1. Перспективы использования эволюционных вычислений при кластеризации данных [41].
2. Кластеризация изображения на основе оценки геометрической формы объектов [42].

В рамках выполнения магистерской диссертации разработано программное обеспечение для моделирования генетических алгоритмов и для кластеризации данных с использованием предложенных подходов. Результаты тестирование программного обеспечения показали состоятельность предложенной в диссертации методики кластеризации данных с использованием эволюционных вычислений.

1 АНАЛИЗ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ ДАННЫХ

1.1 Сравнение алгоритмов кластеризации данных

Использование алгоритмов кластеризации данных связано с решением различных практических задач, таких как:

- сегментация изображения, которая связана с кластеризацией его элементов составных – пикселей;
- подготовка обучающей выборки при создании классификатора, которая связана с предварительной кластеризацией данных для оценки их состава;
- другие задачи из различных областей науки и техники.

В таблице 1 приведен анализ вычислительной сложности различных алгоритмов кластеризации [1].

Таблица 1 – Сравнение вычислительной сложности алгоритмов

Название алгоритма	**Вычислительная сложность**
Иерархический	$O(n^2)$
k-means	$O(nkl)$, где k – число кластеров, l – число итераций
c-means	
Выделение связанных компонент	Зависит от алгоритма
Минимальное покрывающее дерево	$O(n^2 \log n)$
Послойная кластеризация	$O(\max(n,m))$, где $m < \frac{n(n-1)}{2}$

Результаты сравнения форм кластеров, входных и выходных данных для разных алгоритмов кластеризации представлены в таблице 2 [2, 3].

Таблица 2 – Сравнительная характеристика алгоритмов кластеризации.

Алгоритм кластеризации	**Форма кластеров**	**Входные данные**	**Результаты**
Иерархический	Произвольная	Число кластеров или порог расстояния для усечения иерархии	Бинарное дерево кластеров
k-means	Гиперсфера	Число кластеров	Центры кластеров
c-means	Гиперсфера	Число кластеров, степень нечеткости	Центры кластеров, матрица принадлежности
Выделение связанных компонент	Произвольная	Порог расстояния R	Дерево кластеров
Минимальное покрывающее дерево	Произвольная	Число кластеров или порог расстояния для удаления ребер	Дерево кластеров
Послойная кластеризация	Произвольная	Последовательность порогов расстояния	Дерево кластеров с различными уровнями иерархии

В настоящее время наибольший интерес с точки зрения перспектив развития представляют адаптивные алгоритмы кластеризации, основанные на машинном обучении. К ним относятся алгоритмы k-means, c-means, расширяющийся нейронный газ, самоорганизующиеся карты Кохонена. Наибольшую популярность из-за сочетания простоты математического аппарата и высокой эффективности получил метрический алгоритм классификации k-means.

Работа алгоритма k-means основана на случайном выборе центров кластеров и циклическом уточнении на каждой итерации их расположения.

Для оценки принадлежности объектов кластеризации к одному из центров кластеров обычно используются следующие метрики (1.1-1.3):

- Евклидово расстояние (метрика L_2) :

$$d_E(x,y) = \sqrt{\sum_i (x_i - y_i)^2}, \quad (1.1)$$

где $x = (x_1, x_2, ..., x_m)$ и $y = (y_1, y_2, ..., y_m)$ векторы значений признаков двух сравниваемых объектов *x* и *y*. *m* – размерность векторов.

- Расстояние Манхэттена (метрика L_1) :

$$d_M(x,y) = \sum_i |x_i - y_i| \quad (1.2)$$

- Расстояние Чебышева (метрика L_∞) :

$$d_C(x,y) = \max(|x_1 - y_1|, |x_2 - y_2|, ..., |x_m - y_m|) \quad (1.3)$$

Для оценки плотности кластерной структуры, найденной алгоритмом k-means, применяется показатель *E* (чем он меньше – тем лучше):

$$E = \sum_{i=1}^{m} \sum_{p \in Cluster_i} (M - C_i)^2 \quad (1.4)$$

где $M \in Cluster_i$ - объект, принадлежащий кластеру $Cluster_i$, C_i – центр данного кластера, *m* – общее количество объектов подвергаемых кластеризации.

Алгоритм k-means обладает существенным недостатком (который свойственен всем алгоритмам, основанным на машинном обучении) – отсутствие гарантии получения оптимальной кластерной структуры.

Теоретически, преодоление данного недостатка алгоритма k-means возможно за счет использования технологий эволюционных вычислений (генетического алгоритма).

1.2 Основные элементы задачи оптимизации

Генетические алгоритмы направлены на решение оптимизационных задач различных типов. К задачам оптимизации относится поиск

экстремумов (максимумов и минимумов) заданной функции, которую в теории оптимизации называют целевой.

Практически всегда целевая функция представляет собой сложную зависимость от набора входных параметров. Решением задачи оптимизации называют набор входных параметров, удовлетворяющих условиям задачи и при которых функция целевая функция достигает своего максимального или минимального значения.

Таким образом, основными элементами задачи оптимизации являются:

- целевая функция;
- управляющие (входные) параметры целевой функции
- ограничения задачи оптимизации, накладываемые на значения целевой функции и входных параметров

Ограничения задачи оптимизации бывают различных видов:

- Общие ограничения. К ним относятся ограничения, накладываемые на значения входных параметров целевой функции, например, (1.1):

$$x \geq 0, \text{ где } x = (x_1, ..., x_n), \quad (1.1)$$

- Специальные ограничения. К ним относятся ограничения, накладываемые на значения функций, которые зависят от входных параметров, например, (1.2):

$$g_i(x) \leq 0, \text{ где } x = (x_1, ..., x_n). \quad (1.2)$$

где g_i – i-ая функция, накладывающая ограничения, i – индекс для нумерации ограничений.

- Ограничения типа неравенств. К ним относятся ограничения, накладываемые на значения целевой функции f, например, (1.3):

$$f_i(x) \geq 0, \text{ где } x = (x_1, ..., x_n). \quad (1.3)$$

где i – индекс для нумерации ограничений.

- Ограничения типа равенств. Пример представлен на (1.4):

$$f_i(x) = 0, \text{ где } x = (x_1, ..., x_n), \quad (1.4)$$

где i – индекс для нумерации ограничений.

- Активные ограничения. К ним относятся ограничения, на границах которых находятся решения, пример представлен на 1.5:

$$x_{i\min} \le x_i \le x_{i\max}, \quad (1.4)$$

Набор рассмотренных выше ограничений при решении конкретной задачи оптимизации формируются допустимую область, которая может обладать разными свойствами.

Для того, чтобы решить допустимую оптимизационную задачу необходимо найти оптимальное решение из допустимой области. При этом решение, представляющее из себя набор значений x (1.5) называется допустимым решением, если оно удовлетворяет всем ограничениям задачи:

$$x = (x_1, ..., x_n) \quad (1.5)$$

Задача оптимизации является допустимой лишь в том случае, если множество допустимых решений является непустым, т.е. существует как минимум одно допустимое решение, в противном случае такая задача оптимизации является недопустимой.

Для того, чтобы решение x являлось оптимальным решением задачи оптимизации оно должно удовлетворять следующим условиям:

1. Решение должно являться допустимым.
2. При данном решении значение целевой функции должно достигать глобального экстремума.

При решении задач с использование стохастических методов могут применяться следующие критерии для остановки поиска оптимального решения:

- Изменение значений $x_1, ..., x_n$ за последние несколько итераций работы алгоритма меньше заданного порогового значения.
- Изменение значения целевой функции за последние несколько итераций работы алгоритма меньше заданного порогового значения.

Таким образом, для того, чтобы задачу поиску кластерной структуры можно было представить, в виде оптимизации необходимо определить ее

основные элементы: целевая функция, оплавляющие параметры и ограничения.

В качестве целевой функции предложено рассматривать формулу расчета плотности *E* кластерной структуры. Чем, меньше значение *E*, ближе расположены объекты рассматриваемого кластера к его центу C_i. Так как необходимо, чтобы кластерная структура была максимально плотной, то решаемая задача оптимизации будет выглядеть следующим образом (1.5):

$$E(\mathbf{X},\mathbf{C}) = \sum_{i=1}^{m} \sum_{M_m \in Cluster_i} d(M_m, C_i)^2$$
$$E(\mathbf{X},\mathbf{C}) \to \min \tag{1.5}$$

Входными параметрами функции $E(\mathbf{X},\mathbf{C})$ являются:

1. $\mathbf{X}^m$ – множество объектов, подвергаемых кластеризации, представленных как (1.6):

$$X^m = M_1, M_2, ..., M_m \ , \tag{1.6}$$

где m – количество объектов кластеризации. При этом каждый объект кластеризации описывается набором параметров $(x_1, ..., x_n)$ (1.7):

$$X^m = (x_1, ..., x_n)_1, (x_1, ..., x_n)_2, ..., (x_1, ..., x_n)_m \ , \tag{1.7}$$

где n – количество параметров объектов *M* кластеризации.

2. $\mathbf{C^k}$ – центры кластеров в полученной кластерной структуре (1.8):

$$C^k = C_1, C_2, ..., C_k \ , \tag{1.8}$$

где k – количество выделяемых кластеров. Каждый центр кластер кластера описывается тем же набором параметров, что и объекты кластеризации (1.9):

$$C^k = (x_1, ..., x_n)_1, (x_1, ..., x_n)_2, ..., (x_1, ..., x_n)_k \tag{1.9}$$

Так как при оптимизации плотности кластерной структуры мы не можем изменять объекты кластеризации, то задача оптимизации сводиться к поиску оптимального расположения центров кластеров **C**.

Расположение центров кластеров влияет на принадлежность объектов к кластерам, ведь каждый объект в результате кластеризации будет относиться

к тому кластеру, к центру которого он ближе (расстояние определяется метрикой).

Таким образом, входными параметрами задачи оптимизации кластерной структуры являют параметры центров кластеров (1.9), а целевая функция выглядит так:

$$f_{np}(C_1,...,C_m) = \sum_{i=1}^{m} \sum_{M_m \in Cluster_i} d(M_m, C_i)^2 \to \min \qquad (2.10)$$

Ограничениями задачи оптимизации являются области определения рассматриваемых параметров $(x_1,...,x_n)$.

Теперь, когда кластеризации данных представлена как задача оптимизации, становится возможным использование методов оптимизации для поиска кластерной структуры.

Следующим этапом исследования является анализ математического аппарата эволюционных вычислений и формирование рекомендаций по выбору оптимальных параметров при решении задачи кластеризации.

2 РАЗРАБОТКА ПРИНЦИПОВ КЛАСТЕРИЗАЦИИ ДАННЫХ НА ОСНОВЕ ЭВОЛЮЦИННЫХ ВЫЧИСЛЕНИЙ

2.1 Анализ математического аппарата эволюционных вычислений

Так как генетический алгоритм основан на принципах теории Дарвина, в нем применяется термины, близкие к понятиям принципов эволюции, но несущие в себе математический смысл. Основные понятия генетических алгоритмов:

- Функция приспособленности $f_{\text{пр}}$ – целевая функция решаемой с помощью генетического алгоритма задачи оптимизации ($f_{np} \to \min$ или $f_{np} \to \max$)
- Особь O_i – набор входных параметров функция приспособленности, один из вариантов решения задачи оптимизации:

$$O_i = (x_1, ..., x_n), \tag{2.1}$$

где n – количество входных параметров задачи оптимизации.

- Хромосома – вектор параметров, состоящий или из вещественных генов или из бинарных генов.
- Вещественный ген – значение одного из параметров, формируемых вариант решения задачи оптимизации (другими словами – один из параметров особи). В этом случае особь O_i состоит из генов в количестве n: $x_1, ..., x_n$.
- Бинарный ген – бит, закодированного в двоичном виде одного из параметров, варианта решения задачи оптимизации (другими словами – бит закодированного в двоичном виде одного из параметров особи). В этом случае гены особи O_i будут представлены бинарной строкой, полученной путем конвертирования значений $x_1, ..., x_n$ в двоичный вид и объединение полученных подстрок. Длина двоичной подстроки для каждого параметра x_i зависит от требуемой точности поиска генетическим алгоритмом решения

поставленной оптимизационной задачи (чем выше точность требуется, тем длинней будет двоичная строка).

- Приспособленность – значение целевой функции для рассматриваемой особи O_i:

$$f_{пр}(O_i) = f_{пр}(x_1,...,x_n) \quad (2.2)$$

- Популяция – набор особей на текущей итерации генетического алгоритма, несколько вариантов решения задачи оптимизации.

$$Pop(t) = (O_1,...,O_N), \quad (2.3)$$

где t – номер итерации выполнения генетического алгоритма, N – размер популяции, который остается постоянным в начале каждой итерации генетического алгоритма.

Принцип работы генетического алгоритма заключается в стохастическом поиске решения задачи оптимизации. Причем процесс поиска носит итерационный характер. На каждой следующей итерации своей работы генетический алгоритм производит отсеивание нежелательных (слабых) решений, а на основе лучших решений, путем их комбинации, алгоритм стремиться получить новые решения с более высокими показателями приспособленности.

Все конфигурации генетических алгоритмов работают на основе последовательного выполнения следующих шагов.

На первом этапе производится генерация начальной популяции особей. При этом случайным образом генерируется набор (заданной размерности) вариантов решения задачи оптимизации.

На втором этапе осуществляется проверка достижения критерия остановки поиска решения задачи оптимизации. В качестве такого критерия можно использовать один из следующих вариантов:

- Все решения текущей популяции находятся в области экстремума и новые итерации не позволяют сгенерировать решения с большей приспособленностью.

- На протяжении нескольких итераций в популяции не изменяется особь с наибольшей приспособленностью. Это значит, что текущая конфигурация генетического алгоритма наилучшее решение и не может его улучшить.

Если критерий поиска оптимального решения не достигнут, то к текущей популяции последовательно применяются операторы: выбора родителей для скрещивания, скрещивание, мутация и отбор особей в новую популяцию.

Общий вид генетических алгоритмов представлен на рисунке 1.1.

Генетический алгоритм является метаэвристическим, это значит, что он является основой для создания конкретных конфигураций эвристических алгоритмов. Дело в том, что существует большое количество вариантов реализации оператора выбора родителей, оператора скрещивания, оператора мутации и оператора отбора особей. Причем у каждого варианта реализации еще существует и большое количество параметров, влияющих на процесс выполнения используемого оператора.

В этой связи, под конкретной конфигурацией генетического алгоритма принято понимать набор используемых в текущей реализации операторов и и конкретные значение их параметров.

Рассмотрим подробнее варианты реализации операторов генетического алгоритма.

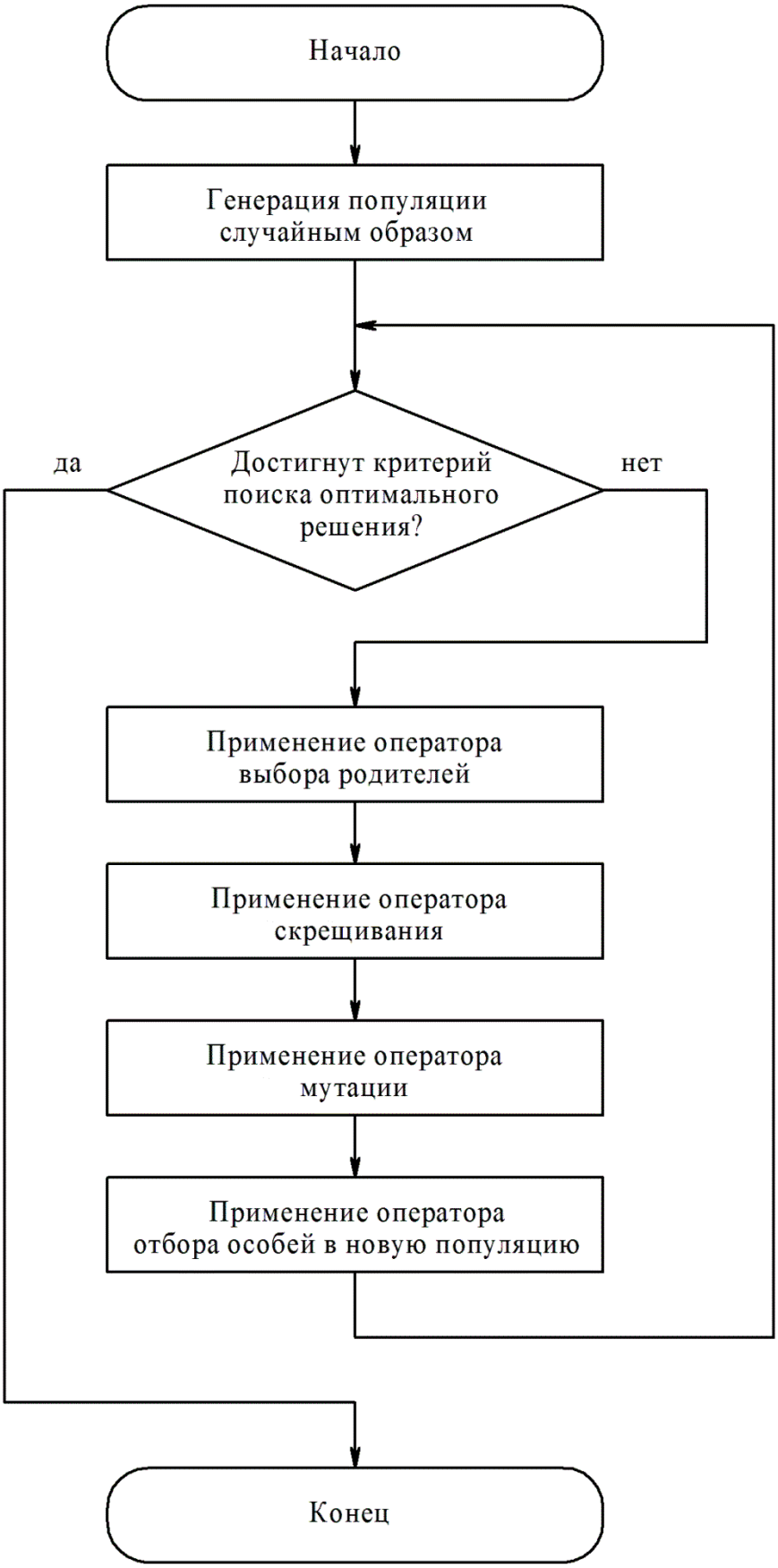

Рисунок 2.1 – Общий вид генетических алгоритмов

Операторы выбора родителя существуют в следующих вариантах: панмиксия, генотипный инбридинг, фенотипный инбридинг, генотипный аутбридинг, фенотипный инбридинг, селекция, турнирный отбор, отбор методом рулетки

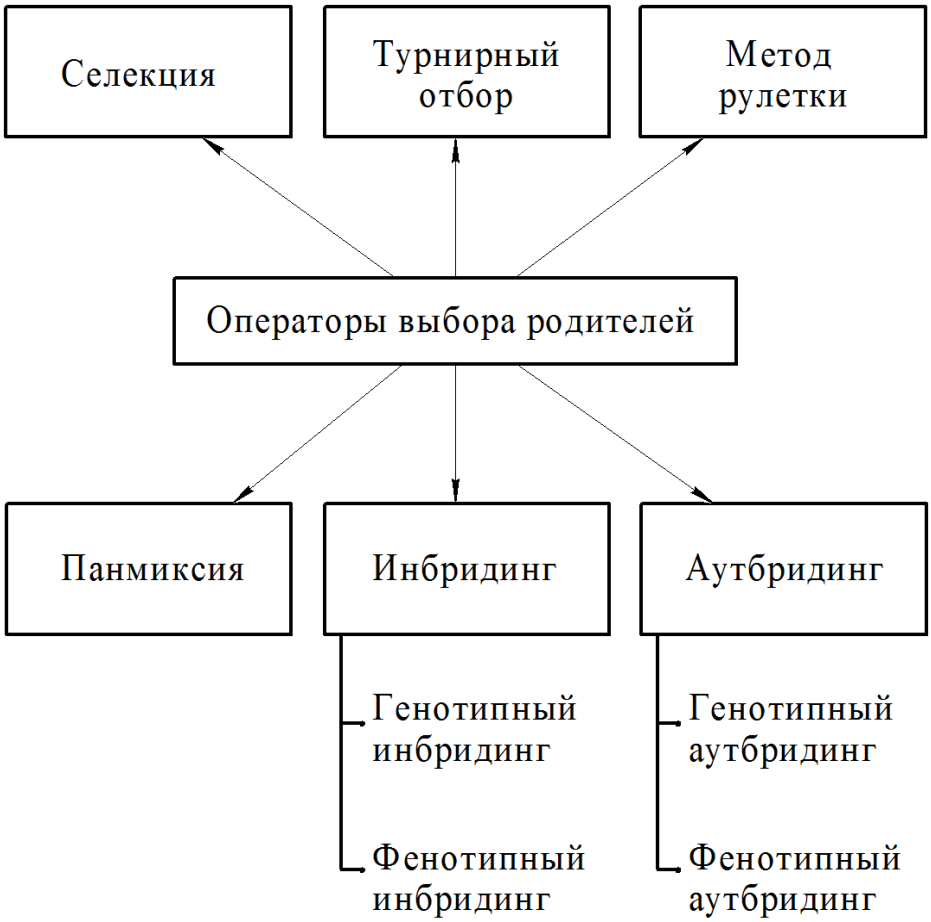

Рисунок 2.2 – Операторы выбора родителей

При использовании панмиксии оба родителя P_1 и P_2 для скрещивания выбираются из популяции Pop случайным образом. Т.е. выполняются действия представленные на 2.4:

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow random(O_1,...,O_N) \\ P_1 \neq P_2 \end{cases} \quad (2.4)$$

В этом случае возможны следующие варианты: особь O_i может принимать участие в нескольких родительских парах, некоторые особи не участвуют в скрещивании.

Данный оператор выбора родителей является самым распространённым при использовании в генетических алгоритмах, так он отличается универсальностью и простотой реализации. При его программной реализации чаще всего используется генератор целых случайных чисел в диапазоне [0; N], где N – размер популяции.

Недостатком панмиксии является снижение эффективности данного оператора с ростом размера популяции.

При использовании инбридинга родители выбираются следующим образом. Первый родитель P_1 выбирается случайным образом, а второй родитель P_2 выбирается таким, чтобы обеспечить наименьшее различие между ними.

Инбридинг бывает генотипным и фенотипным. При генотипном инбридинге второй родитель P_2 выбирается из расчета обеспечения наименьших различий в генах родителей. Таким образом, расчеты выглядят так (2.5):

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow O_k \;\; if \;\; [d(P_1,O_k) \rightarrow \min], \\ P_1 \neq P_2 \end{cases} \qquad (2.5)$$

где d (A, B) – функция расчета расстояния между генами особей A и B. Если гены представлены в бинарном виде, то функция d () может возвращать расстояние Хэмминга между последовательностями генов. Если гены вещественные, то функция в качестве функции используется метрика Евклида или Манхэттена. k – индекс с наиболее близким набором генов к первому родителю P_1 ($k \in 1,...,N$).

В фенотипном инбридинге второй родитель P_2 выбирается из расчета обеспечения наименьших различий в значениях функции приспособленности $f_{\text{пр}}$. Т.е. фенотипный инбридинг происходит следующим образом (2.6):

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow O_k \;\; if \;\; [\,| f_{\text{пр}}(P_1) - f_{\text{пр}}(O_k)\,| \rightarrow \min], , \\ P_1 \neq P_2 \end{cases} \tag{2.6}$$

Инбридинг обоих типов применяется, когда требуется ограничить поиск решений локальными узлами. Применение данного оператора приводит к разбиению популяции на группы решений, локализованных по экстремумам.

При использовании аутбридинга выбор родителей для скрещивания осуществляется следующим образом. Первый родитель P_1 выбирается случайным образом, а второй родитель P_2 выбирается так, чтобы обеспечить максимальное различие с первым родителем. Аутбридинг, также, как и инбридинг, бывает генотипным и фенотипным. Генотипный аутбридинг выполняется следующим образом (2.7):

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow O_k \;\; if \;\; [d(P_1,O_k) \rightarrow \max], \\ P_1 \neq P_2 \end{cases} \tag{2.7}$$

где $d()$ – функция расчета расстояния между генами особей.

При этом фенотипный аутбридинг, с учетом значений функции приспособленности $f_{\text{пр}}$ особей O_i рассчитывается так (2.8):

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow O_k \;\; if \;\; [\,| f_{\text{пр}}(P_1) - f_{\text{пр}}(O_k)\,| \rightarrow \max], \\ P_1 \neq P_2 \end{cases} \tag{2.8}$$

Использование аутбридинга позволяет избежать предварительной сходимости решений к локальным экстремумам, заставляя генетический алгоритм анализировать не затронутые на предыдущих итерациях области значений входных параметров.

Использование селекции для выбора родителей связано со следующими особенностями. Данный оператор позволяет выбирать для скрещивания только тех родителей, для которых значение функции приспособленности $f_{\text{пр}}$ выше среднего значения приспособленности популяции *Pop* данной итерации. Т.е. селекция выглядит следующим образом (2.9):

$$\begin{cases} k = random(1, N) \\ m = random(1, N) \\ P_1 \leftarrow O_k \;\; if \;\; f_{\text{пр}}(O_k) > \dfrac{\sum_{i=1}^{N} f_{\text{пр}}(O_i)}{N} \\ P_2 \leftarrow O_m \;\; if \;\; f_{\text{пр}}(O_m) > \dfrac{\sum_{i=1}^{N} f_{\text{пр}}(O_i)}{N} \\ P_1 \neq P_2 \end{cases}, \tag{2.9}$$

где *random* (1, *N*) – функция выбора случайного целого значения на участке [1, *N*], где *N* – размер текущей популяции.

Селекция обеспечивает быструю сходимость генетического алгоритма, но часто приводит к нахождению локальных экстремумов вместо глобальных.

Также одним из вариантов выбора родителей для скрещивания является турнирный отбор. Основная его идея заключается в формировании случайным образом групп особей (турнир) и определение из каждой группы особи с наилучшей приспособленностью. Победа в турнире позволяет особи стать одним из родителей. Входными параметрами турнирного отбора являются:

- *t* – размер турнира;
- *n* – количество турниров, равно количеству родителей, которых необходимо отобрать.

На первом этапе выполнения выбора обителей формируется массив M турнира (2.10):

$$m_{ij} = random(O_1,...,O_N)$$
$$i = 1...n, \quad j = 1...t \tag{2.10}$$
$$M = \begin{pmatrix} m_{11} & m_{12} & ... & m_{1t} \\ m_{21} & m_{22} & ... & m_{2t} \\ ... & ... & ... & ... \\ m_{n1} & m_{n2} & ... & m_{nt} \end{pmatrix}$$

При проведении турниров из каждой строки массива выбирается особь с максимальным значением функции приспособленности, таким образом формируется вектор особей *results* которые могут быть родителями (2.11):

$$results = \begin{pmatrix} \max(f_{np}(m_{11}), f_{np}(m_{12}),..., f_{np}(m_{1t})) \\ \max(f_{np}(m_{21}), f_{np}(m_{22}),..., f_{np}(m_{2t})) \\ ... \\ \max(f_{np}(m_{n1}), f_{np}(m_{n2}),..., f_{np}(m_{nt})) \end{pmatrix} \tag{2.12}$$

Формирование пар для скрещивания на основе вектора *results* формируется случайным образом (1.17):

$$\begin{cases} k = random(1, rows(results)) \\ m = random(1, rows(results)) \\ P_1 \leftarrow results_k \\ P_2 \leftarrow results_m \\ P_1 \neq P_1 \end{cases}, \tag{2.13}$$

где *row*(*results*) – функция, возвращающая количество элементов вектора *results.*

При выборе родителей методом рулетки используются следующие принципы. Для текущей популяции рассчитывается сумма приспособленности всех особей $O_1,\ldots,O_N$. Полученное значение используется для того, чтобы определить длину диапазона значений для каждой особи $O_1,\ldots,O_N$ на отрезке [0; 1]. При этом должна быть соблюдена пропорциональность – чем больше приспособленность особи, тем больше

диапозон значений, занимаемой данной особью на отрезке [0; 1]. Длину D диапазона значений на отрезке на отрезке [0; 1] для каждой особи можно рассчитать по формуле 2.14:

$$D_i = \frac{f_{\text{пр}}(O_i)}{\sum_{i=1}^{N} f_{\text{пр}}(O_i)} \qquad (2.14)$$

Графическая интерпретация выбора диапазона значений для каждой особи популяции показана на рисунке 2.3.

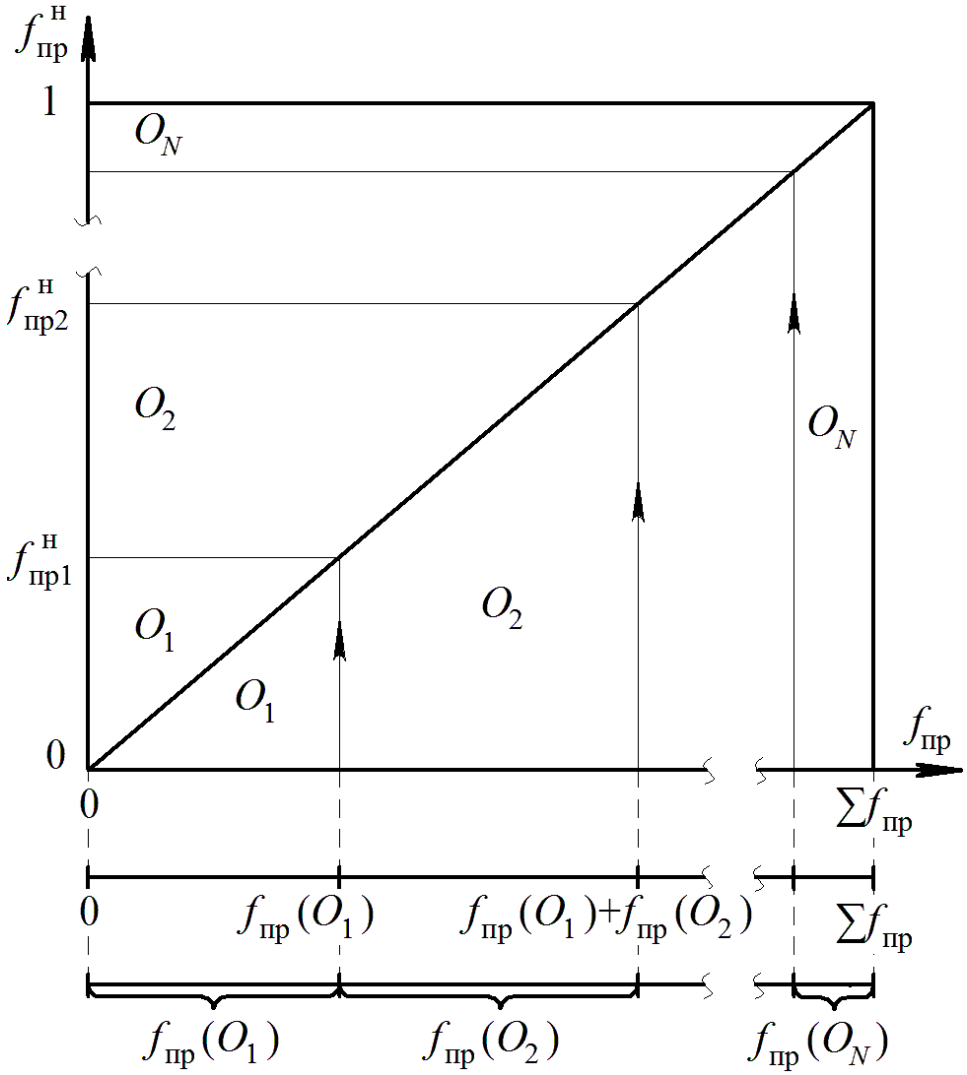

Рисунок 2.3 – Выбор размеров секторов для определения родителей методом рулетки на основе приспособленности $f_{\text{пр}}$ особей $O_1,\ldots,O_2$ текущей популяции (N – размер популяции, $f_{\text{пр}}^{\text{н}}$ – нормированные значения функции приспособленности)

Теперь, для определения того, какие особи будут являться родителями, необходимо генерировать случайные числа на диапазоне [0; 1] и определять, к диапазону какой особи относиться данное число (2.15).

$$\begin{aligned} & k = 1...K \\ & temp_k = random(0,1) \quad , \\ & P_k = \begin{cases} O_1 \ \ if \ \ 0 \le temp_k < D_1 \\ O_2 \ \ if \ \ D_1 \le temp_k < D_2 \\ \qquad\qquad ... \\ O_N \ \ if \ \ D_{N-1} \le temp_k < D_N \end{cases} \end{aligned} \qquad (2.15)$$

где K – количество отбираемых родителей.

При использовании данного оператора у особей, обладающих большей приспособленностью, вероятность стать родителем больше, чем у особей с меньшей приспособленностью.

После формирования пар родителей, выбранных с использованием одного из операторов, можно переходить к формированию на их основе потомков. Для выполнения этого действия необходимо воспользоваться одним из операторов скрещивания родителей. Операторы скрещивания родителей делятся на две группы: операторы для работы с вещественными генами и операторы для работы с бинарными генами (рисунок 2.4).

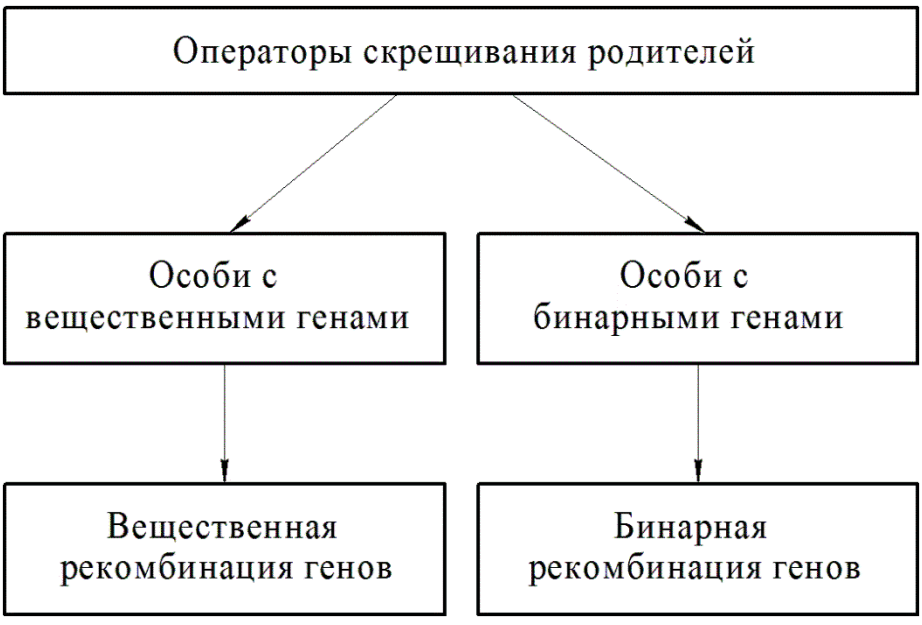

Рисунок 2.4 – Операторы скрещивания родителей

К операторам для работы с вещественными генами относятся: дискретная рекомбинация, промежуточная рекомбинация и линейная рекомбинация (рисунок 2.5).

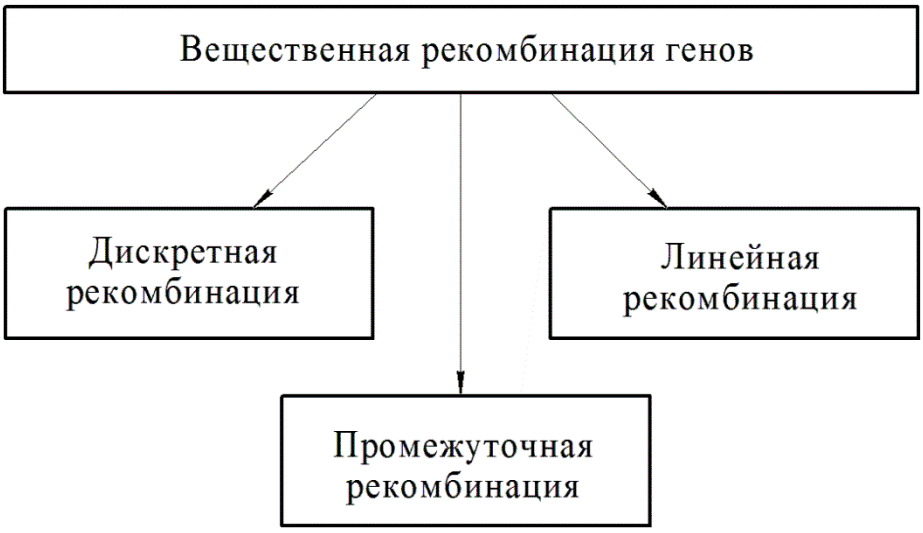

Рисунок 2.5 – Операторы вещественной рекомбинации генов

Дискретная рекомбинация применяется в тех случаях, когда гены особи представлены вещественными числами. В этом случае для формирования двух потомков *D*1 и *D*2 на основе родителей *P*1 и *P*2 необходимо определить схемы *scheme_f* и *scheme_s* обмена генами. Схемы обмена генами – это вектор с количеством компонентов равным количеству генов особи. Значения компонентов схем могут принимать два значения либо 0, либо 1. Схемы в данном случае генерируются случайным образом с помощью функции *random* (), возвращающей одно из двух значений с одинаковой вероятностью (2.16):

$$\begin{aligned} scheme\_f &= (random(0;1)_1,...,random(0;1)_n) \\ scheme\_s &= (random(0;1)_1,...,random(0;1)_n) \end{aligned} \qquad (2.16)$$

Затем на основе двух полученных схем, производится генерация двух потомков *D*1 и *D*2 на основе родителей *P*1 и *P*2. Если в схеме встречается значение компонента равное 0, то ген наследуется от родителя *P*1, если значение равное 1, то ген наследуется от родителя *P*2. Схема *scheme_f*

предназначена для генерации потомка $D1$, а *scheme_s* – для потомка $D1$ (2.17):

$$i = 1...n$$
$$D1_i = \begin{cases} P1_i \ \ if \ \ scheme\_f_i = 0 \\ P2_i \ \ \ if \ \ scheme\_f_i = 1 \end{cases} \qquad (2.17)$$
$$D2_i = \begin{cases} P1_i \ \ if \ \ scheme\_s_i = 0 \\ P2_i \ \ \ if \ \ scheme\_s_i = 1 \end{cases}$$

Промежуточная рекомбинация применяется при вещественных генах особей. При использовании данного оператора скрещивания сначала генерируется множитель рекомбинации α, который выбирается случайным образом на участке от [-0,25; 1,25]. Данный множитель генерируется для каждого *i*-ого гена потомка *D* по отдельности. Таким образом, гены потомка *D* формируются по формуле (2.18):

$$D_i = P1_i + \alpha_i \cdot (P2_i - P1_i) \qquad (2.18)$$

Для снижения количество требуемых вычислений при получении потомков на основе родителей возможно применение линейной рекомбинации. Данный оператор отличается от промежуточной рекомбинации тем, что множитель α генерируется один раз на каждого потомка. Таким образом, гены потомков рассчитывается по формуле (2.19):

$$\begin{cases} D1_i = P1_i + \alpha_1 \cdot (P2_i - P1_i) \\ D2_i = P1_i + \alpha_2 \cdot (P2_i - P1_i) \end{cases} \qquad (2.19)$$

К операторам для работы с бинарными генами относятся: одноточечный кроссинговер, двухточечный кроссинговер, многоточечный кроссинговер, однородный кроссинговер, триадный кроссинговер и перетасовочный кроссинговер (рисунок 2.6).

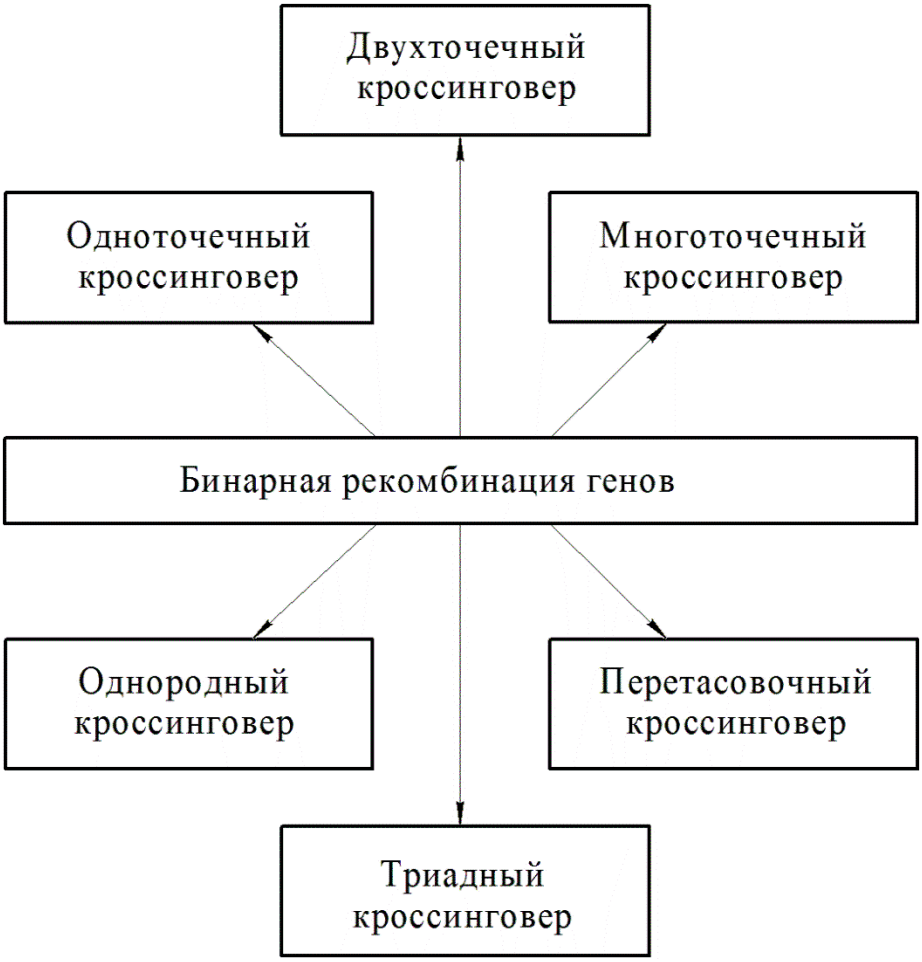

Рисунок 2.6 – Операторы бинарной рекомбинации генов

Одноточечный кроссинговер предназначен для работы с генами, представленными в бинарном виде. В этом случае, родители представлены набором из k генов. Для одноточечного кроссинговера необходимо выбрать точку M, в которой набор генов будет разделен на две подстроки. Точка M выбирается случайным образом из следующих значений $\{1, k-1\}$. На основе двух родителей $P1$ и $P2$ с помощью одноточечного кроссинговера можно получить двух потомков $D1$ и $D2$, путем объединения подстрок от разных родителей.

Расчет двух потомков $D1$ и $D2$ на основе точки M кроссинговера выглядит, так как это представлено в (2.20):

$$
\begin{aligned}
&P1 = (a_1, a_2, ..., a_k) \\
&P2 = (b_1, b_2, ..., b_k) \\
&D1_i = \begin{cases} a_i \;\; if \;\; i < M \\ b_i \;\; if \;\; i \geq M \end{cases} \\
&D2_i = \begin{cases} a_i \;\; if \;\; i \geq M \\ b_i \;\; if \;\; i < M \end{cases}
\end{aligned}
\tag{2.20}
$$

Двухточечный кроссинговер отличается от одноточечного наличием дополнительной точки M_2 разрыва хромосом родителей $P1$ и $P2$. При формировании первого потомка $D1$ набор генов, заключённых между точками M_1 и M_2 наследуется от второго родителя $P2$, а остальные гены от первого родителя $P1$. А при формировании второго потомка $D2$ обмен генами происходит в противоположной комбинации.

Таким образом, выполняемые расчеты при двухточечном кроссинговере выглядят так, как это представлено в формулах (2.21):

$$
\begin{aligned}
&P1 = (a_1, a_2, ..., a_k) \\
&P2 = (b_1, b_2, ..., b_k) \\
&D1_i = \begin{cases} a_i \;\; if \;\; (i < M_1) \; or \;\; (i \geq M_2) \\ b_i \;\; if \;\; M_1 \leq i < M_2 \end{cases} \\
&D2_i = \begin{cases} a_i \;\; if \;\; M_1 \leq i < M_2 \\ b_i \;\; if \;\; (i < M_1) \; or \;\; (i \geq M_2) \end{cases} \\
&M_1 < M_2
\end{aligned}
\tag{2.21}
$$

Когда количество t точек разрыва хромосом родителей больше чем две, то принято говорить, что используется многоточечный кроссинговер (t>2).

В этом случае расчёт потомков $D1$ и $D2$ на основе родителей $P1$ и $P2$ и точек разрыва хромосом $M_1,\ldots, M_t$ выглядит следующим образом (2.22):

$$
\begin{aligned}
&P1 = (a_1, a_2, ..., a_k)\\
&P2 = (b_1, b_2, ..., b_k)\\
&D1_i = \begin{cases} a_i \;\; if \;\; [(i \ge 0)\; and \;\; (i < M_1)]\; or\; [(i \ge M_2)\; and \;\; (i < M_3)]\; or\; ... \\ \qquad\qquad ...or\; [(i \ge M_{t-2})\; and \;\; (i < M_{t-1})] \\ b_i \;\; if \;\; (i \ge M_1)\; and \;\; (i < M_2) or\; [(i \ge M_3)\; and \;\; (i < M_4)]\; or\; ... \\ \qquad\qquad ...or\; [(i \ge M_{t-1})\; and \;\; (i < M_t)] \end{cases}\\
&D2_i = \begin{cases} a_i \;\; if \;\; (i \ge M_1)\; and \;\; (i < M_2) or\; [(i \ge M_3)\; and \;\; (i < M_4)]\; or\; ... \\ \qquad\qquad ...or\; [(i \ge M_{t-1})\; and \;\; (i < M_t)] \\ b_i \;\; if \;\; [(i \ge 0)\; and \;\; (i < M_1)]\; or\; [(i \ge M_2)\; and \;\; (i < M_3)]\; or\; ... \\ \qquad\qquad ...or\; [(i \ge M_{t-2})\; and \;\; (i < M_{t-1})] \end{cases}\\
&M_1 < M_2 < ... < M_t
\end{aligned}
\qquad (2.22)
$$

Однородный кроссинговер применяется для получения потомков $D1$ и $D2$ на основе двух родителей $P1$ и $P2$ с бинарными генами.

При использовании данного оператора для того чтобы определить, как будет производиться обмен генами при скрещивании необходимо случайным образом сгенерировать схему *scheme* рекомбинации. Схема представляет собой бинарный вектор с количеством компонентов равным количеству генов особи. Значения компонентов схем могут принимать два значения либо 0, либо 1.

При скрещивании родителей $P1$ и $P2$ наследование генов потомками $D1$ и $D2$ определяется схемой на основе расчётов, представленных на (2.23).

$$
\begin{aligned}
&P1 = (a_1, a_2, ..., a_k)\\
&P2 = (b_1, b_2, ..., b_k)\\
&scheme_i \leftarrow random(0, 1)\\
&D1_i = \begin{cases} P1_i \;\; if \;\; scheme_i = 0 \\ P2_i \;\; if \;\; scheme_i = 1 \end{cases}\\
&D2_i = \begin{cases} P1_i \;\; if \;\; scheme_i = 1 \\ P2_i \;\; if \;\; scheme_i = 0 \end{cases}
\end{aligned}
\qquad (2.23)
$$

Триадный кроссинговер используется, когда требуется снизить количество расчётов при выполнении скрещивании родителей. В этом случае схема *scheme* рекомбинации копируется из вектора значений генов случайной особи, присутствующей в популяции на текущей итерации работы генетического алгоритма.

Таким образом, генерирование потомков $D1$ и $D2$ на основе родителей $P1$ и $P2$ осуществляется следующим образом:

$$\begin{aligned} &P1 = (a_1, a_2, ..., a_k) \\ &P2 = (b_1, b_2, ..., b_k) \\ &scheme_i \leftarrow random(O_1, ..., O_N) \\ &D1_i = \begin{cases} P1_i \ \ if \ \ scheme_i = 0 \\ P2_i \ \ if \ \ scheme_i = 1 \end{cases} \\ &D2_i = \begin{cases} P1_i \ \ if \ \ scheme_i = 1 \\ P2_i \ \ if \ \ scheme_i = 0 \end{cases} \end{aligned} \qquad (2.24)$$

Перетасовочный кроссинговер применяется для родителей с бинарными генами. Его основная идея заключается в модифицировании родителей путем обмена между ними случайными генами и выполнение после этого одноточечного кроссинговера для генерирования потомков.

Таким образом, популяция меняется не только за счет добавления в нее потомков, но и за счет изменения генов родителей.

После выполнения скрещивания к потомкам применяется один из операторов мутации. Мутация необходимо для того, чтобы препятствовать схождению генетического алгоритма к локальному решению задачи оптимизации.

К операторам мутации относятся: мутация вещественных генов, двоичная мутация, плотность мутации, мутация присоединением, мутация вставкой, мутация удалением, мутация обменом (рисунок 2.7).

Рисунок 2.7 – Операторы мутации

Идея мутация вещественных генов заключается в выборе знака и величины изменения гены. Очевидно, что вероятность выбора знака должна быть одинаковой. При этом величину изменения гена необходимо ограничить, так как мутация не должна приводить к сильным изменениям особи. На практике величину δ изменения гена ограничивают половиной размера области изменения входного параметра, с которой данный ген связан (2.27):

$$\begin{aligned} & j = 1...n \\ & \delta_j \leftarrow \text{random}([0.5 \cdot \min(x_j)], [0.5 \cdot \max(x_j)]) \end{aligned}, \tag{2.27}$$

где $\min(x_j)$ и $\max(x_j)$ - минимальное и максимальное значения области определения параметра x_j.

Таким образом, величину δ изменения гена выбирают случайным образом из описанного выше диапазона. А сама мутация вещественных генов осуществляется следующим образом

$$
\begin{aligned}
&j = 1...n \\
&\delta_j \leftarrow \text{random}([0.5 \cdot \min(x_j)], [0.5 \cdot \max(x_j)]) \\
&\alpha_j \leftarrow \text{random}(0,1) \\
&D_j = \begin{cases} D_j - \delta \;\; if \;\; \alpha_j \le 0.5 \\ D_j + \delta \;\; if \;\; \alpha_j > 0.5 \end{cases}
\end{aligned}
\qquad (2.26)
$$

Двоичная мутация применяется к генам, представленным в двоичном виде. При использовании данного оператора задается фиксированная вероятность того, что ген потомка будет мутирован. Если текущий ген равен 0, то при мутации он изменяется на значение 1 и наоборот.

Рассмотрим оператор «плотность мутации». При использовании данного оператора задается не только вероятность того, что данный потомок участвует в мутации, но также и вероятности мутаций для каждого гена по отдельности. Значение этих вероятностей выбираются такими, чтобы количество мутировавших генов состояло от 1% до 10%.

При решении задач оптимизации, когда особь представляет собой набор генов $a_1,\ldots,a_k$ динамической длины, то в качестве мутаций можно использовать операторы присоединения, удаления, вставки и обмена.

Примером такой задачи может являться оптимизация распределения студентов по группам (когда у группы нет фиксированного количества свободных мест).

Применение оператора присоединения подразумевает присоединение к генам потомка подстроки *s* содержащей в себе один или несколько генов. В этом случае присоединение будет выглядеть следующим образом:

$$a_1, a_2, ..., a_k \\ \downarrow \\ a_1, a_2, ..., a_k, s \tag{2.27}$$

Применение оператора вставки подразумевает включение в последовательность генов потомков подстроки s содержащей в себе один или несколько генов. В этом случае включение будет выглядеть следующим образом:

$$a_1, a_2, ..., a_k \\ \downarrow \\ a_1, a_2, ..., a_{i-1}, s, a_i, ..., a_k \tag{2.28}$$

Применение оператора удаление подразумевает исключение и последовательности генов потомка один или несколько генов. В этом случае удаление будет выглядеть следующим образом:

$$a_1, a_2, ..., a_k \\ \downarrow \\ a_1, a_2, ..., a_i, a_j, ..., a_k \tag{2.29}$$

при этом должно соблюдаться условие $i < j - 1$.

Применение оператора обмена подразумевает изменение последовательности одного или нескольких генов потомка. Обмен выглядит следующим образом:

$$a_1, a_2, ..., a_i, a_j, ..., a_k \\ \downarrow \\ a_1, a_2, ..., a_j, a_i, ..., a_k \tag{2.30}$$

В целом также стоит отметить, что вероятность P_m мутации генов при выполнении генетического алгоритма задается, как привило, намного меньше единицы ($P_m << 1$).

После выполнения скрещивания родителя в популяции формируются новые особи-потомки, при этом родители также являются элементами популяции. Это приводит к тому, что размер популяции становится больше, чем исходное заданное значение. Поэтому перед переходом на новую

итерацию генетического алгоритма текущую популяцию надо проредить. Для этого используется оператор отбора особей в новую популяцию.

К операторам отбора особей в новую популяцию относятся: отбор усечением, отбор вытеснением, элитарный отбор, отбор методом Больцмана (рисунок 2.8).

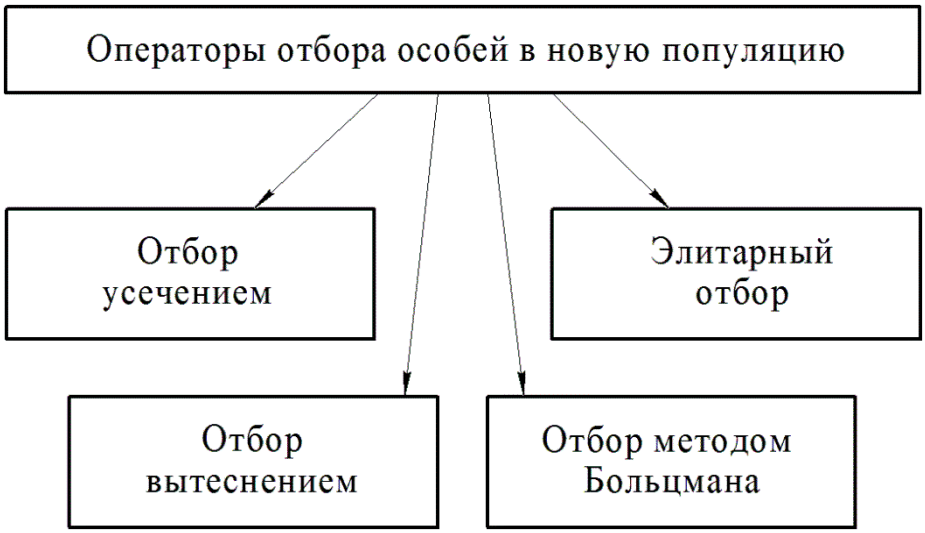

Рисунок 2.8 – Операторы отбора особей в новую популяцию

При формировании новой популяции с использование отбора усечением применяются следующие подходы. В первую очередь необходимо отсортировать текущую популяцию в порядке снижения приспособленности особей. Затем выбирается доля T текущей популяции с лучшими особями, из которых будет производиться отбор в новую популяцию. Теоретически, значение T можно выбирать из диапазона [0; 1], при значении равном 1 все особи текущей популяции участвуют в отборе. На практике оптимальны диапазоном значений параметра T является [0,1; 0,5]. Теперь из выбранной доли лучших особей текущей популяции случайным образом выбирается решение, которое помещается в новую популяцию. Если исходный размер популяции равен значению N, то данная процедура повторяется N раз. При этом возможно попадание дубликатов особи в новую популяцию.

Таким образом, формулы, для расчета новой популяции *Pop_new* размером *N* особей с использование отбора усечением выглядят, так как это показано на (2.31):

$$\begin{aligned} & T \in [0{,}1;\ 0{,}5] \\ & i = 1...N \\ & \begin{cases} k = randomInt(1;\ [(N + numD)\ div\ 1]) \\ Pop\_new_i = Pop\_sort_k \end{cases} \end{aligned} \quad , \qquad (2.31)$$

где *N* – размер популяции, *numD* – количество потомков, полученных на этапе скрещивания родителей, *randomInt* – функция генерирования случайного целого числа в указанном диапазоне, *Pop_sort* – текущая популяция, отсортированная в порядке снижения приспособленности особей

При использовании элитарного отбора для формирования новой популяции применяются следующие подходы. Текущая популяция сортируется в порядке уменьшения приспособленностей особей. 10% новой популяции заполняется лучшими особями текущей популяции, а остальные 90% генерируются заново.

Таким образом, для расчета новой популяции *Pop_new* размером *N* особей с использование элитарного отбора применяются формулы, представленные на (2.32):

$$\begin{aligned} & i = 1...N \\ & Pop\_new_i = \begin{cases} Pop\_sort_i \ \ if \ \ i \le (0.1 \cdot N) \\ generate(x_1, ..., x_n) \ \ if \ \ i > (0.1 \cdot N) \end{cases} \end{aligned} \quad , \qquad (2.32)$$

где *N* – размер популяции, *Pop_sort* – текущая популяция, отсортированная в порядке снижения приспособленности особей, generate () – функции генерирования новой особи случайным образом.

Отбор вытеснением при формировании новой популяции основан на следующих идеях. При данном методе отбора учитывается не только приспособленность особи, но наличие отсутствие в новой популяции особей с близким набором генов.

Метод Больцмана для выбора особей в новую популяцию работает следующим образом. Из текущей популяции выбираются две особи случайным образом. Затем рассчитывается вероятность *P* попадания каждой особи в новую популяцию. В любом случае одна особь из каждой пары попадет в новую популяцию. Вероятность попадания в новую популяцию зависит от приспособленности особи. Математически метод больцмана выглядит так (2.32):

$$
\begin{aligned}
&i = 1...N \\
&j = randomInt(1,\ N + numD) \\
&k = randomInt(1,\ N + numD) \\
&K = random(0,1) \\
&Pop\_new_i = \begin{cases} Pop_j \ \ if \ \ K < \dfrac{1}{1+\exp\left(\dfrac{f_{np}(Pop_i) - f_{np}(Pop_j)}{T(t)}\right)} \\ Pop_k \ \ if \ \ K \geq \dfrac{1}{1+\exp\left(\dfrac{f_{np}(Pop_i) - f_{np}(Pop_j)}{T(t)}\right)}, \end{cases}
\end{aligned}
\qquad (2.32)
$$

где f_{np} – функция приспособленности особи, N – исходный размер популяции, *randomInt()* – функция возвращающая целое число из указанного диапазона, *random()* – функция возвращающая случайное число диапазона [0; 1], *Pop* – текущая популяция особей, размер которой, после скрещивания родителей составляет $N + numD$. T – управляющий параметр, значение которого зависит от номера t текущей итерации генетического алгоритма: чем выше значение t, тем меньше значение функции $T(t)$

2.2 Функции для тестирования работы генетических алгоритмов

Генетический алгоритм является метаэвристическим. Это означает, что под генетическим понимают большое множество алгоритмов, формируемых

путем различного сочетания операторов отбора родителей, скрещивания, мутации, отбора особей в новую популяцию.

Для изучения свойств любой конфигурации генетического алгоритма можно использовать одну или несколько тестовых функций, которые будут представлены ниже. Каждая функция обладает своими особенностями: в некоторых из них несколько глобальных экстремумов; в других – большое количество локальных экстремумов, затрудняющих поиск глобального минимума (или максимума); в других функциях рядом находятся области, в которых значение функции практически не меняются и области с резким изменение значения целевой функции.

Рассмотрим наиболее известные тестовые функции:

- Сферическая функция (2.33):

$$f(x_1,...,x_n)=\sum_{i=1}^{n}x_i^2 \qquad (2.33)$$

При тестировании генетического алгоритма на данной функции решают задачу поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-5{,}12;5{,}12)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Локальные минимумы отсутствуют.

- Гиперэллипсоидная функция с параллельными осями (2.34):

$$f(x_1,...,x_n)=\sum_{i=1}^{n}ix_i^2 \qquad (2.34)$$

Данную функцию используют для проверки у генетического алгоритма способности осуществлять поиск глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-5{,}12;5{,}12)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Данная функция является выпуклой и унимодальной.

- Повернутая гиперэллипсоидная функция (2.35):

$$f(x_1,...,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{i}x_j^2 \qquad (2.35)$$

Гиперэллипсоидную функцию используют для тестирования конфигурации генетического на задаче поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-65,536;65,536)$ Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Эта функция получена поворотом гиперэллипсоидной функции, поэтому она является выпуклой и унимодальной.

- Ступенчатая функция (2.36):

$$f(x_1,...,x_n)=\sum_{i=1}^{n}\left|round(x_i)\right|\,, \qquad (2.36)$$

где round() – функция округляющее число до целой части.

При тестировании генетического алгоритма на данной функции решают задачу поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-5,12;5,12)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$.

- Седло Розенброка (2.37) :

$$f(x_1,...,x_n)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)\,. \qquad (2.37)$$

Данную функцию используют для проверки у генетического алгоритма способности осуществлять поиск глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-2,048;2,048)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Примечательно, что глобальный минимум находится внутри параболической сильно вытянутой поверхности, что осложняет задачу по его поиску.

- Функция с гауссовским распределением случайной величины (2.38):

$$f(x_1,...,x_n)=\sum_{i=1}^{n-1}(x_i^4+gauss(0,1)), \qquad (2.38)$$

где *gauss*() – функция генерирующая случайное число в соответствии с законом нормального распределения, первый параметр функции – математическое ожидание, второй – дисперсия.

Область изменения входных параметров при тестировании генетического алгоритма ограничена диапазоном $x\in(-1,28;1,28)$

- Функция Растригина (2.39):

$$f(x_1,...,x_n)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)) \qquad (2.39)$$

Функция Растригина используют для тестирования конфигурации генетического на задаче поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-5,12;5,12)$ Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Примечательно, что данная функция имеет большое количество локальных минимумов, осложняющих работу генетического алгоритма.

- Функция Швефеля (2.40):

$$f(x_1,...,x_n)=418,9829n+\sum_{i=1}^{n}(-x_i\cdot\sin\sqrt{|x_i|}) \qquad (2.40)$$

При тестировании генетического алгоритма на данной функции решают задачу поиска глобального минимума, который достигается при $x_i=420,9829\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-500;50)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Примечательно, что данная функция имеет дополнительный локальный минимум, осложняющий работу генетического альгоритма.

- Функция Грайвенка (2.41):

$$f(x_1,...,x_n)=1+\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (2.41)$$

Данную функцию используют для проверки у генетического алгоритма способности осуществлять поиск глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-600;600)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$. Примечательно, что у функции есть большое количество распределенных локальных минимумов, осложняющих работу генетического алгоритма.

- Функция Эккли (2.42):

$$f(x_1,...,x_n)=20+e-20\cdot\exp\left(-0,2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) \quad (2.42)$$

Функцию Эккли используют для тестирования конфигурации генетического на задаче поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-30;30)$ Минимальное значение функции $f(x_1,\ldots,x_n)=0$.

- Функция суммы различных степеней (2.43):

$$f(x_1,...,x_n)=\sum_{i=1}^{n}\left|x_i\right|^{i+1} \quad (2.43)$$

При тестировании генетического алгоритма на данной функции решают задачу поиска глобального минимума, который достигается при $x_i=0\,(i=1...n)$. Область изменения входных параметров при тестировании ограничена диапазоном $x\in(-1;1)$. Минимальное значение функции $f(x_1,\ldots,x_n)=0$.

- Функция Михалевича (2.44):

$$f(x_1,...,x_n) = -\sum_{i=1}^{n}\left(\sin(x_i)\cdot\sin^{20}\left(\frac{i}{\pi}x_i^2\right)\right) \qquad (2.44)$$

Данную функцию используют для проверки у генетического алгоритма способности осуществлять поиск глобального минимума. Минимальное значение функции при n=5: $f(x_1,\ldots,x_n)$=–4,69, при n=10: $f(x_1,\ldots,x_n)$=–9,66. Область изменения входных параметров при тестировании ограничена диапазоном $x \in [0;\pi]$. Примечательно, что рельеф функции представлен набором оврагов и плато, что осложняет работу генетического алгоритма.

- Функция Бранинса (2.45):

$$f(x_1,x_2) = \left(x_2 - \frac{5{,}1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos(x_1) + 10 \qquad (2.45)$$

Функцию Бранинса используют для тестирования конфигурации генетического на задаче поиска нескольких глобальных минимумов, которые достигаются при $(x_1,x_2) = (-\pi, 12.275)$, $(x_1,x_2) = (\pi, 2.275)$, $(x_1,x_2) = (9.42478, 2.475)$. Область изменения входных параметров при тестировании ограничена диапазоном $x_1 \in [-5;10], x_2 \in [0;15]$. Минимальное значение функции $f(x_1, x_2)$=0,397887. От генетического алгоритма требуется найти все 3 глобальных минимума.

- Функция Изома (2.46):

$$f(x_1,x_2) = -\cos(x_1)\cos(x_2)\exp\ -(x_1-\pi)^2 + (x_2-\pi)^2 \qquad (2.46)$$

При тестировании генетического алгоритма на данной функции решают задачу поиска глобального минимума, который достигается при $x_1 = -\pi, x_2 = \pi$. Область изменения входных параметров при тестировании ограничена диапазоном $x_1 \in [-100;100], x_2 \in [-100;100]$. Минимальное значение функции $f(x_1, x_2)$=–1.

Примечательно, что у данной унимодальной функции минимум находится в малой области по сравнению размером поискового пространства. Это усложняет работу генетического алгоритма.

- Функция Голдстейна (2.47):

$$f(x_1, x_2) = [1 + (1 + x_1 + x_2)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times \\ \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \quad (2.47)$$

Данную функцию используют для проверки у генетического алгоритма способности осуществлять поиск глобального минимума, который достигается при $x_1 = 0; x_2 = -1;$. Область изменения входных параметров при тестировании ограничена диапазоном $x_1 \in [-2;2], x_2 \in [-2;2]$. Минимальное значение функции $f(x_1, x_2)$=3.

- Функция шестигорбая (2.48):

$$f(x_1, x_2) = \left(4 - 2{,}1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 - 4x_2^2) \quad (2.48)$$

Эту функцию используют для тестирования конфигурации генетического на задаче поиска нескольких глобальных минимумов, которые достигаются при $(x_1, x_2) = (-0.0898, 0.7126)$ и $(x_1, x_2) = (0,0898, -0,7126)$. Область изменения входных параметров при тестировании ограничена диапазоном $x_1 \in [-3;3], x_2 \in [-2;2]$. Минимальное значение функции $f(x_1, x_2)$=–1,0316. От генетического алгоритма требуется найти оба глобальных минимума.

2.3 Результаты анализа принципов работы генетического алгоритма

Анализ математического аппарата генетических алгоритмов и принципов их построения позволил выявить следующие достоинства эволюционных вычислений при решении задач оптимизации:

- Применение генетических алгоритмов возможно в условиях отсутствия информации о свойствах целевой функции. Т.е. не нужно ничего знать, например, о дифференцируемости, непрерывности функции чтобы применять генотипические алгоритмы.

- Наличие в решаемой задаче большого количества локальных оптимумов не оказывает значимого влияния на успешность работы генетического алгоритма.
- Генетические алгоритмы обладают свойством масштабируемости и могут успешно решать крупные задачи оптимизации с большим количеством ограничений и входных параметров.
- Генетические алгоритмы с точки зрения проектирования программного обеспечения просты в реализации.
- Генетические алгоритмы с точки зрения моделирования обладают простым математическим аппаратом.
- Генетические алгоритмы могут быть использованы для решения широкого класса задач из различных областей науки и техники.
- Важной особенностью генетических алгоритмов является возможность их использования в динамических задачах с изменяющейся средой, т.е. когда форма целевой функции изменяется со временем. В этом случае генетические алгоритмы способны корректировать изменение входных параметров для поддержания критерия оптимальности.

Помимо достоинств, у генетических алгоритмов можно выделить следующие недостатки, ограничивающих сферы их применения:

- Генетические алгоритмы не предназначены для поиска точных решений.
- Если расчет значения целевой функции при заданных значениях входных параметров требует длительного времени, то в этом случае неэффективно использовать генетические алгоритмы для решения задачи оптимизации.

- Применение генетических алгоритмов неэффективно в том случае, если целевая функция является изолированной (пример функции представлен на рисунке 2.9).

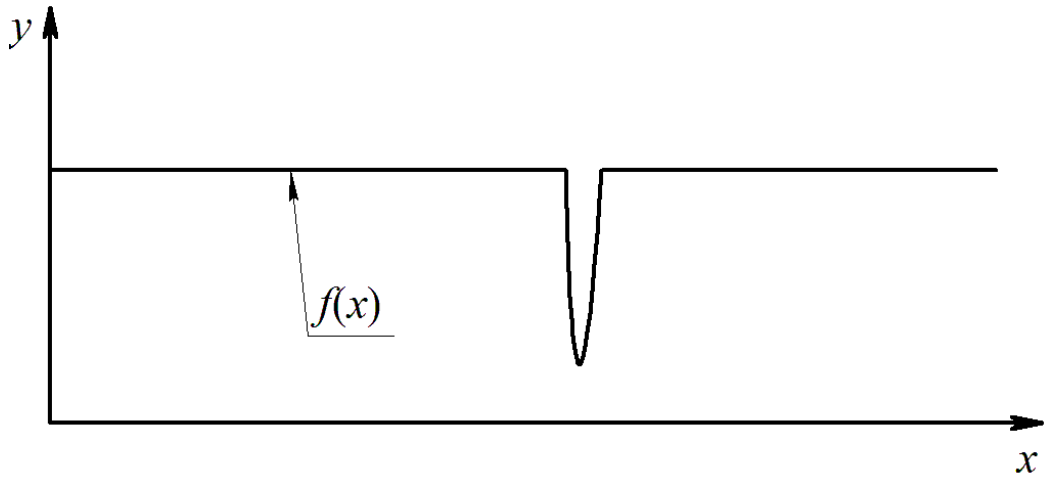

Рисунок 2.9 – Пример изолированной функции $f(x)$

Функция называется изолированной, если она не предоставляет алгоритму оптимизации информацию о направлении поиска экстремумов. В случае применения генетического алгоритма это означает, глобальный экстремум может быть найден только за счет случайного падания в него особи на этапе генерации популяции.

- Применение генетических алгоритмов осложнено в условиях зашумленной целевой функции (пример функции представлен на рисунке 2.10).

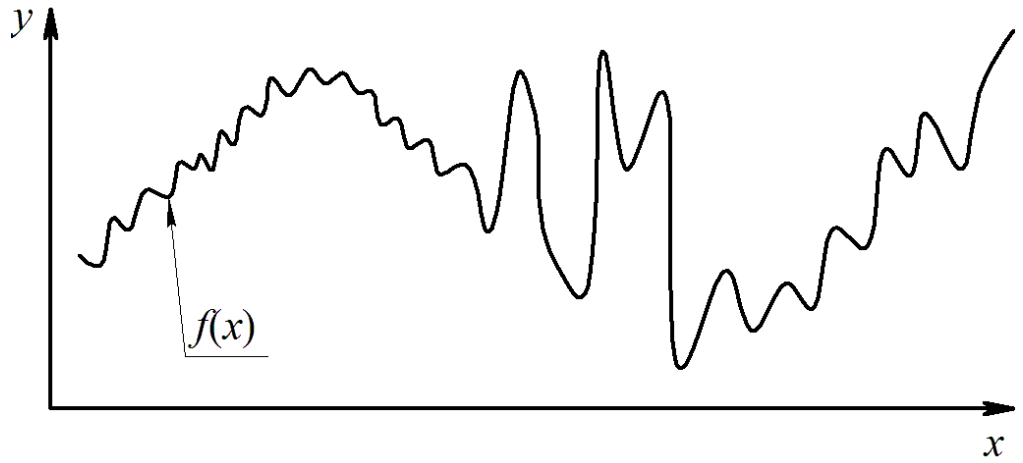

Рисунок 2.10 – Пример зашумленной функции *f*(*x*)

Наличие шума приводит к разбрасыванию значений приспособленности шим, что существенно замедляет поиск решения оптимизационной задачи.

2.4 Синтез алгоритма кластеризации на основе эволюционных вычислений

Опишем предложенные в рамках исследования оригинальные подходы по использованию генетического алгоритма для кластеризации данных.

Особями O_i в рамках решения задачи кластеризации будут являться параметры центров кластеров. Если в результате кластеризации требуется получить кластерную структуру с количеством кластеров равно k, а размерность признакового пространства объектов кластеризации равна n, то в общем виде особь O можно представить, как:

$$O = (C_1, C_2, ..., C_k) = ((x_1, x_2, ..., x_n)_1, (x_1, x_2, ..., x_n)_2, ..., (x_1, x_2, ..., x_n)_k) \quad (2.49)$$

Пример, что размер популяции составляет N особей, тогда популяция Pop в начале итерации t будет иметь вид:

$$Pop(t) = (O_1, O_2, ..., O_N) \quad (2.50)$$

На первом этапе работы генетического алгоритма случайным образом генерируются N вариантов расположения центров кластеров.

```
GeneratePopulation(N, k, n)
 1  pop ← ⟨ ⟩
 2  for i=1 to N do
 3      for j=1 to k do
 4          for q=1 to n do
 5              x_q ← random(x_q)
 6              Add(C_j, x_q)
 7          Add(O_i, C_j)
 8      Add(pop, O_i)
 9  return pop
```

Рисунок 2.11 – Псевдокод функции для генерирования начальной популяции

На втором этапе работы генетического алгоритма проверяется условие остановки поиска решения. В качестве такого условия предложено проверять – удалось ли найти более оптимальное расположение центров кластеров $(C_1, C_2, ..., C_k)$ за последние несколько итераций (tt – количество последних итераций).

```
CheckEnd(BestO(iter), BestO(iter − 1), ..., BestO(iter − tt − 1))
 1  check ← true
 2  for i=1 to tt − 1 do
 3      if BestO(iter) ≠ BestO(iter − 1) then
 4          check ← false
 5          break
 6  return check
```

Рисунок 2.12 – Псевдокод функции для остановки работы генетического алгоритма

Оптимальным является то расположение центров кластеров, при котором удается добиться максимальной плотности кластеров (чем меньше значение функции f_{np}, тем выше суммарная плотность кластеров) (2.51):

$$f_{np}(C_1,...,C_m)=\sum_{i=1}^{m}\sum_{M_m\in Cluster_i} d(M_m,C_i)^2 \to \min \qquad (2.51)$$

Если критерий остановки генетического алгоритма сработал, то из текущей популяции возвращается вариант расположения центров кластеров с наименьшим значением функции $f_{np}(C_1,...,C_m)$.

На этапе выбора родителей P1 и P2 для скрещивания применяется панмиксия (2.52):

$$\begin{cases} P_1 \leftarrow random(O_1,...,O_N) \\ P_2 \leftarrow random(O_1,...,O_N) \\ P_1 \neq P_2 \end{cases} \qquad (2.52)$$

На этапе скрещивания родителей, для получения потомков $D1$ и $D2$ применяется линейная рекомбинация с заданием случайного значения множителя α из диапазона [-0.25; 1.25]:

$$\begin{cases} D1_i = P1_i + \alpha_1 \cdot (P2_i - P1_i) \\ D2_i = P1_i + \alpha_2 \cdot (P2_i - P1_i) \end{cases} \qquad (2.53)$$

Для полученных в результате скрещивания потомков с вероятностью 10% применяется мутация вещественных особей. При этом значения генов потомка меняются по формуле (2.54):

$$\begin{aligned} & j = 1...n \\ & \delta_j \leftarrow \text{random}([0.5 \cdot \min(x_j)],[0.5 \cdot \max(x_j)]) \\ & \alpha_j \leftarrow \text{random}(0,1) \\ & D_j = \begin{cases} D_j - \delta \ \ if \ \alpha_j \le 0.5 \\ D_j + \delta \ \ if \ \alpha_j > 0.5 \end{cases} \end{aligned} \quad , \qquad (2.54)$$

где $\min(x_j)$ и $\max(x_j)$ - минимальное и максимальное значения области определения параметра x_j.

DoIterationGA(pop(*iter*), N, n, *NumDescendant*)

1 $new\_pop \leftarrow \langle\ \rangle$
2 **for** i=1 to *NumDescendant* do
3 $m1 \leftarrow$ randomInt(1, N)
4 $m2 \leftarrow$ randomInt(1, N)
5 $P1 \leftarrow$ extract($Pop, m1$)
6 $P2 \leftarrow$ extract($Pop, m2$)
7 $\alpha \leftarrow$ random($-0.25, 1.25$)
8 **for** j=1 to n do
9 $D_j \leftarrow P1_j + \alpha \cdot (P2_j - P1_j)$
10 **if** random$(0,1) \le 0.1$ **then**
11 **for** j=1 to n do
13 $\delta \leftarrow$ random$([0.5 \cdot \min(x_j)], [0.5 \cdot \max(x_j)])$
14 $\alpha \leftarrow$ random$(0,1)$
15 **if** $\alpha \le 0.5$ **then** $D_j \leftarrow D_j - \delta$
16 **else** $D_j \leftarrow D_j + \delta$
17 Add(pop, D)
18 Sort(pop) / / сортировка индексов в порядке воздастания E
19 **for** i=1 to N do
20 **if** $i \le (N$ div $0.1)$ **then** $New\_pop_i \leftarrow pop_i$
21 **else** $New\_pop_i \leftarrow generateRandom(x_1, ..., x_n)$
22 $iter \leftarrow iter + 1$
23 **return** new_pop

Рисунок 2.13 – Псевдокод функции, выполняющей операторы выбора родителей, скрещивание, мутацию и отбор особей в новую популяцию

Переход особей в новую популяцию осуществляется с применением элитарного отбора, при котором 10% вариантов расположения центров кластеров, обеспечивающих наибольшее значение f_{np} переносится в новую популяцию, а остальные 90% процентов популяции генерируется случайным образом заново (2.55):

$$i = 1...N$$

$$Pop\_new_i = \begin{cases} Pop\_sort_i \ \ if \ \ i \le (0.1 \cdot N) \\ generate(x_1, ..., x_n) \ \ if \ \ i > (0.1 \cdot N) \end{cases}, \qquad (2.55)$$

где N – размер популяции, *Pop_sort* – текущая популяция, отсортированная в порядке снижения приспособленности особей, *generate* () – функции генерирования нового расположения кластеров случайным образом.

Псевдокод функции, выполняющей операторы выбора родителей, скрещивание, мутацию и отбор особей в новую популяцию для каждой итерации представлен на рисунке 2.13

При окончании работы генетического алгоритма в качестве решения предъявляется расположение кластеров с наименьшим значением $f_{np}(C_1,...,C_m)$.

3 МОДЕЛИРОВАНИЕ АЛГОРИТМА КЛАСТЕРИЗАЦИИ

3.1 Программное моделирование предложенного алгоритма кластеризации

Для моделирования генетического алгоритма, с учетом предложенных подходов при решении задач кластеризации данных, было разработано программное обеспечение, обладающее следующими функциональными особенностями:

- программное моделирование процесса поиска оптимальной кластерной структуры с использованием эволюционных вычислений (программная реализация предложенных подходов);
- программное моделирование алгоритма k-means (для сравнения с результатами работы генетического алгоритма);
- наличие графического интерфейса;
- возможность задания следующих параметров генетического алгоритма: размер популяции (population size), количество пар, участвующих в скрещивании (pair amount), процент лучших особей, участвующих в образовании пар (percent for cross), процент лучших особей, переходящих в новую популяцию (percent for new), максимальное отклонение вещественного гена при мутации (parameter ‘d’), вероятность мутации гена (mutation chance), вероятность участия мутации особи (mutation power);
- возможность задания следующих параметров объектов кластеризации: количество объектов (entities), количество кластеров (clusters);
- визуализация кластерной структуры на каждой итерации кластеризации данных (цветом отмечает принадлежность объектов к кластерам, крестами показаны центры кластеров);
- построение графиков изменения ошибки кластеризации в зависимости от номера итерации алгоритма k-means и генетического алгоритма.

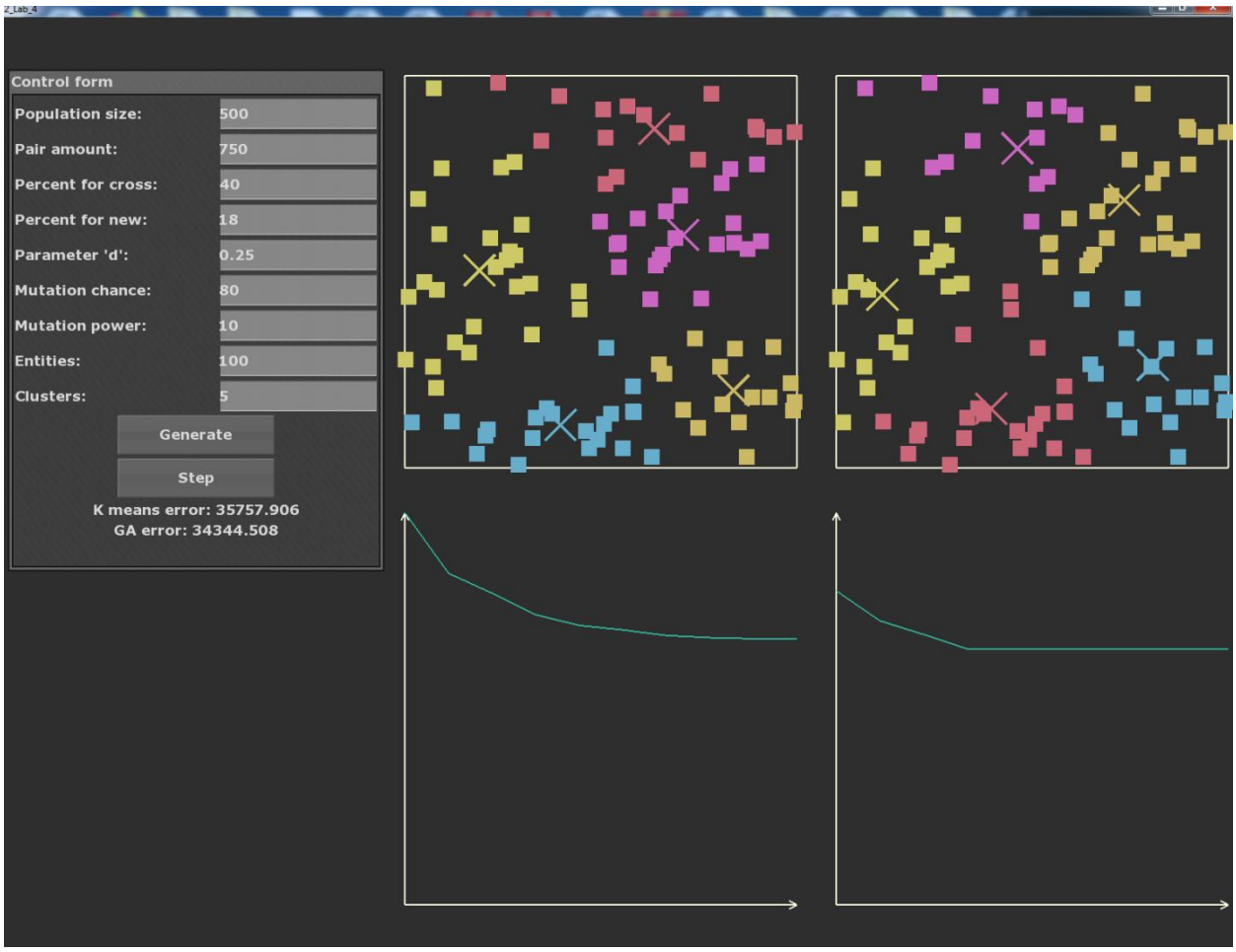

Рисунок 3.1 – Основное окно программы для моделирования эволюционных вычислений при решении задачи кластеризации

Главное окно программы представлено на рисунке 3.1. Оно содержит в себе: панель для задания параметров генетического алгоритма и параметров объектов кластеризации, две декартовых системы координат для визуализации процесса кластеризации на каждой итерации (левая – для алгоритма k-means, правая – для генетического алгоритма); два графика изменения ошибки кластеризации E в зависимости от итерации работы алгоритмов (левый график для алгоритма k-means, правый – для генетического алгоритма).

3.2 Вычислительный эксперимент

Вычислительный эксперимент проводится для решения следующих задач:

- проверка возможности использования генетических алгоритмов для решения задачи кластеризации данных;
- оценка динамики поиска оптимальной кластерной структуры с использованием генетического алгоритма;
- сравнение скорости (количество итераций) процесса поиска оптимальной кластерной структуры с использование алгоритма k-means и генетического алгоритма;
- сравнение плотности кластерной структуры (посредством ошибки кластеризации E) двух алгоритмов k-means и генетического алгоритма.

Алгоритм проведение вычислительного эксперимента состоит из следующих шагов:

- Генерация объектов кластеризации, состоящих из двух параметров x_1 и x_2. Такая конфигурация параметров объектов кластеризации позволит их визуализировать в виде точек в декартовом пространстве.
- Случайным образом выбираются центры кластеров, которые в процессе кластеризации будут уточняться с использование алгоритма k-means и с использованием генетического алгоритма.
- На каждой итерации работы алгоритмов их кластерные структуры будут оцениваться с помощью показателя E – ошибки кластеризации (формула 1.5).
- На каждой итерации рассчитывается отношение $(E_k–E_{ga})/E_k$, показывающее насколько % в данной итерации кластерная структура, полученная генетическим алгоритмом, лучше структуры, полученной алгоритмом k-means.

Результаты проведения вычислительных экспериментов представлены на рисунках 3.2 – 3.8.

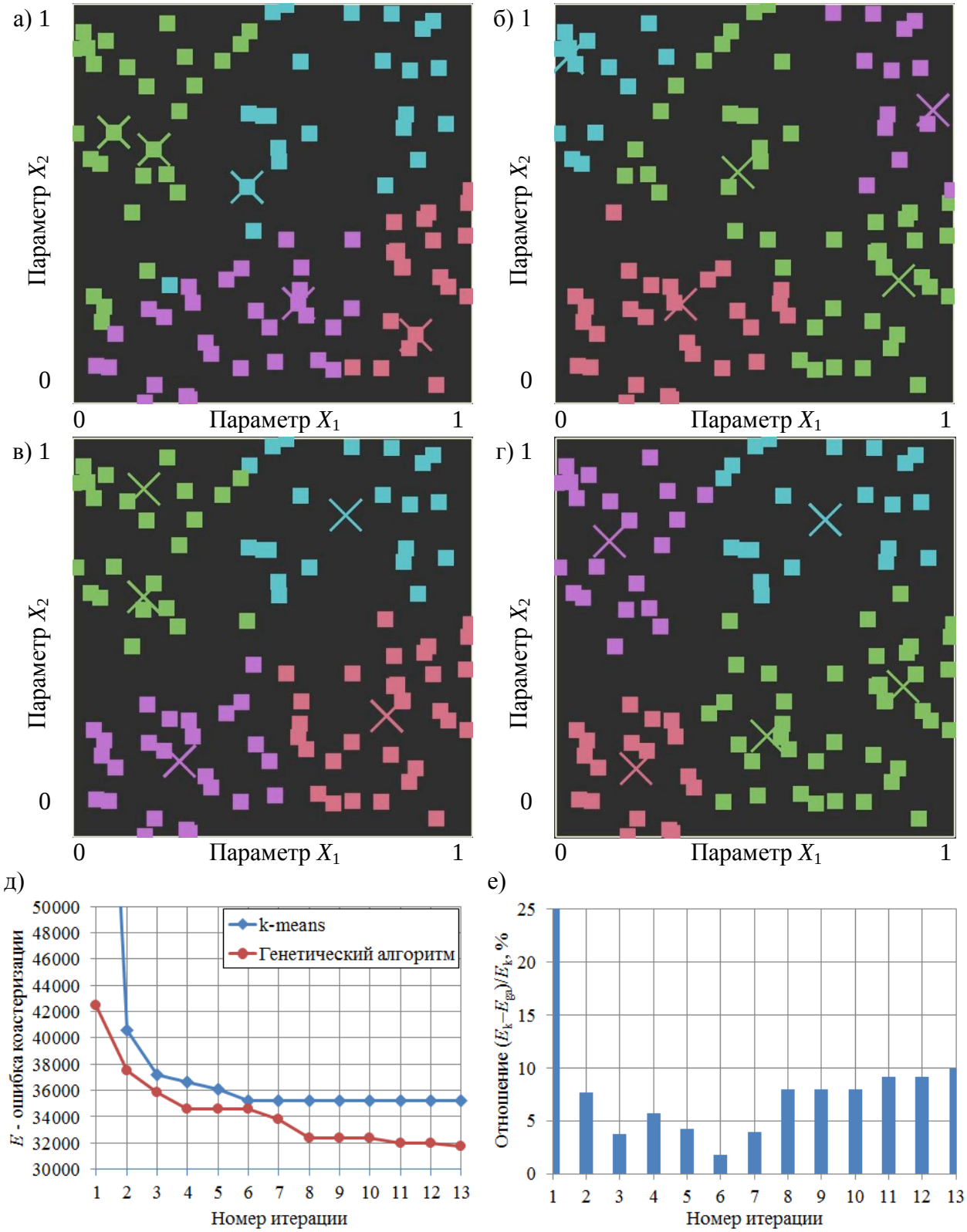

Рисунок 3.2 – Результаты эксперимента №1: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации (д и е)

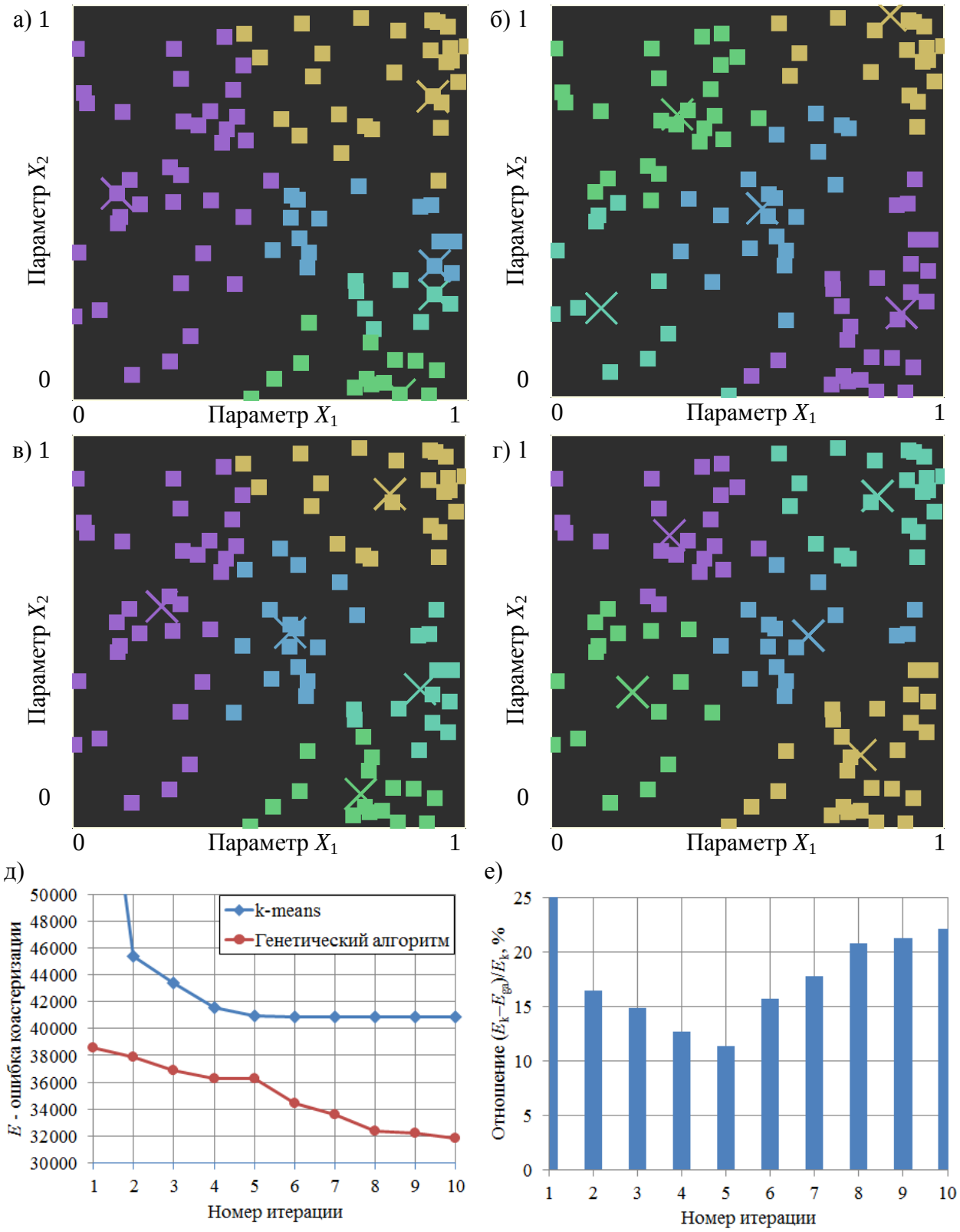

Рисунок 3.3 – Результаты эксперимента №2: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации (д и е)

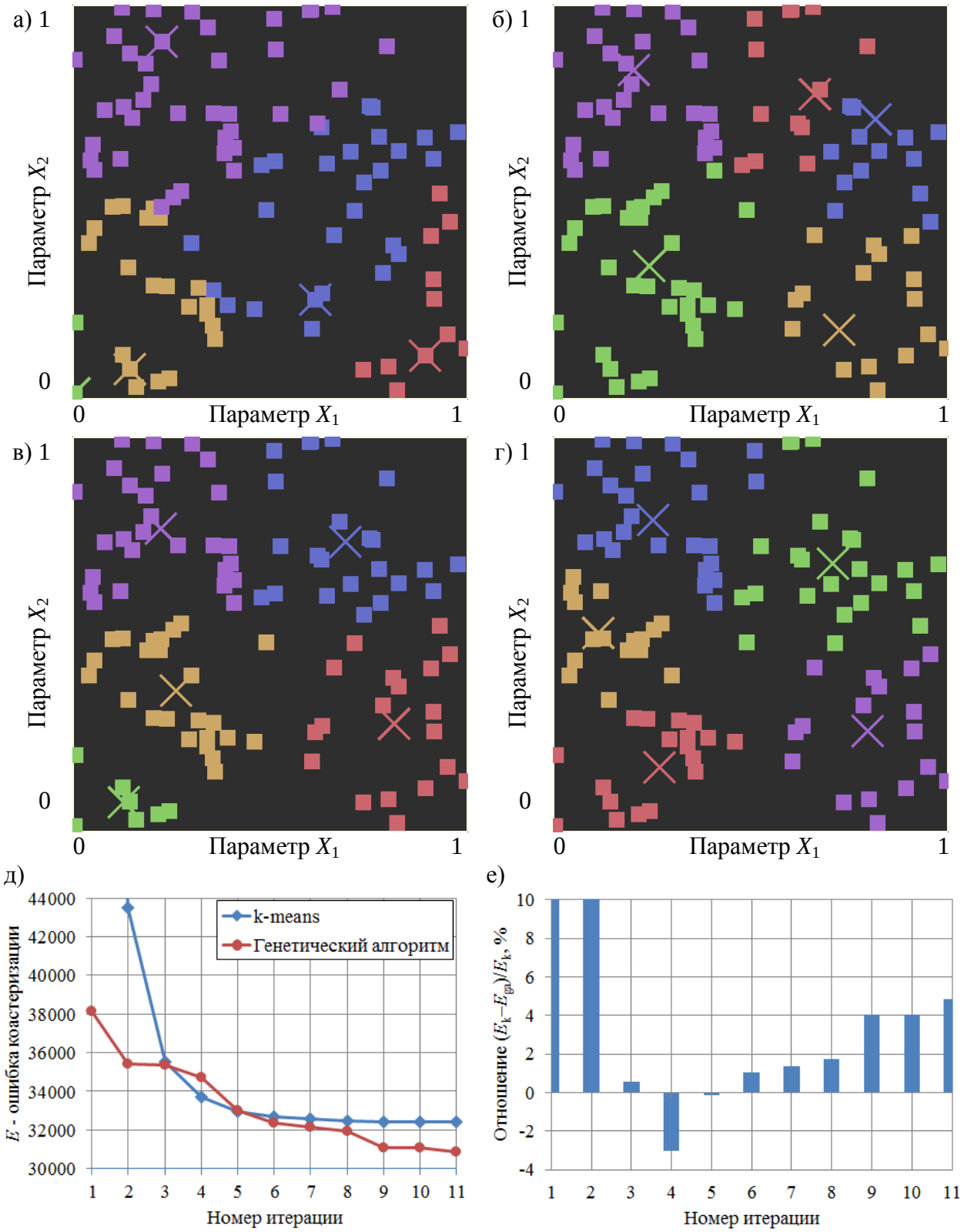

Рисунок 3.4 – Результаты эксперимента №3: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации (д и е)

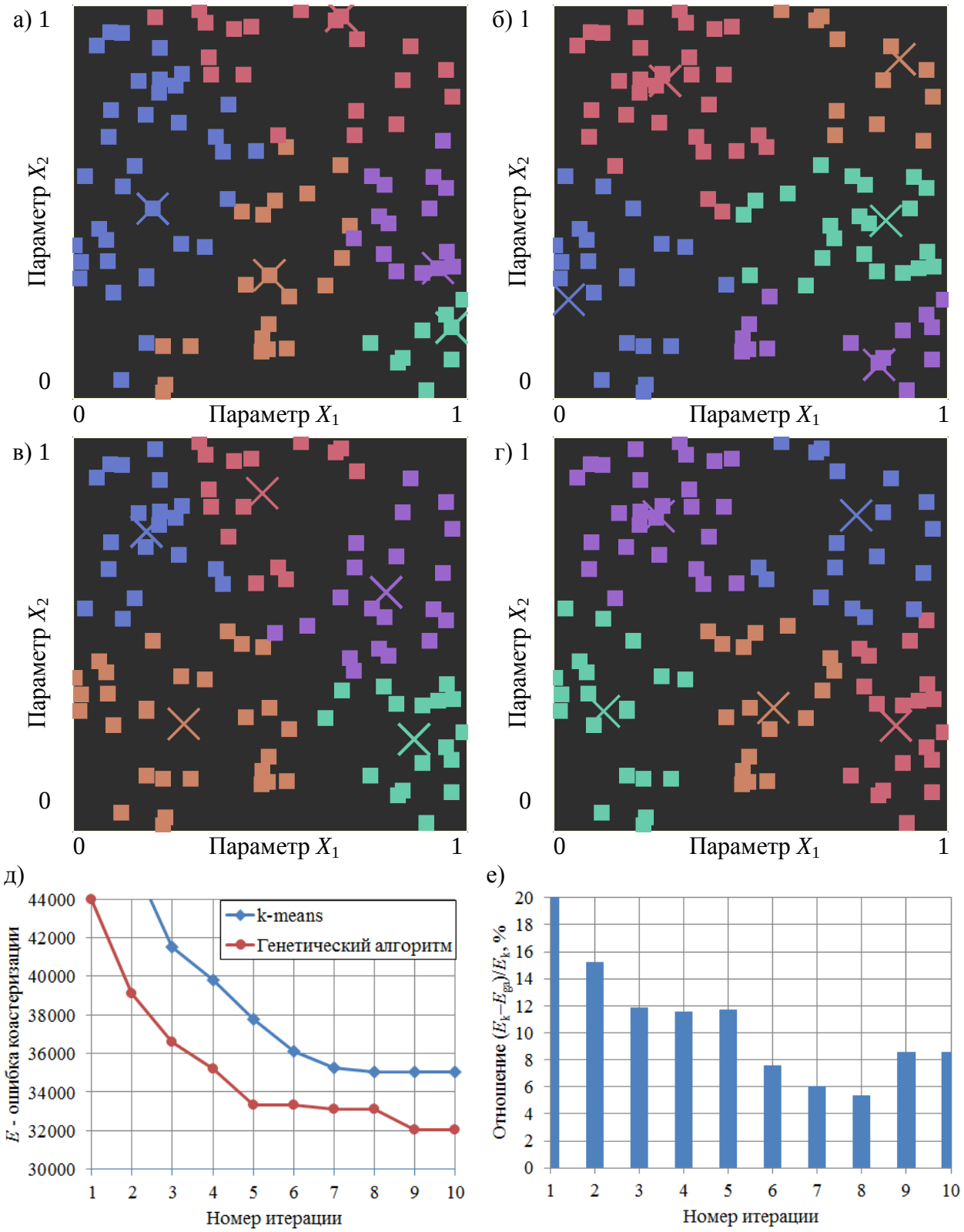

Рисунок 3.5 – Результаты эксперимента №4: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации (д и е)

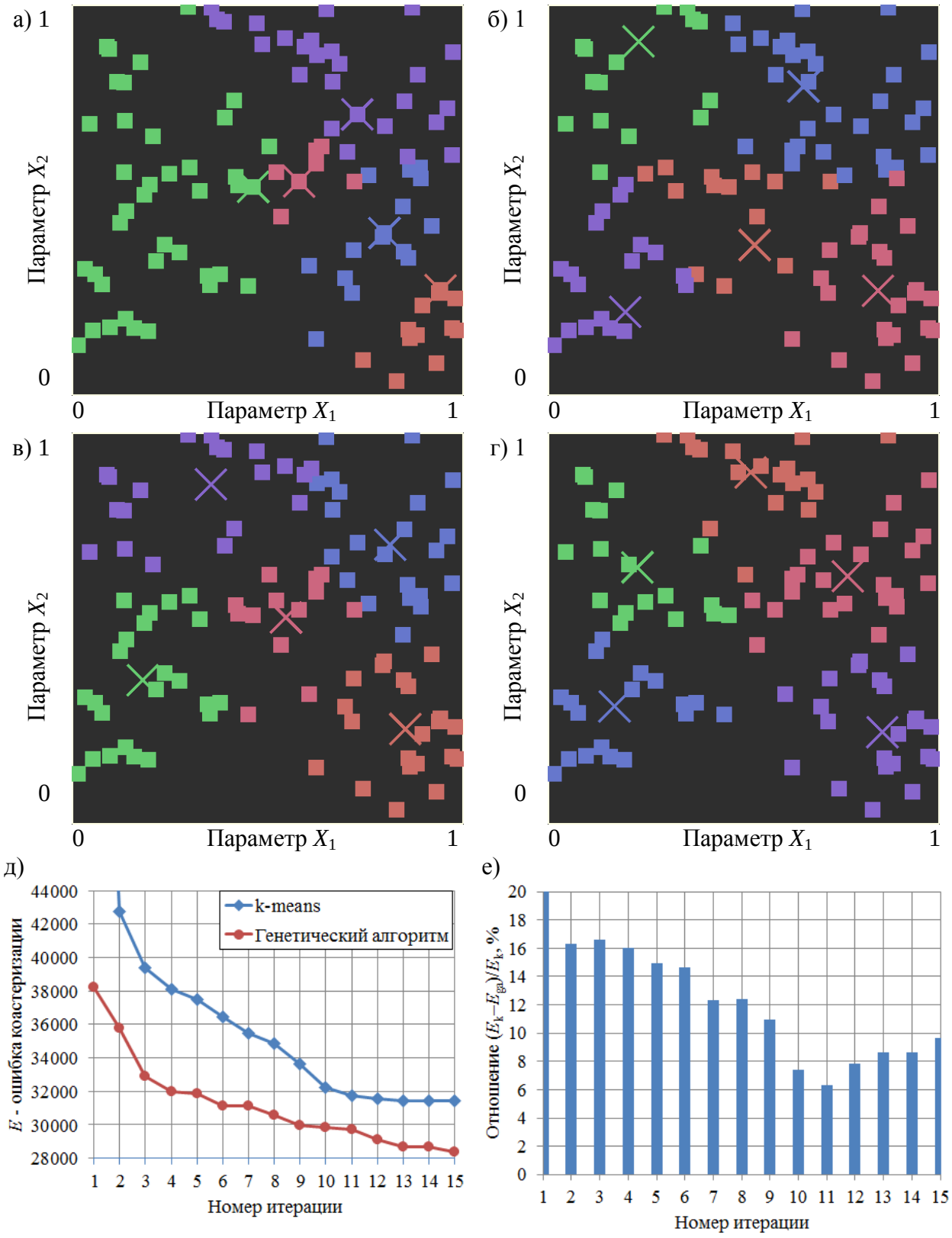

Рисунок 3.6 – Результаты эксперимента №5: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации (д и е)

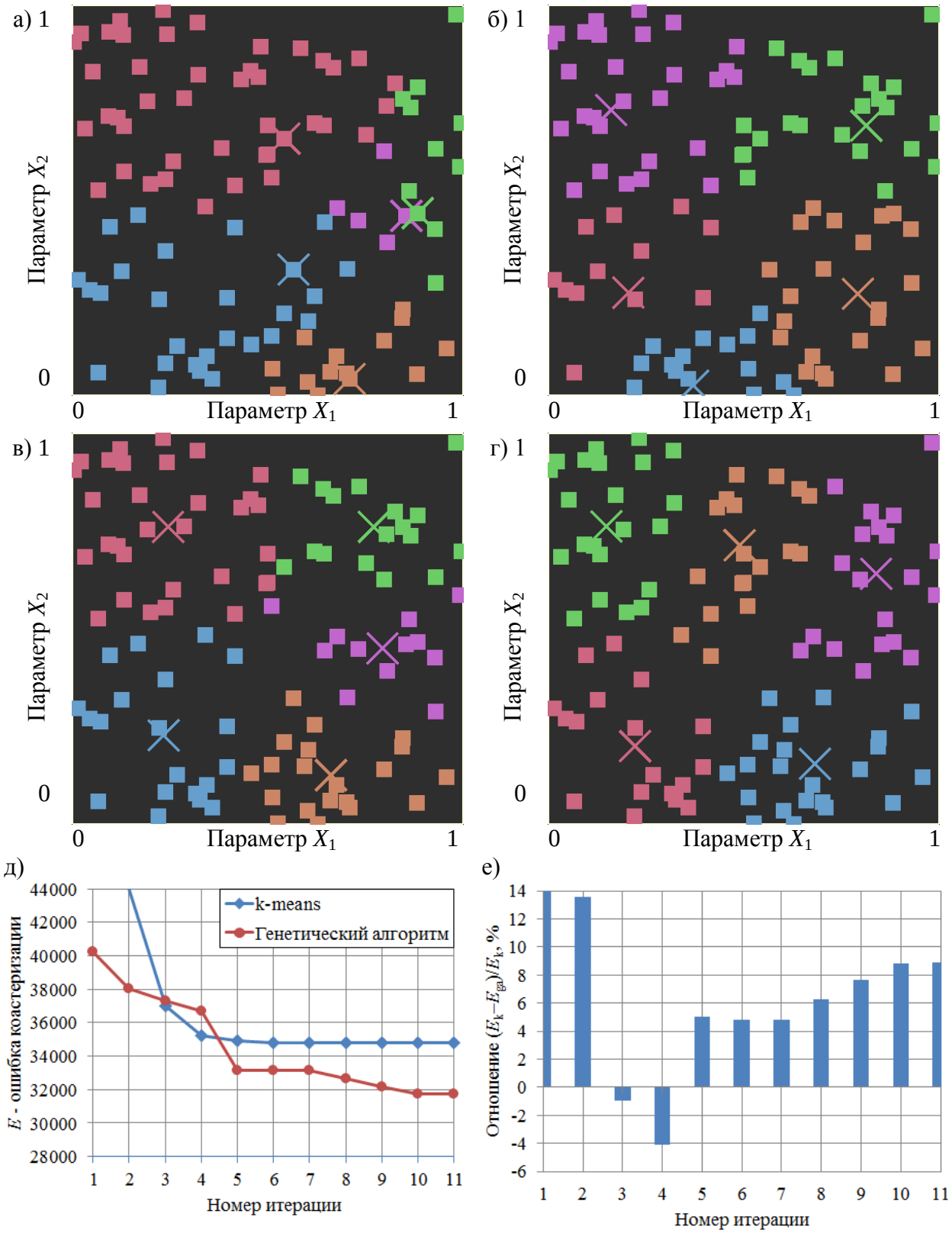

Рисунок 3.7 – Результаты эксперимента №6: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации

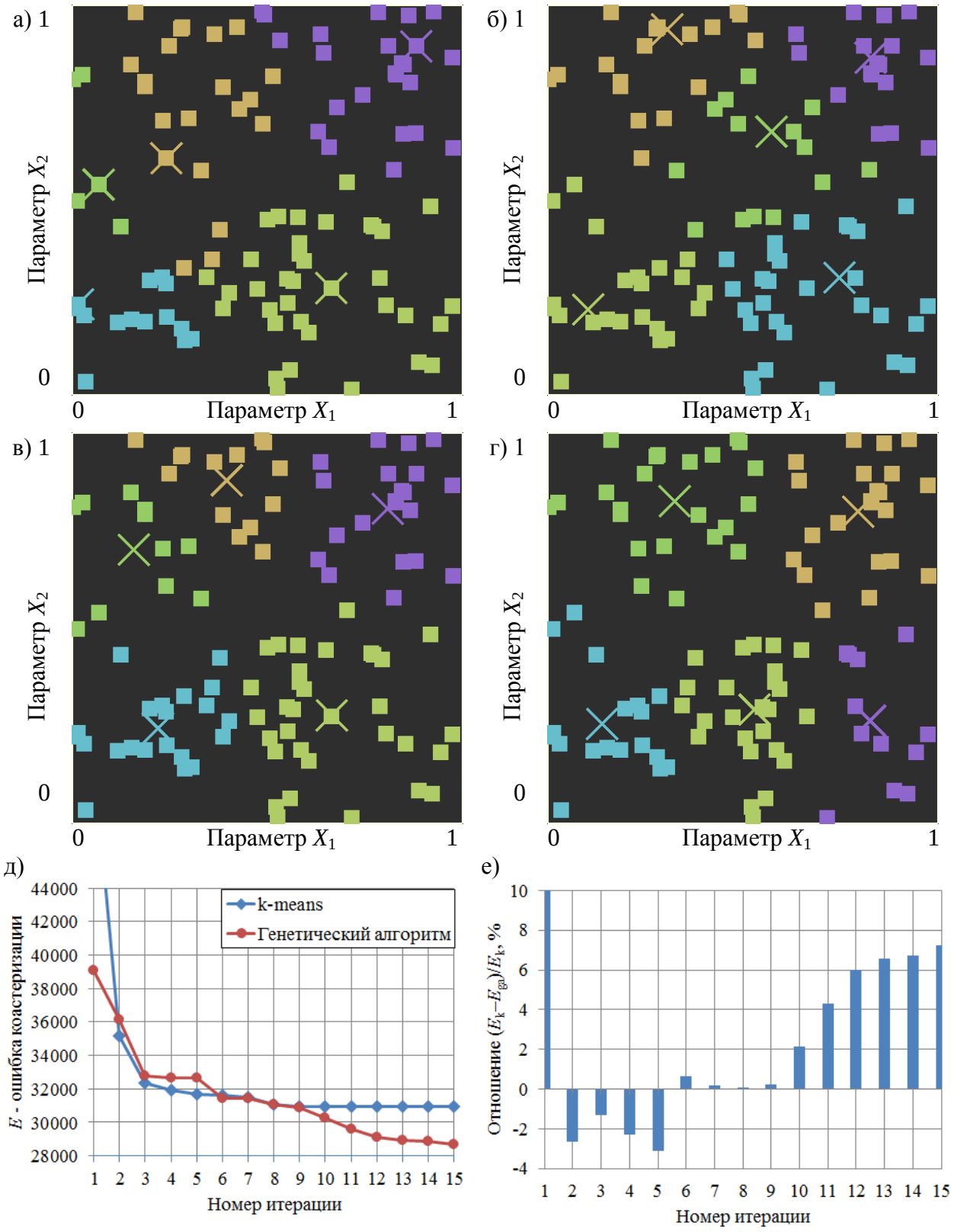

Рисунок 3.8 – Результаты эксперимента №7: исходная кластерная структура (а – алгоритм k-means, б – генетический алгоритм), конечная кластерная структура (в – алгоритм k-means, г – генетический алгоритм), сравнение динамики изменения ошибки E в процессе кластеризации

3.3 Обсуждение результатов

Представленные результаты вычислительного эксперимента (рисунки 3.2-3.8) позволяют сделать следующие выводы:

1. Как видно из графиков изменения ошибки кластеризации *E* (рисунки 3.2-3.8, д, красная линия) генетический алгоритм, основанный на предложенных в магистерской диссертации подходах, может успешно решать задачу в подборе кластерной структуры для исходных данных. Об этом свидетельствует тот факт, что ошибка кластеризации E на каждой итерации снижается.

2. При сравнении динамики изменения ошибки кластеризации k-means и генетического алгоритма (рисунки 3.2-3.8, е) видно, что во всех случаях с помощью генетического алгоритма удается получить более плотную кластерную структуру (с меньшим значением ошибки *E*). При этом у решения, получаемого с помощью генетического алгоритма значение E в среднем меньше на 10-15%.

3. Даже при незначительном различии по значению *E* двух решений представляемые в них кластерные структуры могу отличаться значительно. Это видно при сравнении рисунков 3.2в-3.8в с рисунками 3.2г-3.8г.

4. Генетический алгоритм способен самостоятельно выходить из локального решения и продолжать поиск оптимальной кластерной структуры. Это можно увидеть на рисунках рисунки 3.2-3.8д (красная линия), когда в середине поиска оптимального решение график на несколько итераций остается на текущем уровне, но затем снова продолжает снижаться. Одновременно с этим алгоритм k-means, поиск лучшей кластерной структуры сразу же после попадания в локальное решение.

5. Генетическому алгоритму требуется в среднем 2 раза больше итераций, чем алгоритму k-means для того, чтобы найти оптимальную кластерную структуру.

ЗАКЛЮЧЕНИЕ

1. В обзор литературных источников по теме исследования позволил установить, что при решении практических задач из различных областей науки техники требуется выполнение кластеризации данных (например, при сегментации изображений кластеризации подвергаются его пиксели). По этой причине актуальной задачей остается развитие методов кластеризации данных.

2. Генетические алгоритмы предназначены для решения задач оптимизации. В исследовании предложены подходы, позволяющие представить задачу кластеризации, как задачу оптимизации кластерной структуры. Данные подходы включают в себя использования в качестве целевой функции сумму квадратов расстояний от объектов кластеризации до центра ближайшего кластера (при этом решается задача минимизации значения функции). В качестве входных параметров функции предложено использовать координаты центров кластеров. Данные подходы позволяют применять различные методы оптимизации, но уже для кластеризации данных.

3. Проведено исследования математического аппарата генетических алгоритмов, по результатам которого был сделан вывод о возможности его использования при кластеризации данных (раздел 2.3). Также произведен синтез конфигурации генетического алгоритма, направленный на решения задачи кластеризации данных (раздел 2.4).

4. Разработано программное обеспечение для моделирования генетических алгоритмов и для кластеризации данных с использованием предложенных подходов (раздел 3.1).

5. Для апробации предложенной конфигурации генетического алгоритма в рамках решения задачи кластеризации данных были проведены вычислительные эксперименты (раздел 3.2). Результаты экспериментов (см. раздел 3.3) показывают состоятельность предложенных подходов.

Основные результаты работы были доложены на V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук».

Также в рамках написания магистерской диссертации были опубликованы следующие статьи:

1. Перспективы использования эволюционных вычислений при кластеризации данных [41].

2. Кластеризация изображения на основе оценки геометрической формы объектов [42].

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Kanzawa, Y. On Bezdek-Type Possibilistic Clustering for Spherical Data, Its Kernelization, and Spectral Clustering [Text] / Yuchi Kanzawa // Approach International Conference on Modeling Decisions for Artificial Intelligence – 13th International Conference, MDAI 2016, Sant Julià de Lòria, Andorra, September 19-21, 2016. Proceedings: Modeling Decisions for Artificial Intelligence. – Springer International Publishing Switzerland 2016. – pp. 178-190

2. Bao, L. Document Clustering Based on Spectral Clustering and Non-negative Matrix Factorization [Text] / Lei Bao, Sheng Tang, Jintao Li, Yongdong Zhang, Wei-ping Ye // International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems – 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2008 Wrocław, Poland, June 18-20, 2008. Proceedings: New Frontiers in Applied Artificial Intelligence. – Springer-Verlag Berlin Heidelberg 2008. – pp. 149-158

3. Oshio, S. A Deterministic Clustering Framework in MMMs-Induced Fuzzy Co-clustering [Text] / Shunnya Oshio, Katsuhiro Honda, Seiki Ubukata, Akira Notsu // International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making – 4th International Symposium, IUKM 2015, Nha Trang, Vietnam, October 15-17, 2015, Proceedings: Integrated Uncertainty in Knowledge Modelling and Decision Making. – Springer International Publishing Switzerland 2015. – pp. 204-213

4. Chaimontree, S. Best Clustering Configuration Metrics: Towards Multiagent Based Clustering [Text] / Santhana Chaimontree, Katie Atkinson, Frans Coenen // International Conference on Advanced Data Mining and Applications – 6th International Conference, ADMA 2010, Chongqing, China, November 19-21, 2010, Proceedings, Part I: Advanced Data Mining and Applications. – Springer-Verlag Berlin Heidelberg 2010. – pp. 48-59

5. Yan, Q. Similarity Matrix Construction Methods in Sparse Subspace Clustering Algorithm for Hyperspectral Imagery Clustering [Text] / Qing Yan, Yun Ding, Jing-Jing Zhang, Li-Na Xun, Chun-Hou Zheng // International Conference on Intelligent Computing – 13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part I: Intelligent Computing Theories and Application. – Springer International Publishing AG 2017. – pp. 695-700

6. Ryazanov, V. Clustering of Incomplete Data and Evaluation of Clustering Quality [Text] / Vladimir V. Ryazanov // Iberoamerican Congress on Pattern Recognition – 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. – Springer-Verlag Berlin Heidelberg 2012. – pp. 146-153

7. LaPlante, F. A Heuristic Automatic Clustering Method Based on Hierarchical Clustering [Text] / François LaPlante, Nabil Belacel, Mustapha Kardouchi // International Conference on Agents and Artificial Intelligence – 6th International Conference, ICAART 2014, Angers, France, March 6-8, 2014, Revised Selected Papers: Agents and Artificial Intelligence. – Springer International Publishing Switzerland 2015. – pp. 312-328

8. Handl, J. Clustering Criteria in Multiobjective Data Clustering [Text] / Julia Handl, Joshua Knowles // International Conference on Parallel Problem Solving from Nature – 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part II: Parallel Problem Solving from Nature - PPSN XII. – Springer-Verlag Berlin Heidelberg 2012. – pp. 32-41

9. Yang, X. Research on Domain-Specific Features Clustering Based Spectral Clustering [Text] / Xiquan Yang, Meijia Wang, Lin Fang, Lin Yue, Yinghua Lv // International Conference in Swarm Intelligence – Third International Conference, ICSI 2012, Shenzhen, China, June 17-20, 2012 Proceedings, Part II: Advances in Swarm Intelligence. – Springer-Verlag Berlin Heidelberg 2012. – pp. 84-92

10. Araújo, D. Comparative Study on Information Theoretic Clustering and Classical Clustering Algorithms [Text] / Daniel Araújo, Adrião Dória Neto, Allan Martins // International Conference on Artificial Neural Networks – 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part II: Artificial Neural Networks and Machine Learning – ICANN 2012. – Springer-Verlag Berlin Heidelberg 2012. – pp. 459-466

11. Aszalós, L. Rough Clustering Generated by Correlation Clustering [Text] / László Aszalós, Tamás Mihálydeák // International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing – 14th International Conference, RSFDGrC 2013, Halifax, NS, Canada, October 11-14, 2013. Proceedings: Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. – Springer-Verlag Berlin Heidelberg 2013. – pp. 315-324

12. Almutairi, N. K-Means Clustering Using Homomorphic Encryption and an Updatable Distance Matrix: Secure Third Party Data Clustering with Limited Data Owner Interaction [Text] / Nawal Almutairi, Frans Coenen, Keith Dures // International Conference on Big Data Analytics and Knowledge Discovery – 19th International Conference, DaWaK 2017, Lyon, France, August 28–31, 2017, Proceedings: Big Data Analytics and Knowledge Discovery. – Springer International Publishing AG 2017. – pp. 274-285

13. Zhang, Y. A Novel Clustering and Verification Based Microarray Data Bi-clustering Method [Text] / Yanjie Zhang, Hong Wang, Zhanyi Hu // International Conference in Swarm Intelligence – First International Conference, ICSI 2010, Beijing, China, June 12-15, 2010, Proceedings, Part II: Advances in Swarm Intelligence. – Springer-Verlag Berlin Heidelberg 2010. – pp. 611-618

14. Zhu, Q. Semi-supervised Affinity Propagation Clustering Based on Subtractive Clustering for Large-Scale Data Sets [Text] / Qi Zhu, Huifu Zhang, Quanqin Yang // International Conference of Young Computer Scientists, Engineers and Educators – International Conference of Young Computer Scientists, Engineers and Educators, ICYCSEE 2015, Harbin, China, January 10-

12, 2015. Proceedings: Intelligent Computation in Big Data Era. – Springer-Verlag Berlin Heidelberg 2015. – pp. 258-265

15. Jin, H. Scaling-Up Model-Based Clustering Algorithm by Working on Clustering Features [Text] / Huidong Jin, Kwong-Sak Leung, Man-Leung Wong // International Conference on Intelligent Data Engineering and Automated Learning – Third International Conference Manchester, UK, August 12–14, 2002 Proceedings: Intelligent Data Engineering and Automated Learning — IDEAL 2002. – Springer-Verlag Berlin Heidelberg 2002. – pp. 569-575

16. Vega-Pons, S. Partition Selection Approach for Hierarchical Clustering Based on Clustering Ensemble [Text] / Sandro Vega-Pons, José Ruiz-Shulcloper // Iberoamerican Congress on Pattern Recognition – 15th Iberoamerican Congress on Pattern Recognition, CIARP 2010, Sao Paulo, Brazil, November 8-11, 2010. Proceedings: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. – Springer-Verlag Berlin Heidelberg 2010. – pp. 525-532

17. Bhattacharyya, D. A New Distributed Algorithm for Large Data Clustering [Text] / D. K. Bhattacharyya, A. Das // International Conference on Intelligent Data Engineering and Automated Learning – IDEAL 2000: Intelligent Data Engineering and Automated Learning — IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents Second International Conference Shatin, N.T., Hong Kong, China, December 13–15, 2000 Proceedings. – Springer-Verlag Berlin Heidelberg 2000. – pp. 29-34

18. Daliri, M. Unsupervised Clustering of Shapes [Text] / Mohammad Reza Daliri, Vincent Torre // International Symposium on Visual Computing ISVC 2006: Advances in Visual Computing – Second International Symposium, ISVC 2006 Lake Tahoe, NV, USA, November 6-8, 2006, Proceedings, Part I. – Springer-Verlag Berlin Heidelberg 2006. – pp. 712-720

19. Castellani U. Towards Information Visualization and Clustering Techniques for MRI Data Sets [Text] / Umberto Castellani, Carlo Combi, Pasquina Marzola, Vittorio Murino, Andrea Sbarbati, Marco Zampieri //

Conference on Artificial Intelligence in Medicine in Europe – AIME 2005: Artificial Intelligence in Medicine, 10th Conference on Artificial Intelligence in Medicine, AIME 2005, Aberdeen, UK, July 23-27, 2005. Proceedings. – Springer-Verlag Berlin Heidelberg 2005. – pp. 315-319

20. Boryczka U. Adaptive Ant Clustering Algorithm with Pheromone [Text] / Urszula Boryczka, Jan Kozak // Asian Conference on Intelligent Information and Database Systems – ACIIDS 2016: Intelligent Information and Database Systems, 8th Asian Conference, ACIIDS 2016, Da Nang, Vietnam, March 14–16, 2016, Proceedings, Part II. – Springer-Verlag Berlin Heidelberg 2016. – pp. 117-126

21. Mzili, I. Hybrid Penguins Search Optimization Algorithm and Genetic Algorithm Solving Traveling Salesman Problem [Text] / Ilyass Mzili, Mohammed Essaid Riffi, Fatiha Benzekri // Proceedings of the International Conference on Advanced Information Technology, Services and Systems (AIT2S-17) Held on April 14/15, 2017 in Tangier: Advanced Information Technology, Services and Systems. – Springer International Publishing AG 2018. – pp. 461-473

22. Karthikeyan, P. Improvement in Genetic Algorithm with Genetic Operator Combination (GOC) and Immigrant Strategies for Multicast Routing in Ad Hoc Networks [Text] / P. Karthikeyan, Subramanian Baskar // International Conference on Swarm, Evolutionary, and Memetic Computing – 4th International Conference, SEMCCO 2013, Chennai, India, December 19-21, 2013, Proceedings, Part I: Swarm, Evolutionary, and Memetic Computing. – Springer International Publishing Switzerland 2013. – pp. 481-491

23. Oliveira, S. Genetic Seam Carving: A Genetic Algorithm Approach for Content-Aware Image Retargeting [Text] / Saulo A. F. Oliveira, Francisco N. Bezerra, Ajalmar R. Rocha Neto // Iberian Conference on Pattern Recognition and Image Analysis – 7th Iberian Conference, IbPRIA 2015, Santiago de Compostela, Spain, June 17-19, 2015, Proceedings: Pattern Recognition and Image Analysis. – Springer International Publishing Switzerland 2015. – pp. 700-707

24. Smith, M. Feature Construction and Selection Using Genetic Programming and a Genetic Algorithm [Text] / Matthew G. Smith, Larry Bull // European Conference on Genetic Programming – 6th European Conference, EuroGP 2003 Essex, UK, April 14–16, 2003 Proceedings: Genetic Programming. – Springer-Verlag Berlin Heidelberg 2003. – pp. 229-237

25. Seo, K. A Comparative Study between Genetic Algorithm and Genetic Programming Based Gait Generation Methods for Quadruped Robots [Text] / Kisung Seo, Soohwan Hyun // European Conference on the Applications of Evolutionary Computation – EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I: Applications of Evolutionary Computation. – Springer-Verlag Berlin Heidelberg 2010. – pp. 352-360

26. Badariah, Sh. Comparison between Genetic Algorithm and Genetic Programming Performance for Photomosaic Generation [Text] / Shahrul Badariah Mat Sah, Vic Ciesielski, Daryl D'Souza, Marsha Berry // Asia-Pacific Conference on Simulated Evolution and Learning – 7th International Conference, SEAL 2008, Melbourne, Australia, December 7-10, 2008. Proceedings: Simulated Evolution and Learning. – Springer-Verlag Berlin Heidelberg 2008. – pp. 259-268

27. Karr, Ch. Genetic Algorithm Optimization of a Filament Winding Process [Text] / Charles L. Karr, Eric Wilson, Sherri Messimer // Genetic and Evolutionary Computation Conference – Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part II: Genetic and Evolutionary Computation — GECCO 2003. – Springer-Verlag Berlin Heidelberg 2003. – pp. 2406-2407

28. Nyathi, Th. Automated Design of Genetic Programming Classification Algorithms Using a Genetic Algorithm [Text] / Thambo Nyathi, Nelishia Pillay // European Conference on the Applications of Evolutionary Computation – 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April

19-21, 2017, Proceedings, Part II: Applications of Evolutionary Computation. – Springer International Publishing AG 2017. – pp. 224-239

29. Peng, B. An Adaptive Genetic Simulated Annealing Algorithm for QoS Multicast Routing [Text] / Bo Peng, Lei Li // International Conference on Multimedia, Computer Graphics, and Broadcasting – International Conference, MulGraB 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, in Conjunction with GDC 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings, Part II: Multimedia, Computer Graphics and Broadcasting. – Springer-Verlag Berlin Heidelberg 2011. – pp. 338-338

30. Xia, Yi. An Improved FastSLAM Algorithm Based on Genetic Algorithms [Text] / Yi-min Xia, Yi-min Yang // International Symposium on Information and Automation – International Symposium, ISIA 2010, Guangzhou, China, November 10-11, 2010. Revised Selected Papers: Information and Automation. – Springer-Verlag Berlin Heidelberg 2011. – pp. 296-302

31. Tian, B. A Feature Selection Algorithm for Big Data Based on Genetic Algorithm [Text] / Bo Tian, Weizhi Xiong // International Conference on Mechatronics and Intelligent Robotics – Proceedings of the International Conference on Mechatronics and Intelligent Robotics (ICMIR2017) - Volume 1: Recent Developments in Mechatronics and Intelligent Robotics. – Springer International Publishing AG 2018. – pp. 159-163

32. Smith, M. Using Genetic Programming for Feature Creation with a Genetic Algorithm Feature Selector [Text] / Matthew G. Smith, Larry Bull // International Conference on Parallel Problem Solving from Nature – 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings: Parallel Problem Solving from Nature - PPSN VIII. – Springer-Verlag Berlin Heidelberg 2004. – pp. 1163-1171

33. Zhang, Ch. An Effective Feature Selection Scheme via Genetic Algorithm Using Mutual Information [Text] / Chunkai Zhang, Hong Hu // International Conference on Fuzzy Systems and Knowledge Discovery – Second International Conference, FSKD 2005, Changsha, China, August 27-29, 2005,

Proceedings, Part II : Fuzzy Systems and Knowledge Discovery. – Springer-Verlag Berlin Heidelberg 2005. – pp. E1-E1

34. Zhang, M. Using Back Propagation Algorithm and Genetic Algorithm to Train and Refine Neural Networks for Object Detection [Text] / Mengjie Zhang, Victor Ciesielski // International Conference on Database and Expert Systems Applications – DEXA 1999: Database and Expert Systems Applications. – Springer-Verlag Berlin Heidelberg 1999. – pp. 626-635

35. Gao, G. Research on Routing Selection Algorithm Based on Genetic Algorithm [Text] / Guohong Gao, Baojian Zhang, Xueyong Li, Jinna Lv // International Conference on Intelligent Computing and Information Science – International Conference, ICICIS 2011, Chongqing, China, January 8-9, 2011. Proceedings, Part II: Intelligent Computing and Information Science. – Springer-Verlag Berlin Heidelberg 2011. – pp. 353-358

36. Lee, Z. A Hybrid Search Algorithm of Ant Colony Optimization and Genetic Algorithm Applied to Weapon-Target Assignment Problems [Text] / Zne-Jung Lee, Wen-Li Lee // International Conference on Intelligent Data Engineering and Automated Learning – 4th International Conference, IDEAL 2003, Hong Kong, China, March 21-23, 2003. Revised Papers: Intelligent Data Engineering and Automated Learning. – Springer-Verlag Berlin Heidelberg 2003. – pp. 278-285

37. Chiu, Ch. Combining Apriori Algorithm and Constraint-Based Genetic Algorithm for Tree Induction for Aircraft Electronic Ballasts Troubleshooting [Text] / Chaochang Chiu, Pei-Lun Hsu, Nan-Hsing Chiu // International Conference on Natural Computation – Second International Conference, ICNC 2006, Xi'an, China, September 24-28, 2006. Proceedings, Part I: Advances in Natural Computation. – Springer-Verlag Berlin Heidelberg 2006. – pp. 381-384

38. Yang, D. The T-Detectors Maturation Algorithm Based on Genetic Algorithm [Text] / Dongyong Yang, Jungan Chen // Australasian Joint Conference on Artificial Intelligence – 17th Australian Joint Conference on Artificial

Intelligence, Cairns, Australia, December 4-6, 2004. Proceedings: AI 2004: Advances in Artificial Intelligence. – Springer-Verlag Berlin Heidelberg 2004. – pp. 1262-1268

39. Gholaminezhad, I. A Multi-Objective Relative Clustering Genetic Algorithm with Adaptive Local/Global Search based on Genetic Relatedness [Text] / Iman Gholaminezhad, Giovanni Iacca // European Conference on the Applications of Evolutionary Computation – 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers: Applications of Evolutionary Computation. – Springer-Verlag Berlin Heidelberg 2014. – pp. 591-602

40. Wong, K. A technique for improving the convergence characteristic of genetic algorithms and its application to a genetic-based load flow algorithm [Text] / Kit Po Wong, An Li // Asia-Pacific Conference on Simulated Evolution and Learning – First Asia-Pacific Conference, SEAL'96 Taejon, Korea, November 9–12, 1996 Seclected Papers: Simulated Evolution and Learning. – Springer-Verlag Berlin Heidelberg 1997. – pp. 167-176

41. Емельяненко, Н.Ю. Перспективы использования эволюционных вычислений при кластеризации данных / Н.Ю. Емельяненко, Е.А. Гвоздецкий, Б.М. Элчибекова // Прикладная математика и информатика: современные исследования в области естественных и технических наук: материалы V Международной научно-практической конференции (школы-семинара) молодых ученых: 22-24 апреля 2019 г. – Тольятти: Издатель Качалин Александр Васильевич, 2019. – с. 35-38

42. Емельяненко, Н.Ю. Кластеризация изображения на основе оценки геометрической формы объектов / Н.Ю. Емельяненко, Е.А. Гвоздецкий, Б.М. Элчибекова // Прикладная математика и информатика: современные исследования в области естественных и технических наук: материалы V Международной научно-практической конференции (школы-семинара) молодых ученых: 22-24 апреля 2019 г. – Тольятти: Издатель Качалин Александр Васильевич, 2019. – с. 502-506