Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет **Информационных технологий и программирования**

Направление **Прикладная математика и информатика** Специализация

Академическая степень **магистр прикладной математики и информатики**

Кафедра **Компьютерных технологий** Группа **6538**

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

Применение генетических алгоритмов и эволюционных стратегий для построения управляющих конечных автоматов по сценариям работы

Автор магистерской диссертации **А.С. Тяхти**

Научный руководитель **А.А. Шалыто**

Санкт-Петербург, 2012

ОГЛАВЛЕНИЕ

СПИСОК ТЕРМИНОВ **3**

ВВЕДЕНИЕ **4**

1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ **6**

1.1. Методы, использующие моделирование для вычисления функции приспособленности 6
1.1.1. Генетические алгоритмы *7*
1.1.2. Алгоритм отжига *8*
1.1.3. Эволюционные стратегии *11*
1.2. Построение конечных автоматов по сценариям работы 15
1.3. Построение конечных автоматов на основе тестов 17
1.4. Постановка задачи 18
Выводы по главе 1 21

2. МЕТОД ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ **22**

2.1. Задание тестовых сценариев работы автомата 22
2.2. Генерация начального поколения особей 23
2.3. Применение генетического алгоритма 23
2.3.1. Отбор особей для формирования следующего поколения *24*
2.3.2. Рекомбинация *24*
2.3.3. Мутация *25*
2.3.4. Вычисление функции приспособленности *25*
2.4. Применение эволюционной стратегии 28
Выводы по главе 2 29

3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ **30**

3.1. Методология сравнения 30
3.2. Исследование метода на задачах, полученных случайным образом 35
3.3. Построение автомата управления часами с будильником 40
3.4. Применение многокритериальной оптимизации 41
Выводы по главе 3 45

ЗАКЛЮЧЕНИЕ **46**

ИСТОЧНИКИ **48**

ПРИЛОЖЕНИЕ. ИСХОДНЫЕ КОДЫ МЕТОДА ГЛАВНЫХ КОМПОНЕНТ НА ЯЗЫКЕ ПАКЕТА MAPLE **51**

СПИСОК ТЕРМИНОВ

Автоматное программирование [3] – подход к разработке программных систем со сложным поведением, основанный на модели автоматизированных объектов управления (расширении конечного автомата), каждый из которых содержит систему управления и объект управления.

Генетические алгоритмы [2, 4] – современное и быстроразвивающееся направление в искусственном интеллекте. Представляют собой класс методов оптимизации, основанных на имитации процессов протекающих в природе, в частности естественного отбора – концепции, озвученной в эволюционной теории *Чарльза Дарвина*.

Эволюционные алгоритмы [8, 15] – направление в области искусственного интеллекта, моделирующее процессы биологической эволюции и включающее в себя следующие парадигмы: генетические алгоритмы, эволюционное программирование, эволюционные стратегии и генетическое программирование.

Эволюционные стратегии [8,15] – подкласс эвристических методов оптимизации в разделе *эволюционных алгоритмов*, основанный на адаптации и эволюции. Особенностью данного семейства методов является наличие эндогенных (внутренних) параметров, которые вместе с целевыми параметрами могут изменяться вследствие эволюции особи.

ВВЕДЕНИЕ

В рамках концепции автоматного программирования программа рассматривается как набор конечных автоматов, объектов управления и поставщиков событий. Часто задача построения конечных автоматов эвристическими методами является затруднительной. Это утверждение демонстрирует, например, задача об «умном муравье» [7,11]. Поэтому для генерации автоматов используются автоматизированные методы, в том генетические и эволюционные алгоритмы.

Одним из наиболее сложных этапов при программировании эволюционных алгоритмов является разработка алгоритма расчета функции приспособленности и последующий процесс ее вычисления. Функция приспособленности зачастую рассчитывается путем моделирования процесса работы сгенерированного конечного автомата в условиях решаемой задачи. Сценарии работы существенно упрощают данный этап, так как для каждой новой задачи необходимо лишь задать набор тестов, достаточный для покрытия функциональных возможностей объекта, представленного автоматом. «Ядро» функции приспособленности, основанное на вычислении редакционного расстояния, вычисление которого применяется при построении автоматов по сценариям работы, остается при этом неизменным.

В настоящей работе рассматривается метод построения конечных автоматов по сценариям работы с помощью генетических алгоритмов и эволюционных стратегий. В начале работы описываются основные определения области, рассматриваются существующие методы машинного обучения для построения конечных автоматов, выделяются их сильные и слабые стороны. Далее на основе проведенного анализа предлагается методика совместного применения эволюционных алгоритмов и методов построения управляющих конечных автоматов по сценариям работы. Также описывается метод сравнения эффективности работы алгоритмов по экспериментальным данным, проводится сравнение эффективности применения предложенного

метода совместно с генетическими алгоритмами и эволюционными стратегиями на выбранном наборе задач. В заключении работы приведен анализ полученных результатов, формулируются открытые вопросы, а также выделяются направления для проведения дальнейших исследований.

1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ

1.1. Методы, использующие моделирование для вычисления функции приспособленности

Под термином глобальная оптимизация будем понимать процесс поиска экстремума или экстремумов функции. В эволюционных методах данная функция соответствует приспособленности особи, интерпретируемой как ее способность решать поставленную задачу.

Глобальная оптимизация применима к широкому классу задач. Она находит применение в проблемах проектирования, теории управления физическими процессами, распределении ограниченных ресурсов, анализе данных и других областях.

Известные методы глобальной оптимизации можно разделить на *детерминированные* и *стохастические*. Детерминированные методы находят глобальное решение посредством поиска на всем допустимом множестве. Поэтому большинство детерминированных алгоритмов теряют эффективность с возрастанием размерности задачи. Стохастические методы позволяют уйти от проблем детерминированных алгоритмов. Стохастический подход присутствует не только в разработке алгоритма, но и, например, при определении условия остановки.

К числу стохастических методов глобальной оптимизации относят алгоритм имитации отжига, генетические алгоритмы, эволюционные и поведенческие стратегии, метод виртуальных частиц, алгоритм контролируемого случайного поиска.

1.1.1. Генетические алгоритмы

Генетические алгоритмы – класс методов оптимизации, основанных на имитации процессов протекающих в природе, в частности естественного отбора – концепции, озвученной в эволюционной теории Чарльза Дарвина. В соответствие с теорией Дарвина в естественной среде преимущество к выживанию и размножению имеют особи, более приспособленные к условиям конкретной среды обитания. Основным материалом для естественного отбора служат естественные мутации генов и их комбинации, получаемые при размножении.

Работа осуществляется с набором «особей» - строк битов, определяющих решение задачи. В ходе естественного отбора выживают особи с наибольшей функцией приспособленности (*fitness function*) - численной характеристикой, определяющейся в соответствии с конкретной задачей. Наиболее приспособленные особи получают возможность скрещиваться (кроссовер, crossover) и давать потомство. После этого на получившуюся популяцию оказывают влияния случайные мутации. Общая схема «классического» генетического алгоритма приведена на рис.1.

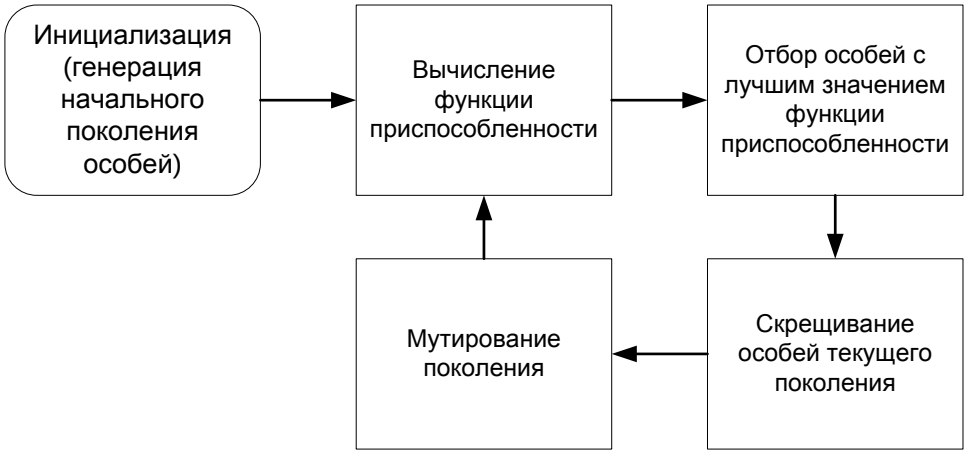

Рис 1. Общая схема классического ГА

Также известны различные модификации генетического алгоритма. Например, в ходе островного (*Island GA*) генетического алгоритма, представляющего собой методику взаимодействия параллельно работающих генетических алгоритмов, особи разбиты на несколько равных групп –

«островов». Особи одного острова могут скрещиваться только с особями своего же острова, однако с периодичностью в несколько итераций алгоритма происходит обмен лучшими особями между островами.

Клеточный генетический алгоритм (*Cellular GA*) является еще одной эффективной методикой взаимодействия параллельно выполняющихся генетических алгоритмов. В данном методе особи размещаются на квадратном поле, разбитом на равные клетки и замкнутом в тор. Таким образом, каждая клетка имеет четырех соседей, с которыми и происходит скрещивание. В ходе генерации очередного поколения особь скрещивается с соседом, имеющим наибольшую функцию приспособленности. Лучший из получившихся потомков помещается на место родителя, либо родитель переходит в следующее поколение без изменений – в случае, если значение fitness-функции детей ниже родительского.

1.1.2. Алгоритм отжига

Метод отжига [20, 21], также известен под названиями: метод обжига, метод симуляции отжига, метод модельной закалки, *simulated annealing*. Этот метод – техника оптимизации, в которой применяется упорядоченный случайный поиск на основе аналогии с процессами, происходящими при охлаждении вещества – когда в этом веществе образуется кристаллическая структура с минимальной энергией.

Общую схему алгоритма можно представить в виде следующей структурной схемы (рис.2).

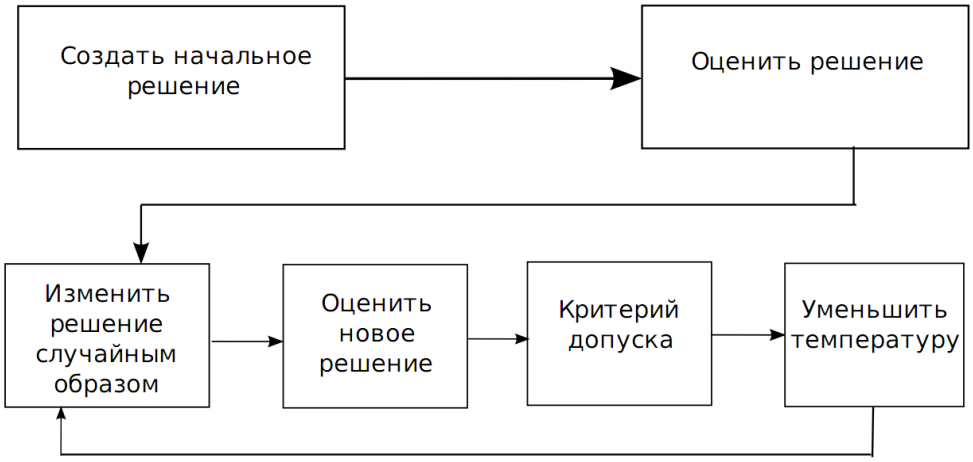

Рис 2. Структурная схема метода отжига

Алгоритм имитации отжига основан на моделировании физического процесса, который происходит при кристаллизации вещества из жидкого состояния в твердое (к примеру, при отжиге металлов). Предполагается, что, во-первых, процесс протекает при понижающейся температуре, а во-вторых, атомы в веществе уже выстроились в кристаллическую решетку, однако переходы отдельных атомов из одной ячейки в другую еще возможны. Вероятность этих переходов в свою очередь обусловлена температурой: чем ниже температура, тем ниже вероятность. Устойчивая кристаллическая структура вещества соответствует минимальному значению энергии. Это значит, что атом либо переходит в состояние с меньшим уровнем энергии, либо остается на месте.

Формализуем данный процесс. Фактически, при моделировании ищется некоторая точка (либо множество точек), на котором достигается минимум некоторой числовой функции F(x). Строится последовательность точек $\vec{x}_0, \vec{x}_1 \ldots \vec{x}_n$, где $\vec{x}_0$ соответствует начальному разделению. При достижении точки $\vec{x}_n$ алгоритм завершает свою работу. Пусть рассматривается текущая точка $\vec{x}_i$. К ней применяется некоторый оператор A, который произвольным образом модифицирует эту точку, в результате чего получается новая точка

$\vec{x}^*$ Вероятность, с которой $\vec{x}^*$ станет следующей точкой $\vec{x}_{i+1}$ равна $P(\vec{x}^*, \vec{x}_{i+1})$, где P- распределение Гиббса:

$$P(\vec{x}^* \to \vec{x}_{i+1} | \vec{x}_i) = \left\{ \begin{array}{ll} 1, & F(\vec{x}^*) - F(\vec{x}_i) < 0 \\ exp(-\frac{F(\vec{x}^*) - F(\vec{x}_i)}{Q_i}), & F(\vec{x}^*) - F(\vec{x}_i) \geq 0 \end{array} \right\}$$

Здесь $Q_i > 0$ – элементы произвольной убывающей, сходящейся к нулю последовательности. Она представляет собой аналог понижающейся температуры во время реального физического процесса.

Достоинством метода отжига является свойство избегать «ловушек» в локальных минимумах оптимизируемой функции, и продолжить поиск глобального минимума. Это достигается за счет принятия не только изменений параметров, приводящих к уменьшению значения функции, но и некоторых изменений, увеличивающих ее значение, в зависимости от температуры T – характеристики моделируемого процесса. Чем выше температура, тем большие «ухудшающие» изменения (аналогичные случайным флуктуациям в нагретом веществе) допустимы, и больше их вероятность. Еще одним достоинством является то, что даже в условиях нехватки вычислительных ресурсов для нахождения глобального минимума, метод отжига, как правило, выдает весьма неплохое решение (один из локальных минимумов). *Л. Ингбером* [16] показано, что метод отжига и его модификации являются одним из наиболее эффективных методов случайного поиска оптимального решения для большого класса задач. Им же проведен сравнительный анализ адаптивного метода отжига (Adaptive Simulated Annealing – ASA) и генетических алгоритмов, из которого следует, что на большинстве задач метод отжига не проигрывает генетическим алгоритмам, а на многих и выигрывает. К настоящему времени разработано множество различных вариантов метода отжига, как общих, так и их специализаций для решения конкретных задач. [6, 18, 19].

1.1.3. Эволюционные стратегии

Эволюционные стратегии, как подкласс эволюционных алгоритмов, были предложены студентами Берлинского Технического Университета Рехенбергом и Швефелем в 1965 г., и впервые применены для физической задачи – оптимизации формы объемного тела в аэродинамической трубе.

В данном разделе приводится базовый алгоритм метода эволюционных стратегий, и дается описание общепринятой терминологии и нотации.

Основная задача метода эволюционных стратегий – оптимизировать заданную n-мерную функцию $F: \mathcal{Y} \to \mathbb{R}$, определенной на множестве, например, вещественных чисел $\mathcal{Y} \subset \mathbb{R}^n$, то есть:

$$F(\boldsymbol{y}) \longrightarrow opt., \boldsymbol{y} \in \mathcal{Y}, \tag{1}$$

где функция F – функция приспособленности, а элементы вектора $\boldsymbol{y}$ – *целевые параметры (object parameters).*

Эволюционные стратегии работают с популяцией $\mathfrak{B}$ особей $\mathfrak{a}$. Каждая особь $\mathfrak{a}_k$ представляет собой не только множество целевых параметров $\boldsymbol{y}_k$ и значение функции приспособленности $F_k = F(\boldsymbol{y}_k)$, но также множеством *эндогенных* (endogenous) параметров стратегии эволюции $\boldsymbol{s}_k$:

$$\mathfrak{a}_k = \left(\boldsymbol{y}_k, \boldsymbol{s}_k, F(\boldsymbol{y}_k)\right) \tag{2}$$

Специфичная особенность эволюционных стратегий, отличающая метод от других эволюционных алгоритмов – эндогенные параметры, которые вместе с целевыми параметрами могут изменяться вследствие эволюции особи. Основная их цель – контролировать статистические свойства *оператора мутации*, о котором будет сказано далее.

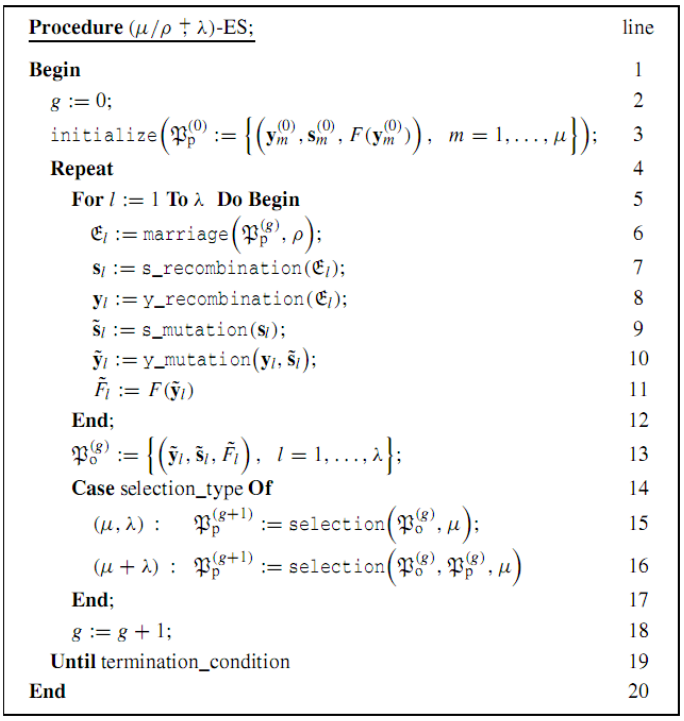

Рис. 3. Базовый алгоритм метода эволюционных стратегий

Способ генерации очередного поколения записывается в следующей нотации: $(\mu/\rho \overset{+}{,} \lambda)$, где ρ – число родительских особей, вовлеченных в построение каждой дочерней особи. В случае $\rho = 1$ получаем вырожденную эволюционную стратегию без *рекомбинации*. Такие стратегии обозначаются $(\mu,\ \lambda)$ и $(\mu + \lambda)$ в зависимости от типа *селекции*. Остальные же варианты $(\rho > 1)$ называются стратегиями с рекомбинацией. "$\overset{+}{,}$" определяет тип селекции.

На рис. 3 приведен базовый алгоритм метода эволюционных стратегий. В строке № 3 инициализируется начальное поколение $\mathfrak{B}_p^{(0)} = (\mathfrak{a}_1, \ldots, \mathfrak{a}_\mu)$. Далее в цикле (строки № 6 – 11) генерируется дочерняя популяция из λ особей из родительской популяции $\mathfrak{B}_p^{(g)}$. Результатом каждой итерации цикла является генерация очередной особи по следующим правилам: в строке № 6 случайным образом из родительской популяции размера μ выбираются ρ особей для

дальнейшей рекомбинации. В строках № 7, 8 целевые и эндогенные параметры новой особи создаются посредством рекомбинации родительских. В случае $\rho = 1$ параметры представляют собой чистую копию родительской особи. Далее в строках № 9, 10 производится мутация эндогенных и целевых параметров. Итого, полученная дочерняя популяции $\mathfrak{B}_0^{(g)}$ размера μ подвергается селекции на основании значения функции приспособленности и типа селекции: прежняя родительская популяция либо принимает участие в селекции, либо игнорируется. Затем проверяется условие выхода: это может быть, например, число поколений или время выполнения программы.

1.1.3.1. Селекция

В зависимости от того, включается ли родительская популяция в процедуру селекции, различают две ее разновидности: $(\mu + \lambda)$ и $(\mu\,,\,\lambda)$ соответственно.

В случае селекции вида $(\mu\,,\,\lambda)$ только λ дочерних особей образуют множество $\mathfrak{B}_0^{(g)}$ над которым операция селекция и осуществляется. Другими словами, родительская популяция (g) исключается из рассмотрения, даже если функции приспособленности ее представителей больше дочерних.

Второй вариант селекции – $(\mu + \lambda)$, который принимает во внимание родительскую популяцию, и отбор осуществляется из $\mu + \lambda$ особей. Данный вид селекции гарантирует выживание особи с наибольшим значением функции приспособленности, найденной в текущем поколении – данная концепция, получившее широкое применение в генетических алгоритмах, имеет название *элитизм*.

Оба варианта селекции имеют свои области применения. В частности, $(\mu + \lambda)$ чаще всего используют для $\mathcal{Y} \subset \mathbb{R}^n$, $(\mu\ ,\ \lambda)$ – для дискретных пространств поиска. Об особенностях применения каждого из видов селекции к задачам построения управляющих конечных автоматов будет сказано далее.

1.1.3.2. Рекомбинация

Оператор рекомбинации предоставляет возможность особям обмениваться генетическим материалом. В отличие от генетических алгоритмов, где оператор скрещивания применяется к двум предкам для получения двух потомков, в эволюционных стратегиях выполняется на ρ предках для получения единственного потомка.

Существует два варианта рекомбинации: в первом каждый элемент целевого параметра вычисляется, как среднее арифметическое значений соответствующего параметра родительских особей, во втором значение каждого элемента целевого параметра определяется случайным выбором у единственного родителя.

1. $\boldsymbol{r}_k = (\boldsymbol{a}_{m_k})_k$, где $m_k = Random\{1, \dots, \rho\}$. (4)
2. $\boldsymbol{r}_k = \frac{1}{\rho}\sum_{m=1}^{\rho}(\boldsymbol{a}_m)_k$.

1.1.3.3. Мутация

Приведем описание оператора мутации, который, как правило, применяется в случае непрерывных пространств поиска. В простейшем варианте в качестве единственного эндогенного параметра $\boldsymbol{s}$ выступает дисперсия нормального распределения σ. В этом случае значение целевого параметра в очередном поколении вычисляется следующим образом:

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{z}, \quad (3)$$

$\boldsymbol{z} = \sigma\big(\mathcal{N}_1(0,1), \dots \mathcal{N}_N(0,1)\big)$, где $\mathcal{N}_i(0,1)$ – случайное число из соответствующего нормального распределения. Также возможно расширение данного подхода, в котором эндогенный параметр представляет собой набор $(\sigma_1, \dots, \sigma_N)$, и $\boldsymbol{z} = (\sigma_1\mathcal{N}_1(0,1), \dots, \sigma_N\mathcal{N}_N(0,1))$.

1.2. Построение конечных автоматов по сценариям работы

Под *сценарием работы* для построения конечного автомата далее будем понимать последовательность троек $T_1 \dots T_n$, где $T_i = \langle e_i, f_i, A_i \rangle$, e_i – входное событие, f_i – охранное условие, построенное на основе булевой формулы от входных переменных, A_i – последовательность выходных воздействий. Отдельная тройка сценария называется его *элементом*. Будем говорить, что автомат, находясь в состоянии *state*, *удовлетворяет элементу сценария* T_i, если из *state* исходит переход, помеченный событием e_i, последовательностью выходных воздействий A_i и охранным условием, тождественно равным f_i как булева формула. Автомат *удовлетворяет сценарию работы* $T_1 \dots T_n$, если он удовлетворяет каждому элементу данного сценария и при этом в нем есть путь из переходов, образованный соответствующими элементами сценария.

В качестве примера, иллюстрирующего данное определение, рассмотрим автомат, приведенный на рис. 8(поменять). Для данного автомата множество входных событий – {*A*, *B*}, охранные условия зависят от логической входной переменной *x*, множество выходных воздействий – {*z*1, *z*2}. В качестве начального состояния выберем состояние с номером 0. Описанный автомат удовлетворяет сценарию работы <*A*, ~*x*, (*z*2)>, <*A*, *x*, (*z*1)>, но не удовлетворяет сценариям работы:

- <*B*, 1, (*z*2)>, так как из начального состояния не исходит перехода, помеченного событием *B*;
- <*A*, *x*, (*z*1)>, <*A*, *x*, (*z*1,*z*1)>, так как из состояния номер 1, при значении

$x = 1$ выполнится переход, помеченный охранным условием 1, а не x;

- <A, x, ($z2$)>, так как из начального состояния выполнится переход, помеченный последовательностью выходных воздействий ($z1$), а не ($z2$).

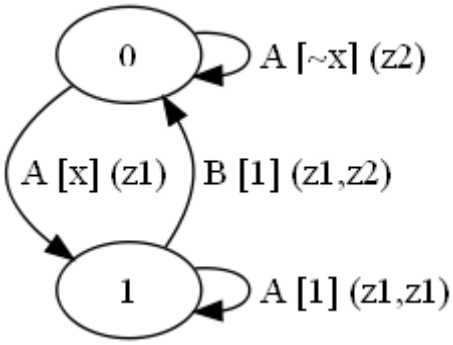

Рис. 4. Пример управляющего автомата

В работе [10] предложен метод построения конечных автоматов на основе сценариев работы, состоящий из следующих этапов.

1. Построение дерева сценариев.

2. Построение графа совместимости вершин дерева сценариев.

3. Построение булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата.

4. Запуск программы, решающей задачу о выполнимости булевой КНФ-формулы.

5. Построение автомата по найденному выполняющему набору значений переменных.

Экспериментальные исследования [10] показали, что данный метод по производительности на порядки превосходит генетические алгоритмы. К минусам данного подхода можно отнести сравнительную сложность его реализации.

1.3. Построение конечных автоматов на основе тестов

После выделения на начальном этапе проектирования программы событий (e1, e2, …), входных переменные (x1, x2, …) и выходных воздействий (z1, z2, …), дальнейшая ее разработка может идти разными путями. Вариант построения автомата по ранее подготовленным сценариям работы программы, описан в предыдущем разделе данной главы. Еще один подход, редко применяющийся для построения автоматов, но широко распространенный при написании программ, заключается в разработке на основе тестов (test-driven development) [12]. В данном случае тест — это процедура, которая позволяет либо подтвердить, либо опровергнуть корректность работы автомата применительно к конкретной задаче.

В качестве тестов для управляющего конечного автомата рассматриваются пары последовательностей, одна из которых *Input[i]* - описывает события и входные переменные, поступающие на вход автомату, другая *Answer[i]* – выходные воздействия, которые должны вырабатываться автоматом при обработке этих событий. Стоит отметить, что структура тестов не определяет какому входному воздействию должно соответствовать конкретное выходное воздействие, что приводит к сложностям при реализации указанного подхода.

При использовании метода построения управляющих конечных автоматов на основе тестов совместно с генетическими алгоритмами особь представляет из себя лишь «скелет» автомата. Это означает, что для ее состояний описаны переходы, которые, в свою очередь, описываются входными событиями, при которых они должны выполняться, а также числом выходных воздействий. Конкретные же выходные воздействия определяются на основе алгоритма расстановки пометок[7].

Алгоритм расстановки пометок

Суть алгоритма расстановки пометок на переходе заключается в расстановке на основе имеющихся тестов выходных воздействий на «скелете» автомата, генерируемом генетическим алгоритмом.

Как было сказано в предыдущем разделе, для каждого перехода особь генетического алгоритма содержит пометку, сколько выходных воздействий должно вырабатываться при выполнении этого перехода. Подадим на вход конечному автомату последовательность входных событий одного из тестов, и будем отмечать – какие переходы выполнит автомат. Зная эти переходы, а также информацию о том, сколько выходных воздействий должно быть сгенерировано на каждом из переходов, можно определить, какие выходные воздействия должны быть расставлены на переходах, использовавшихся при обработке входной последовательности.

Различные ситуации возможны в случае рассмотрения последующих тестов.

- может оказаться, что в разных тестах используется один и тот же переход, однако на всех тестах на данном переходе должны генерироваться одни и те же выходные воздействия;
- в нескольких тестах используется один и тот же переход, и на разных тестах при выборе данного перехода должны генерироваться разные выходные воздействия (противоречие).

В первом случае ситуация аналогична ситуации с одним тестом. Во втором - «скелет» автомата не позволяет пройти все тесты. С другой стороны, на этом противоречивом переходе некоторые выходные воздействия должны выполняться. Поэтому предлагается подсчитать частоту различных выходных воздействий, вырабатываемых на этом переходе на разных тестах и пометить его наиболее часто встречающимся воздействием. Подобный алгоритм поведения применим и в случае, когда на переходе должно вырабатываться несколько выходных воздействий [7].

1.4. Постановка задачи

Управляющие конечные автоматы зачастую имеют сложную структуру и характеризуются множествами управляющих состояний, управляющих функций и действий. Проектирование подобных автоматов является достаточно

сложным, особенно если речь идет о системах, в рамках которых имеет место взаимодействие агентов, управляемых разными конечными автоматами. Поэтому актуальным представляется вопрос об автоматизированном построении управляющих конечных автоматов, в ходе которого работа по генерации автоматов возлагается на компьютер.

Существует несколько подходов для построения управляющих конечных автоматов: разработка на основе, с помощью алгоритма объединения состояний на основе свидетельств EDSM (`Evidence-Driven State Merging`), генерации, посредством применения различных оптимизационных методов (генетические алгоритмы, различные модификации метода отжига (`simulated annealing`), эволюционные стратегии).

Экспериментальные исследования [6, 8] показали, что методы оптимизации могут успешно применяться для построения управляющих конечных автоматов. Однако методы оптимизации чаще всего подразумевает вычисление функции приспособленности с помощью моделирования [1].

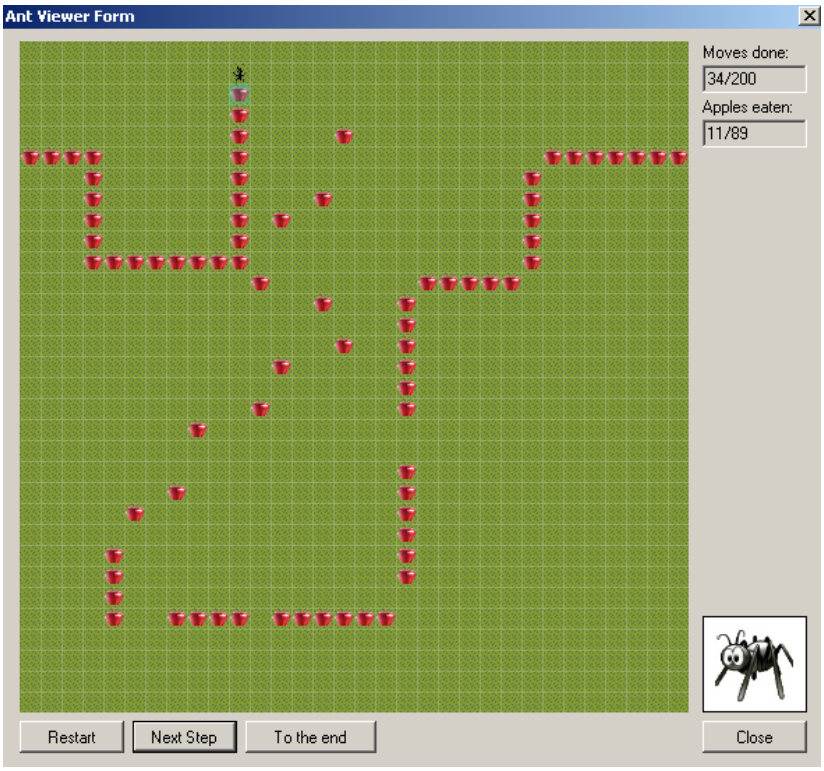

Рис. 5. Пример вычисления функции приспособленности с помощью моделирования. Визуализатор задачи об «умном муравье» виртуальной лаборатоии *Glopt*.

Процесс данного моделирования зачастую трудоемок и требует большого количества вычислительных ресурсов. В то же время методы вычисления функции приспособленности на основе обучающих примеров и сценариев работы не обладают указанным недостатком, так как формулируют требования к поведению искомого автомата еще до начала процесса его построения.

В работах [6, 8] было проведено сравнительное исследование более десятка эволюционных алгоритмов на задачах об «умном муравье», о роботе, огибающем препятствия и о расстановке ферзей. На основании проведенных экспериментов был сделан вывод о том, что наиболее оптимальными показателями с точки зрения производительности и времени работы обладают генетические алгоритмы и эволюционные стратегии. Именно на основе этих данных были выбраны методы для проведения сравнительных экспериментов в настоящей работе.

Таблица 1. Сокращения названий методов, приведенных на рис. 6

Сокр.	Название
A	Оптимизация методом роя частиц
B	Оптимизация методом пчелиной колонии
C	Эволюционная стратегия (1+1)
D	Эволюционная стратегия $(\mu/\rho + \lambda)$
E	Эволюционная стратегия $(\mu/\rho + \lambda)$ с самоадаптацией
F	Эволюционная стратегия $(\mu/\rho\ ,\lambda)$
G	Классический генетический алгоритм
H	Островной генетический алгоритм

I	Клеточный генетический алгоритм
J	Случайный поиск

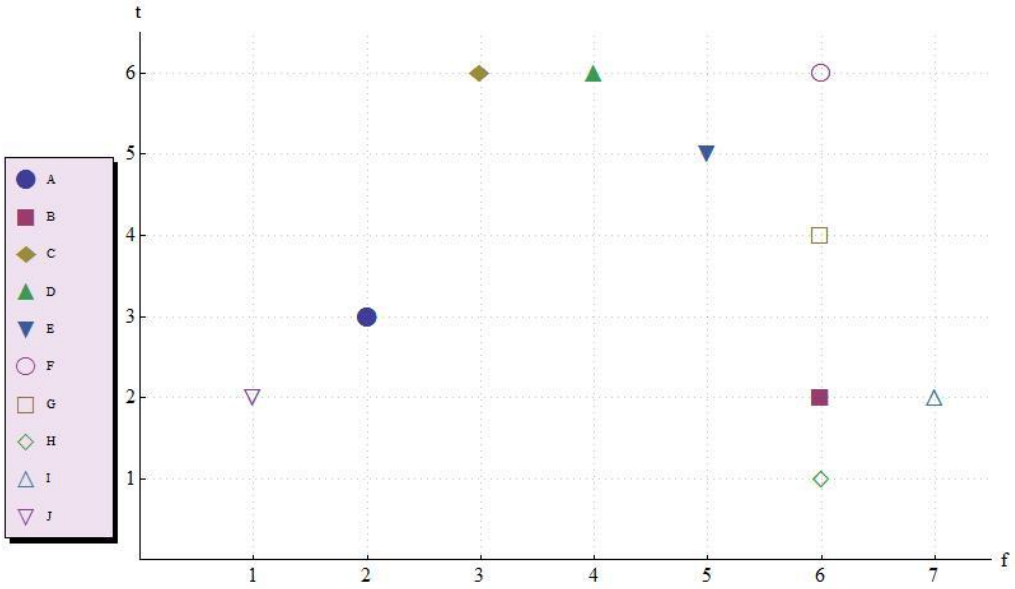

Рис. 6. Результаты сравнения группы эволюционных алгоритмов по времени выполнения и значению функции приспособленности (большая координата точки соответствует большему значению функции приспособленности и меньшему времени выполнения алгоритма.

В настоящей работе предлагается описание метода, объединяющего преимущества алгоритма построения автоматов с помощью сценариев работы и методов оптимизации.

Выводы по главе 1

1. Выполнен обзор существующих методов машинного обучения для построения управляющих конечных автоматов. Выделены основные преимущества и недостатки данных методов.
2. На основе проведенного анализа поставлена задача о применимости методов оптимизации для построения управляющих конечных автоматов по сценариям работы, решаемая в данной работе.

2. МЕТОД ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ

В настоящей главе описывается метод построения управляющих конечных автоматов по тестовым сценариям работы.

Предлагаемый метод содержит три ключевых этапа.

1. Задание\генерация набора тестовых сценариев работы автомата.
2. Генерация начального поколения особей применяемого эволюционного алгоритма – «скелетов» автоматов.
3. Применение эволюционного алгоритма.
   3.1. Запуск очередной итерации алгоритма.
   3.2. Вычисление функции приспособленности особей текущего поколения с использованием сценариев работы.

2.1. Задание тестовых сценариев работы автомата

В зависимости от типа решаемой задачи тестовые сценарии работы автомата могут быть либо заданы вручную, либо сгенерированы по эталонному автомату, с которым в дальнейшем сравниваются решения, найденные в ходе работы алгоритма. В настоящей работе рассматривается одна задача, требующая ручного задания сценариев – задача построения автомата управления часами с будильником [7]. В остальных задачах сценарии строятся автоматически по ранее сгенерированному эталонному автомату.

Входные данные для построения эталонного автомата содержат:

- общее число состояний автомата – N;
- процент переходов от максимально возможного в полном графе с N вершинами;
- минимальное и максимальное число переходов, возможных из состояния;
- число возможных разнообразных входных воздействий;
- число возможных переменных для составления охранного условия;
- число возможных выходных воздействий;

- минимальное и максимальное число выходных воздействий на переходе.

Как было описано выше тестовый сценарий представляет собой тройку входных воздействий, охранных условий и выходных воздействий $T_i = <e_i, f_i, A_i>$. Таким образом, генерация сценария работы по эталонному автомату представляет собой создание набора случайным образом составленных путей в графе, образованном состояниями и переходами автомата. В ходе построения пути генератор сценария запоминает тройки элементов сценария $<e_i, f_i, A_i>$, объединяя их в готовый сценарий работы.

2.2. Генерация начального поколения особей

Генерация начального поколения особей эволюционного алгоритма осуществляется случайным образом. Следовательно, случайным образом определяются следующие ее элементы.

- Число переходов, а также их расположение.
- Число и набор входных воздействий на переходах.
- Число охранных условий на переходе, а также вид итогового охранного выражения с их участием.

Таким образом, особь представляет собой «скелет» автомата. Переходы не содержат описания выходных воздействий, подстановка которых осуществляется уже в ходе итераций эволюционного алгоритма при вычислении значения функции приспособленности особи.

2.3. Применение генетического алгоритма

Дальнейшие шаги метода предполагает запуск одного из эволюционных алгоритмов. Далее описывается методология использования генетического алгоритма совместно с тестовыми сценариями работы автомата.

Генетический алгоритм предполагает цикличное выполнение следующих шагов.

- Отбор особей текущего поколения для формирования следующего.
- Скрещивание особей (кроссовер).

- Мутация.
- Вычисление функции приспособленности для особей текущего поколения.

Также с заданной периодичностью осуществляется проведение массовой мутации (big mutation), в ходе которой мутирует заданный алгоритмом процент особей. Число подверженных массовой мутации особей в несколько раз превышает число особей, изменяющихся в ходе мутации на очередной итерации алгоритма. Сами изменения особей осуществляющиеся в ходе массовой мутации идентичны изменениям, выполняемым в ходе обычной процедуры мутации.

Критерием останова алгоритма может являться достижение функции приспособленности максимально возможного значения, выполнение алгоритмом заданного числа итераций, либо неизменность максимального значения функции приспособленности в течение определенного отрезка времени.

2.3.1. Отбор особей для формирования следующего поколения

Отбор особей текущего поколения для формирования следующего осуществляется с применением двух принципов, характерных для использования классического генетического алгоритма. А именно выбора заданного процента элитных особей, обладающих наибольшим значением функции приспособленности, и формирования оставшейся части нового поколения с помощью метода рулетки, при использовании которого каждая особь имеет шанс быть выбранной для скрещивания с вероятностью пропорциональной значению собственной функции приспособленности.

2.3.2. Рекомбинация

Скрещивание особей, отобранных на предыдущем этапе алгоритма, осуществляется следующим образом. На вход функции кроссовера, возвращающей два новых автомата, поступает два случайно выбранных родительских автомата текущего поколения. Далее случайным образом с

вероятностью 1/2 происходит распределение начальных состояний родительских автоматов между потомками, а также распределение остальных состояний вместе с переходами, выходящими из них.

2.3.3. Мутация

Мутация осуществляется для заданного числа особей текущего поколения. По умолчанию их процент установлен на уровне пяти процентов (значение подобрано опытным путем), но может изменяться параметризацией запускаемой программы.

Мутация предполагает выполнение для поступившего на вход процедуры автомата одного из следующих действий.

- Изменение\удаление\добавление одного из входных воздействий на случайно выбранном переходе автомата.
- Изменение\удаление\добавление части охранного условия на случайно выбранном переходе автомата.
- Полное удаление случайно выбранного перехода автомата
- Добавление сгенерированного случайным образом нового перехода.

Первые два действия осуществляются с вероятностью, превышающей вероятность выполнения последних двух.

2.3.4. Вычисление функции приспособленности

Важной составляющей предлагаемого метода построения управляющих конечных автоматов является вычисление функции приспособленности для особей текущего поколения. Расчет значения фитнесс-функции осуществляется в два прохода.

На первом из них происходит расстановка выходных воздействий на переходах автомата. Для этого последовательно рассматриваются все имеющиеся тестовые сценарии работы автомата. Если из текущего состояния имеется переход, входные воздействия и охранные условия которого совпадают с записанными в элементе сценария, то возможны следующие варианты.

- На переходе отсутствуют выходные воздействия. В этом случае выходные воздействия элемента сценария записываются на переходе автомата, текущим состоянием автомата становится то, в которое ведет переход. Далее рассматривается следующий элемент сценария.
- На переходе уже имеются выходные воздействия, одно из которых совпадает с выходным воздействием элемента сценария. В этом случае никаких дополнительных действий не осуществляется, происходит переход к рассмотрению следующего элемента сценария.
- На переходе уже имеются выходные воздействия, ни одно из которых не совпадает с выходными воздействиями элемента сценария, то есть имеется противоречие. В данном случае для перехода происходит увеличение на единицу счетчика различных возможных для него выходных воздействий, после чего выходные воздействия элемента сценария добавляются на переход.

На втором проходе алгоритм вычисления функции приспособленности работает уже с автоматом, для которого на переходах расставлены все возможные в соответствии со сценарием выходные воздействия. Все сценарии вновь проходят последовательное рассмотрение. Для каждого сценария *i* локальная функция приспособленности вычисляется по следующей формуле:

$$ff_i = \frac{1}{l_i}\sum_{j=1}^{l_i}\frac{1}{m_j},$$

где l_i – число элементов сценария, m_j – число различных выходных воздействий на соответствующем переходе автомата.

После подсчета функций приспособленности автомата для каждого сценария рассчитывается итоговая функция приспособленности автомата:

$$ff = \frac{1}{k}\sum_{i=1}^{k} ff_i,$$

где *k* – число тестовых сценариев. Таким образом, функция приспособленности *ff* принимает значения от нуля до единицы.

Рассмотрим в качестве примера автомат на рисунке 7 и набор тестовых сценариев табл. 2.

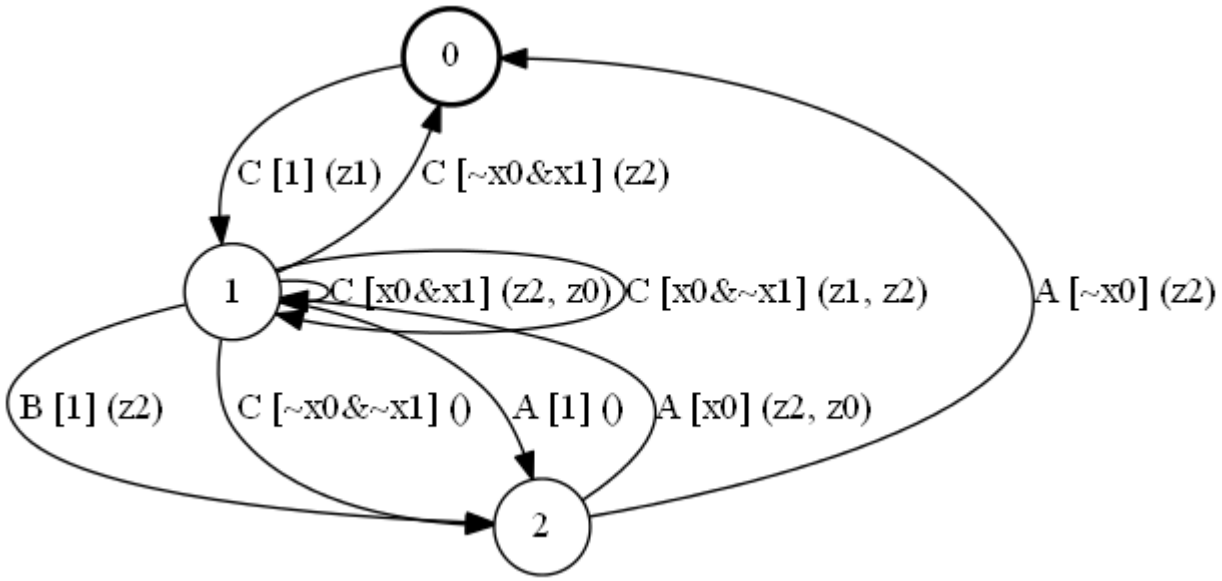

Рис. 7. Пример построенного в ходе работы алгоритма автомата

Таблица 2. Набор тестовых сценариев для автомата на рис. 7

№ сценария	**Входное воздействие**	**Охранное условие**	**Выходное воздействие**
1	C; C	1; ~x0&x1	z1;z2
2	C; C; B	1; x0&x1; 1	z1; z2, z2;
3	C; A; B; A; A;	1; 1; x0; x1	z1; -; z0; z2; z2

Первый сценарий состоит из двух элементов, причем автомат позволяет выполнить оба элемента. Сначала из состояния 0 осуществляется переход по воздействию C и условию 1 в состояние 1, в результате чего выполняется воздействие z1, затем из состоянию 1 по по воздействию C осуществляется переход в состояние 0, на котором выполняется выходное воздействие z2. Приведенный автомат позволяет также полностью пройти второй тестовый сценарий, состоящий из трех элементов. Третий же сценарий выполним автоматом только на 2/5, так как осуществить переход из состояния 2 по входному воздействию B невозможно (третий элемент сценария). Таким образом, подсчет функции приспособленности для третьего сценария

выполняется только для первых двух его элементов. Это обеспечивает выполнение условия: для автомата, удовлетворяющего большему количеству элементов сценария, функция приспособленности будет иметь большее значение.

Сложность вычисления функции приспособленности в предложенном методе зависит только от суммарной длины тестовых сценариев и количества переходов в оцениваемом автомате.

2.4. Применение эволюционной стратегии

В качестве используемой для сравнения с генетическим алгоритмом эволюционной стратегии была выбрана стратегия $(\mu/\rho, \lambda)$, как показавшая, одни из лучших результатов в ранее проводившихся сравнительных исследованиях эволюционных алгоритмов [8].

Данная разновидность эволюционных стратегий предполагает получение популяции из ρ особей для дальнейшей рекомбинации из родительской популяции в μ особей. Стратегия (μ, λ) отличается от стратегии $(\mu + \lambda)$ отсутствием элитизма, т.е. только λ дочерних особей вовлечено в селективный процесс, в то время как в стратегии $(\mu + \lambda)$ отбор осуществляется также и из родительских особей.

Логика основных операций, реализованных в эволюционной стратегии: рекомбинации, мутации, вычисления функции приспособленности соответствует логике данных операций, использующихся в генетическом алгоритме.

Выводы по главе 2

1. Предложен метод машинного обучения для построения управляющих конечных автоматов по сценариям работы.
2. Описанный метод позволяет использовать для построения автоматов эволюционные алгоритмы. Предложена методика вычисления функции приспособленности для автоматов, построение которых ведется на основе тестовых сценариев работы.

3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ

В данной главе приводится описание методологии сравнения эволюционных алгоритмов, описываются задачи, на которых проводилось сравнение, а также приводятся его результаты.

3.1. Методология сравнения

При сравнении детерминированных алгоритмов ключевым критерием их производительности является вычислительная сложность. Такой подход не применим к эволюционным алгоритмам по причине их стохастической природы. Нет гарантии, что каждый запуск алгоритма найдет одно и то же (пусть и не оптимальное) решение, и, даже если такое решение найдено, время его поиска может сильно варьироваться. Большинство эволюционных алгоритмов являются достаточно гибкими в части настройки, и различные варианты параметризации порождают алгоритмы с различающейся производительностью.

Существует несколько подходов к сравнению эволюционных алгоритмов, и, безусловно, основным критерием выбора одного из них является решаемая задача. Если исходить из предположения, что точное решение задачи находится каждым запуском алгоритма, то критерием сходимости решения можно пренебречь, и производить сравнение лишь по затраченным вычислительным ресурсам. Тем не менее, в большинстве случаев, задачи, решаемые эволюционными алгоритмами, являются либо NP-полными, либо нахождение точного решения затруднено по другим причинам. Часть алгоритмов будет находить более грубые решения за небольшое время, некоторые – точное, но за более существенный временной отрезок, поэтому сравнение необходимо проводить по нескольким критериям. Как правило, это затраченные вычислительные ресурсы и точность найденного решения. Удобной моделью для такого рода сравнения является представление критериев случайными

величинами, что позволяет проводить анализ данных и сравнение алгоритмов методами математической статистики.

Описание сравнительной методологии

Приведенная методология сравнения основана на работах [8,14].

Большинство методик сравнения эволюционных алгоритмов, как правило, оперируют сравнением, основанном на среднем значении функции приспособленности, что является причиной потери информации о распределении точек относительного этого среднего. Избежать этого можно, применяя метод *стохастического доминирования*, сравнивая функции распределения критериев вместо их средних значений. В данной работе критериями являются лучшее значение функция приспособленности в текущем запуске алгоритма и затраченное время на его выполнение.

Ниже приводятся основные шаги применяемого алгоритма сравнения, основанного на методе главных компонент (Principal component analysis)[]:

1. Каждый алгоритм выполняется n раз на одной задаче. Критерием останова является неизменное состояние максимального значения функции приспособленности на протяжении двух минут. Результат запуска представляет собой точку $[f_{max}, t] \in \mathbb{R}^2$, где f_{max} – максимальное значение функция приспособленности, t – время выполнения алгоритма.
2. Набор полученных данных подвергается процедуре *декорреляции* по двум критериям, для уменьшения их статистической зависимости.
3. Для каждого критерия выполняются следующие шаги:
    a. Построение плотности вероятности для средних, используя *ресемплинг* (повторные выбороки).
    b. Упорядочивание алгоритмов посредством критериев метода стохастического доминирования.

3.1.1. Процедура декорреляции исходных данных

Дадим описание процедуры в общем виде, которая позволяет провести линейное преобразование исходных данных таким образом, чтобы избавиться от их корреляции.

1. Составить из данных матрицу $X(M \times N)$, где M-критерии, N - измерения.
2. Найти среднее для каждого критерия:

$$u[m,1] = \frac{1}{N}\sum_{i=1}^{N} X[m,n]$$

3. Вычесть среднее u из каждого столбца матрицы X:

$B = X - u \cdot h$, где $h[1,m] = 1 \; \forall m = [1,\dots,M]$

4. Вычислить матрицу ковариации $C(M \times M)$:

$$C = \frac{1}{N} B \cdot B^T$$

5. Найти собственные вектора V матрицы C.
6. Матрица $Y = V \times X$ представляет собой набор данных с нулевой корреляцией.

3.1.2. Построение плотности вероятности средних

Процедура построения плотности вероятности средних для входного массива данных выглядит следующим образом:

1. Выбрать с повторениями S_i - n элементов из исходных данных размера N.
2. Вычислить среднее для S_i.
3. Повторить операции 1 и 2 заданное число раз.

4. Построить по получившимся средним функцию плотности вероятности.

3.1.3. Метод стохастического доминирования

Случайная величина X_1 доминирует над случайной величиной X_2, если $F(X_1) \geq F(X_2) \quad \forall x_i \in X_1 \cup X_2$, где F – функция распределения.

3.1.4. Пример

В качестве примера применения метода главных компонент рассмотрим вычисления для результатов построения управляющего конечного автомата из пяти состояний по десяти сценариям и совокупной длиной 35.

На рис. 8 приведены исходные данные, полученные в результате 200 запусков алгоритма.

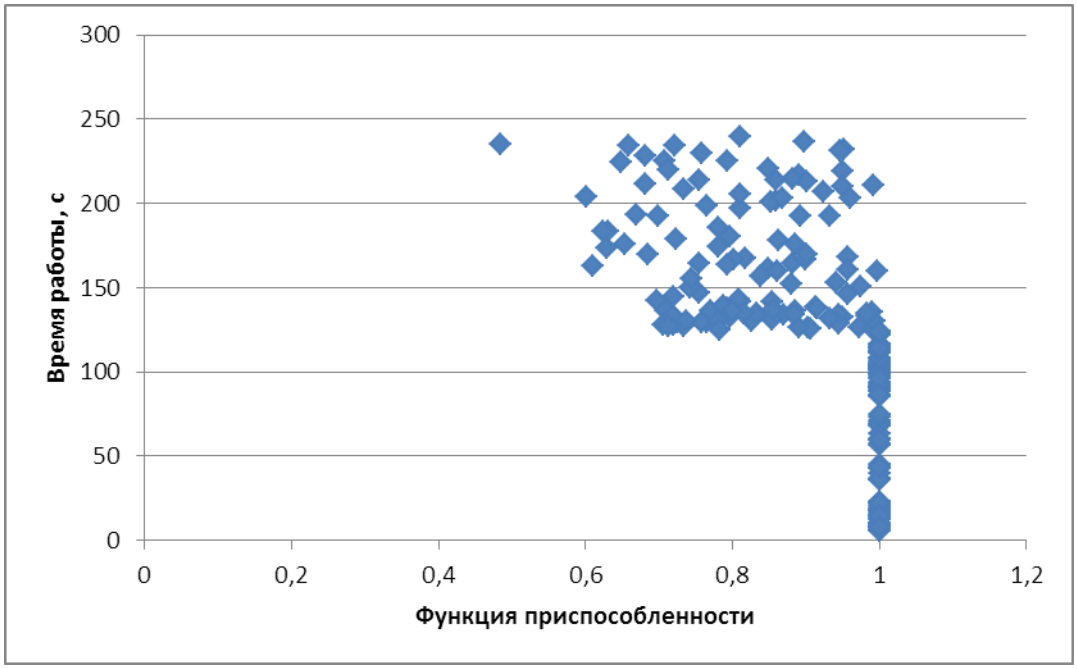

Рис. 8. Исходные данные.

После преобразования исходных данных для обнуления корреляции применяется ресемплинг для получения плотности вероятности средних (рис. 9).

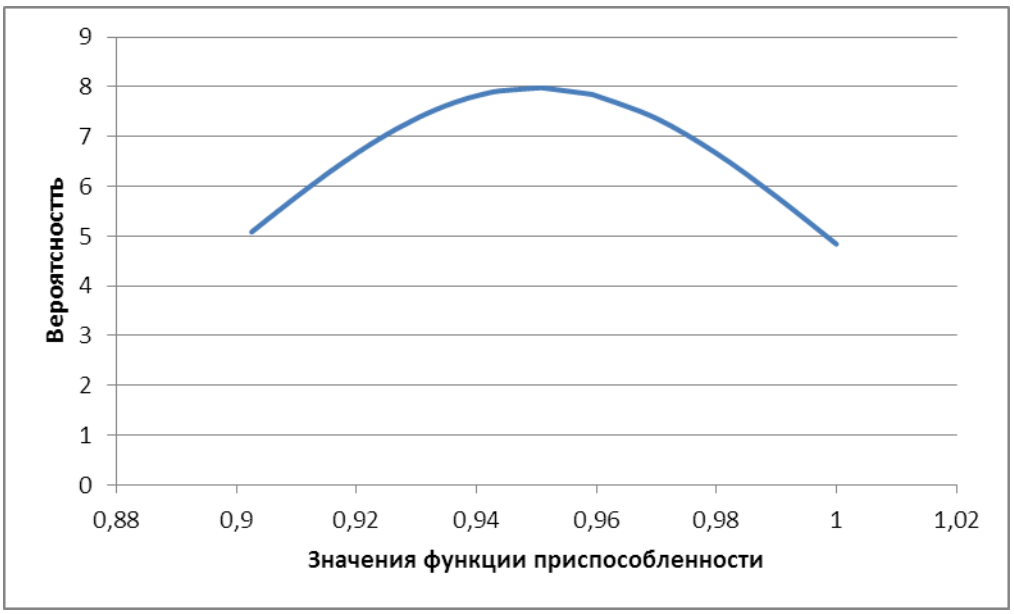

Рис. 9. Функция плотности вероятности.

Следующим шагом метода является построение функции распределения вероятности на основе распределения плотности (рис. 10).

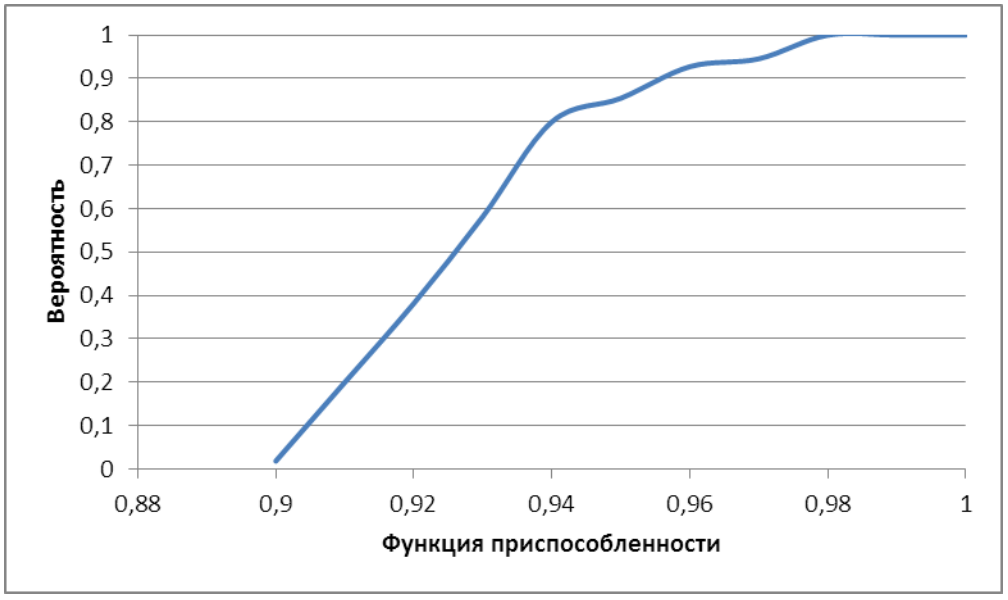

Рис. 10. Функция распределения плотности вероятности для функции приспособленности

Далее применяя метод стохастического доминирования становится возможным сравнить алгоритмы по выбранным критериям (на рис. 11) приведено сравнение для критерия – функции приспособленности).

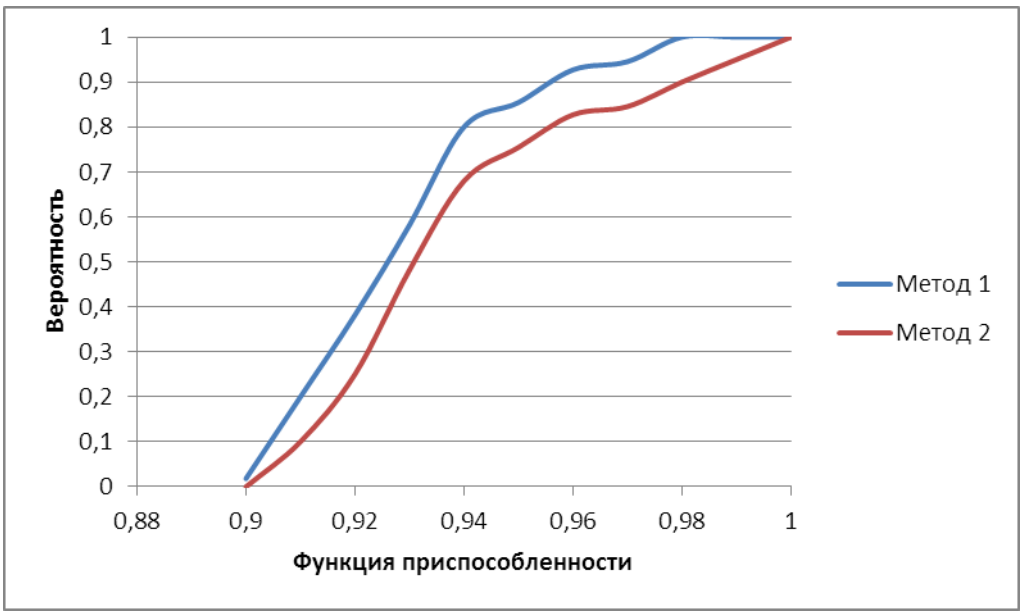

Рис 11. Сравнение алгоритмов с помощью метода стохастического доминирования.

3.2. Исследование метода на задачах, полученных случайным образом

Экспериментальное исследование разработанного метода проводилось на задачах построения управляющего конечного автомата для сгенерированного случайным образом эталонного автомата. На каждом этапе исследования выполнялись следующие действия.

- Генерация эталонного автомата с заданным количеством состояний, входных и выходных воздействий, охранных условий.
- Генерация набора сценариев работы по эталонному автомату.
- Запуск эволюционного алгоритма с использованием полученного набора сценариев.

При генерации эталонного автомата использовались следующие параметры.

- Число различных входных воздействий равно трем.
- Число переменных, использующихся в охранных условиях равно двум.

- Число различных выходных воздействий равно трем. При этом на одном переходе автомата одновременно может находиться не более двух выходных воздействий.
- Процент сгенерированных переходов от их максимально возможного количества для автомата с заданным числом состояний – 25%.

При генерации сценариев работы по эталонному автомату использовались следующие параметры.

- Минимальная длина сценария равна двум элементам.
- Максимальная длина сценариев и общее число элементов сценариев зависит от количества состояний эталонного автомата.

Сравнение эволюционных алгоритмов проводилось на наборе задач, условия которых представлены в табл. 3.

Таблица 3. Условия для задач, полученных случайным образом

Число состояний эталонного автомата	Общее число элементов сценариев
5	20
7	30
10	40
15	60
20	80
25	100

Набор тестовых сценариев для эталонного автомата должен максимально покрывать возможные переходы, кроме того, сценарии должны иметь достаточную длину – в противном случае сгенерированный автомат, хотя и будет удовлетворять сценариям, получится не эквивалентным эталонному. Пример подобного результата приведен на рис. 13, 14 – хотя полученный автомат имеет функцию приспособленности равную единице, он не позволяет выполнить все переходы эталонного.

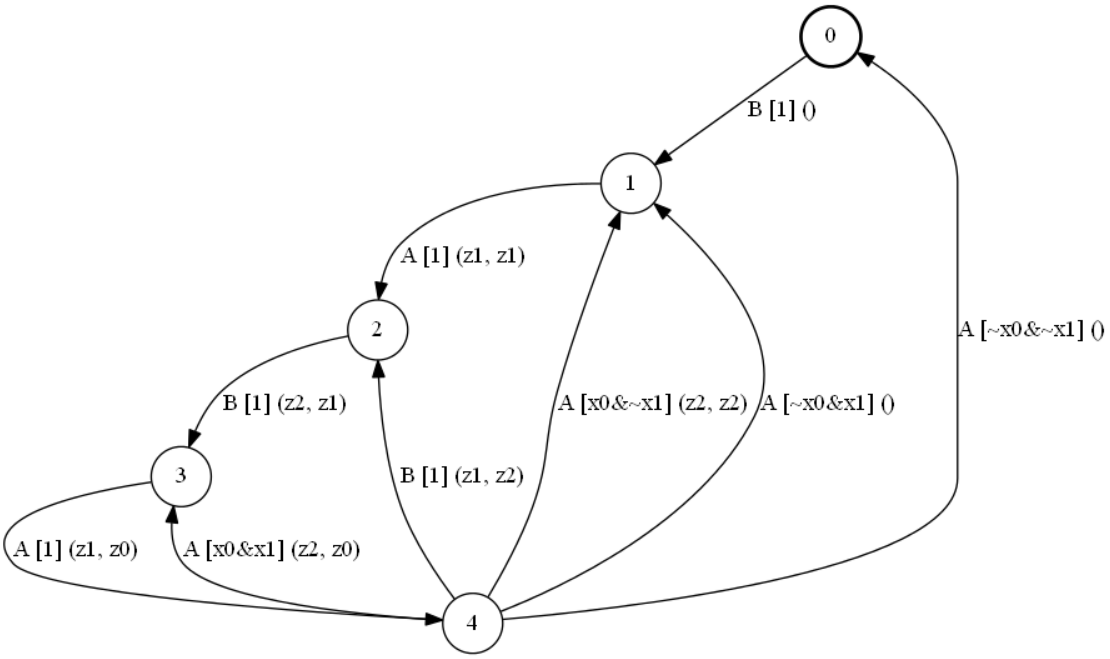

Рис. 13. Пример эталонного автомата из пяти состояний

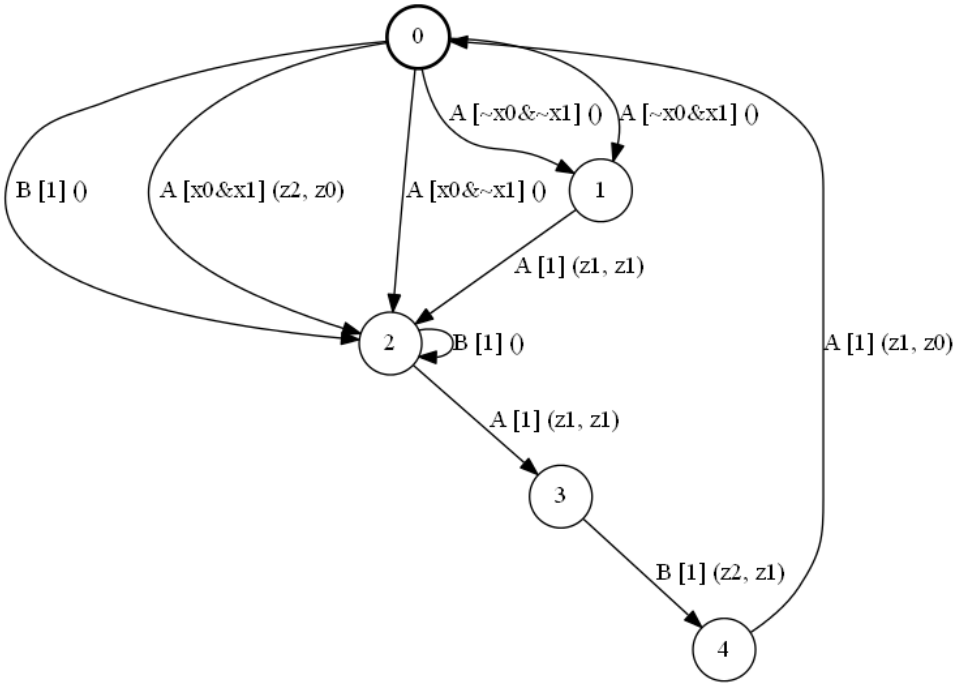

Рис. 14. Пример полученного автомата из пяти состояний с ff=1

Табл. 4. Сценарии работы для автоматов на рисунках 13, 14

Входные воздействия и охранные условия	Выходные воздействия
B[1]; A[1]; B[1]; A[1]; A[x0&x1]	; z1, z1; z2, z1; z1, z0; z2, z0
B[1]; A[1]; B[1]; A[1]; A[~x0&x1]; A[1]	; z1, z1; z2, z1; z1, z0; ; z1, z1
B[1]; A[1]; B[1]	; z1, z1; z2, z1
B[1]; A[1]; B[1]; A[1]	; z1, z1; z2, z1; z1, z0
B[1]; A[1]	; z1, z1
B[1]; A[1]; B[1]	; z1, z1; z2, z1

Рассматриваемые алгоритмы на выбранном наборе задач ведут себя схожим образом. На рис. 15 приведены результаты сравнения генетического алгоритма и эволюционной стратегии по величине функции приспособленности при построении автомата из семи состояний и суммарной длине сценариев равной 30 элементам. Как видно из графика, функций распределения вероятности получения функции приспособленности – генетический алгоритм показывает в среднем чуть лучшие показатели, нежели эволюционная стратегия. Похожие результаты наблюдаются и на задачах большой размерности – график распределения случайной величины для функции приспособленности эволюционного алгоритма всегда оказывался левее графика для генетического алгоритма. Сравнение по времени (рис. 16) выполнения методов также выявило небольшое преимущество генетического алгоритма.

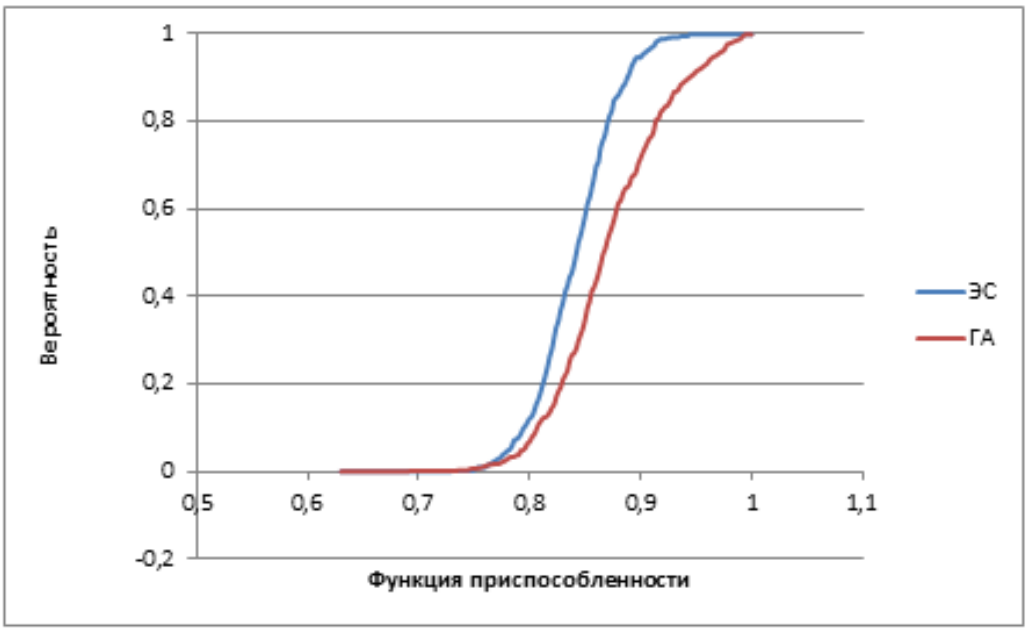

Рис. 15. Распределение случайной величины функций приспособленности ГА и ЭС, семь состояний

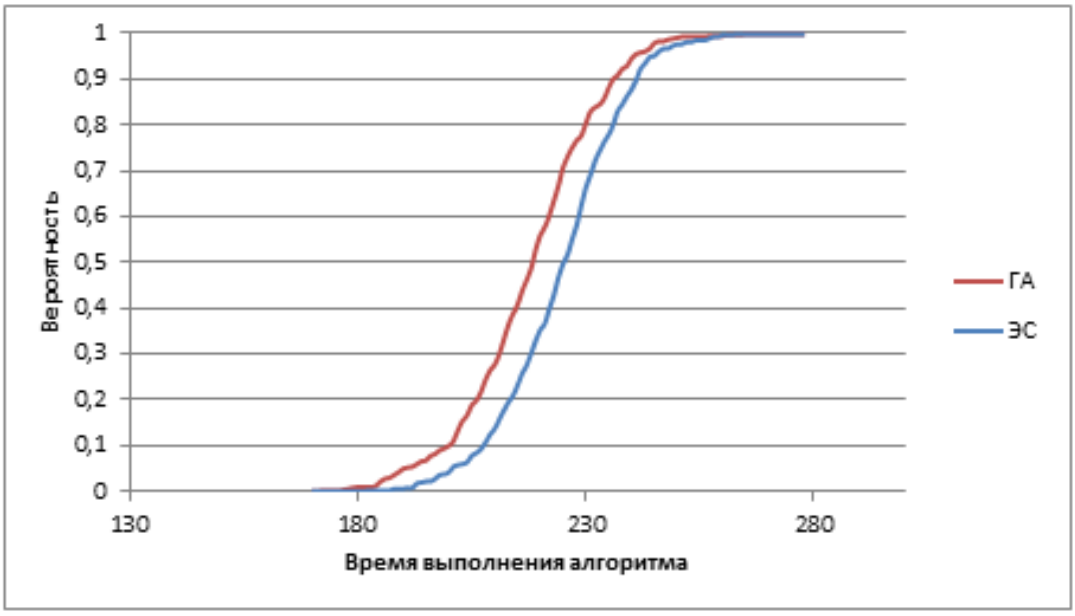

Рис. 16. Распределение случайной величины времени выполнения ГА и ЭС, семь состояний

Аналогичные результаты были получены также и для больших размерностей задачи, полученной образом. На рис. 17, 18 приведены графики сравнения функций распределений алгоритмов для критериев функции приспособленности и времени при построении автомата из десяти состояний и набора сценариев длиной не более 40 элементов.

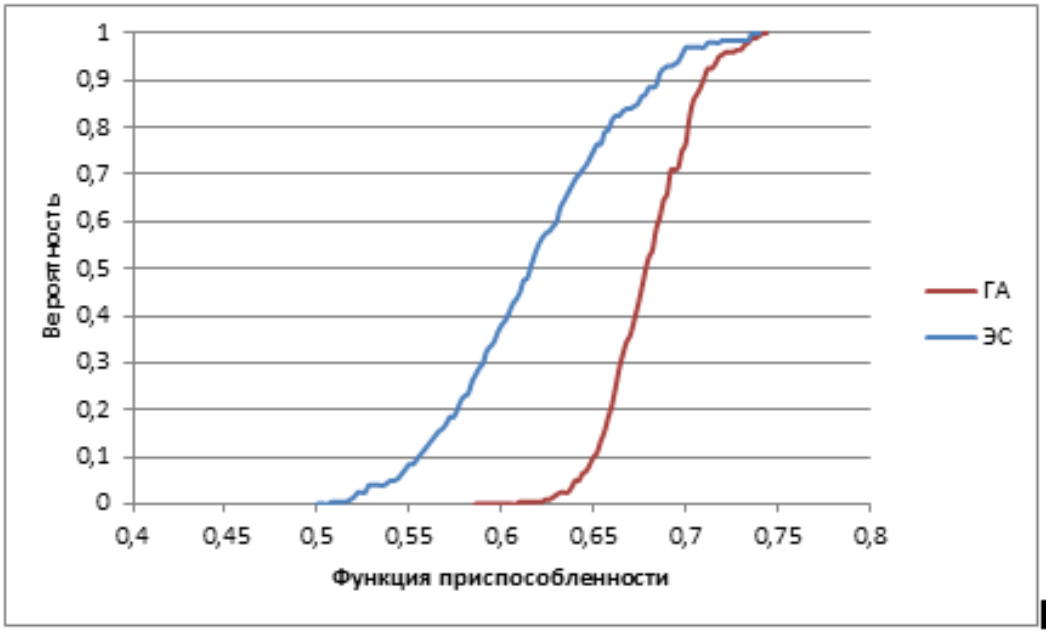

Рис. 17 Распределение случайной величины функций приспособленности ГА и ЭС, 10 состояний

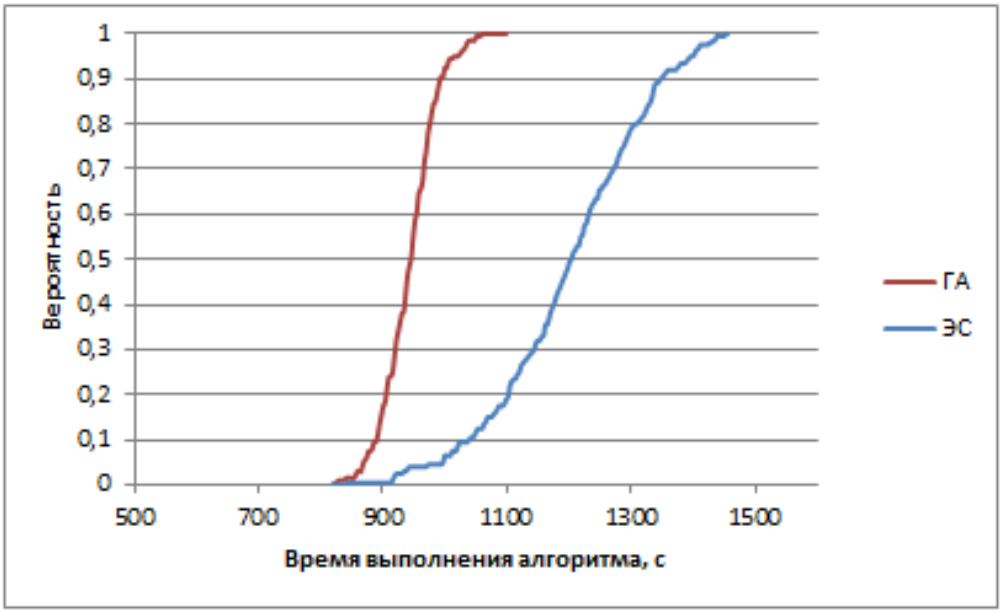

Рис. 18. Распределение случайной величины времени выполнения ГА и ЭС, 10 состояний

3.3. Построение автомата управления часами с будильником

Задача о построении автомата управления часами с будильником подробно описана в работе [7]. Система, соответствующая задаче, описывается четырьмя входными воздействиями, двумя входными переменными и семью выходными воздействиями.

Реализованные методы позволили построить автомат, состоящий из трех состояний и 14 переходов и удовлетворяющий всем сценариям тестового набора. Время работы генетического алгоритма, потребовавшееся для нахождения лучшего решения составило порядка 11 минут на компьютере с процессором *Intel Core i5-2520M*. Эволюционной стратегии для достижения аналогичного результата потребовалось порядка 14 минут. Большой временной показатель для поиска решения, превышающей время, требующееся на поиск аналогичного решения в работах [7,10] объясняется сложностью расчета функции приспособленности, требующей для получения результата двух проходов по набору сценариев работы автомата.

3.4. Применение многокритериальной оптимизации

Зачастую эволюционные алгоритмы при поиске решения находят точку локального максимума искомой функции, выход из которой может быть затруднительным. Связана эта сложность, прежде всего, с необходимостью пройти путь в поисках глобального максимума через соседние точки области значений функции, решения в которых могут быть хуже текущего. Существуют различные подходы для решения этой проблемы. Например, можно увеличить число операций мутации, выполняемых над текущим поколением решений эволюционного алгоритма. Это позволяет расширить радиус поиска решений, находящихся вблизи точки найденного локального максимума. Другим подходом является использование многокритериальной оптимизации.

Многокритериальная оптимизация (*multiobjective optimization*) [22] – процесс одновременной оптимизации двух или более конфликтующих целевых функций в заданной области определения. Суть метода заключается в добавлении дополнительных критериев, по которым оценивается решение. В настоящей работе помимо значения функции приспособленности рассматривался также критерий числа переходов в построенном автомате. Благодаря переходу от однокритериальной оптимизации к многокритериальной

становится возможным сравнивать решения с одинаковыми значениями функции приспособленности, считая лучшими автоматы с меньшим числом переходов. Формально в случае однокритериальной оптимизации требуется максимизировать значение функции *f(x),* $x \in X$, где x – дискретный вектор решения, а X – множество допустимых решений. В случае же многокритериальной оптимизации задача принимает вид максимизации функции $f(x) = f_1(x), \ldots, f_k(x), x \in X$, где $k \geq 2$ – число рассматриваемых критериев.

Для сравнения найденных решений между собой в случае многокритериальной оптимизации используется критерий оптимальности *Парето* [13, 22]. В соответствие с этим критерием найденное решение x считается лучше (доминирует по Парето) другого решения x', в том и только том случае, если для всех сравнительных критериев x не хуже решения x', а по одному из них превосходит x'.

Таким образом, для всех построенных автоматов текущего поколения становится возможным построение фронта Парето – в случае k-критериев, k-мерной области, все решения снаружи которой являются доминирующими по Парето. В случае если вновь полученное в ходе шага мутации эволюционного алгоритма решение лежит снаружи фронта Парето, оно допускается для выполнения дальнейших шагов алгоритма, если же решение не удовлетворяет этому условию – на очередных этапах алгоритма оно не используется. На рис. 19 представлено изменение фронта Парето в зависимости от номера текущего поколения автоматов в генетическом алгоритме. С течением времени фронт смещается в сторону решений с большими значениями функции приспособленности и меньшим числом переходов.

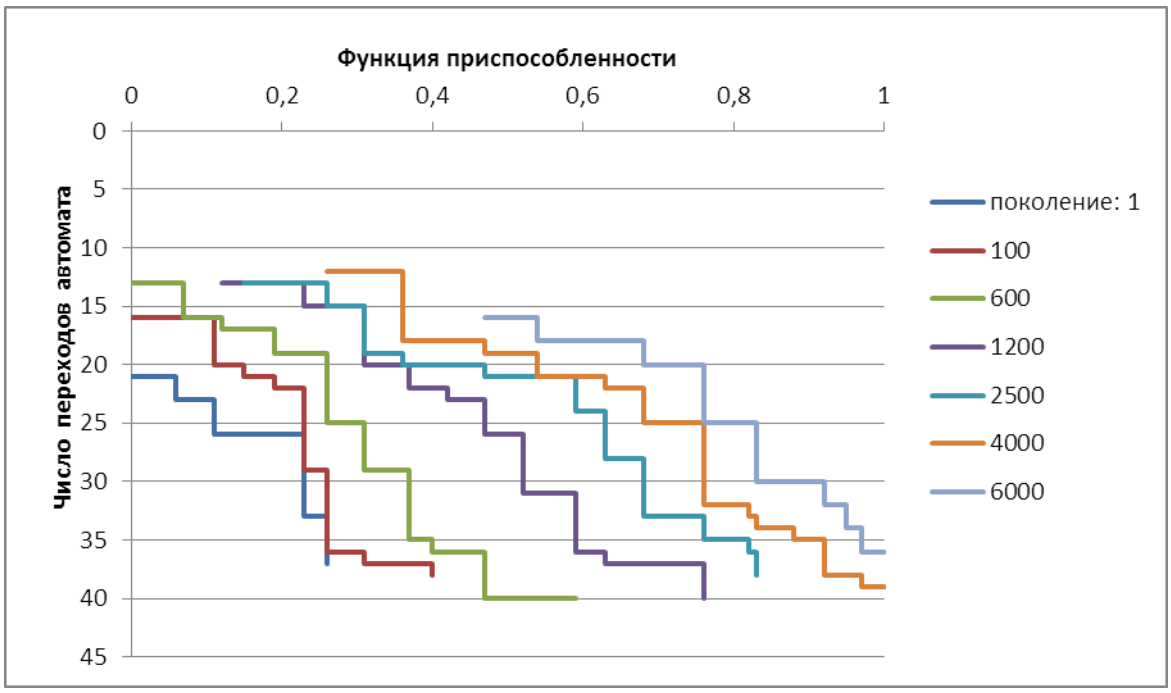

Рис. 19. Фронт Парето для выбранных поколений генетического алгоритма

Применительно к задаче о построении управляющих конечных автоматов дополнительным критерием также может выступать число состояний полученного автомата. Лучшим решением при равных значениях функции приспособленности при использовании данного подхода будет считаться автомат с меньшим числом состояний. В настоящей работе анализ числа состояний автомата в качестве сравнительного критерия не проводился, так как в рамках одного запуска алгоритма все автоматы строились с одинаковым числом состояний.

Использование в качестве сравнительного критерия числа переходов в построенном автомате для генетического алгоритма позволило увеличить скорость нахождения автомата, проходящего все сценарии эталонного тестового набора. Подобный результат можно объяснить тем, что среднее число переходов среди автоматов одного поколения уменьшилось, автоматы с неиспользуемыми переходами реже отбирались для использования в следующих поколениях – как следствие операция мутация стала работать более эффективно, стали чаще изменяться входные воздействия и охранные условия на переходах, задействованных при прохождении сценариев работы.

На рис. 20 приведен пример сравнения распределений времени работы простого генетического алгоритма и алгоритма с применением многокритериальной оптимизации (МКО) при решении задачи, полученной случайным образом для построения автомата из семи состояний.

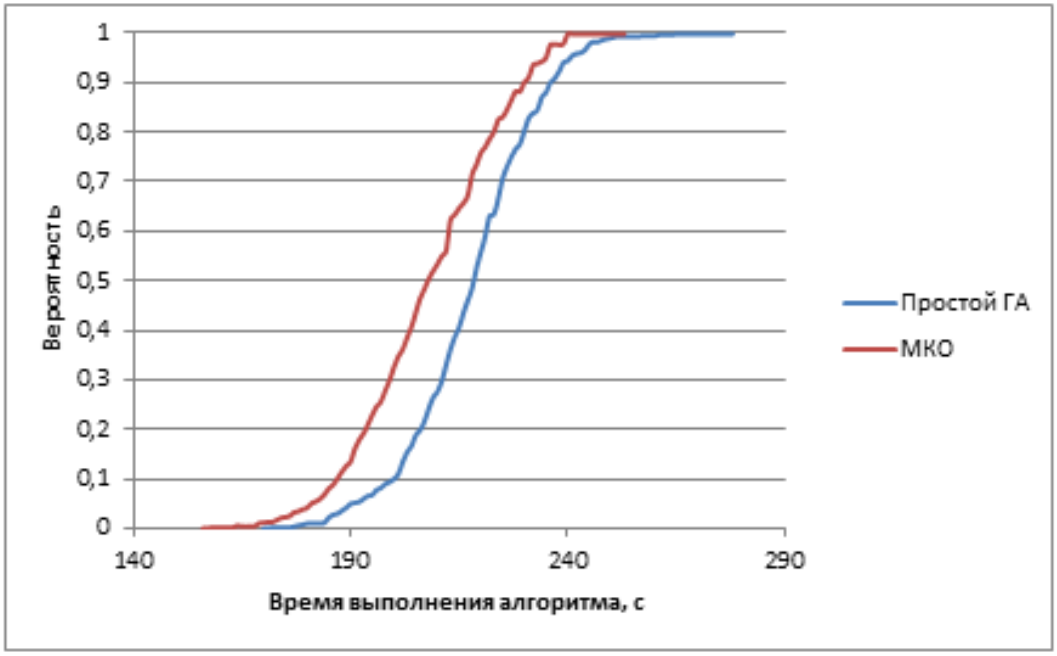

Рис. 20. Распределение случайной величины времени выполнения ГА и ГА+МКО

Больший прирост к эффективности алгоритма многокритериальная оптимизация обеспечивает для автоматов с большим числом состояний и средним количеством переходов, что также является закономерным – для небольших автоматов значительное количество решений, имеющих одинаковые значения функции приспособленности, также имеет небольшой разброс и по числу переходов.

В случае применения многокритериальной оптимизации для эволюционной стратегии заметных изменений в скорости нахождения решения выявлено не было. Объяснить этот факт можно меньшей по сравнению с генетическим алгоритмом процентом мутирующих особей, а также отсутствием этапа периодичной большой мутации.

Выводы по главе 3

1. В данной главе была выбрана и описана методология сравнения эволюционных алгоритмов, использующихся для построения конечных автоматов по сценариям работы.
2. Приведен пример использования метода главных компонент на данных, полученных в результате построения автомата из пяти состояний по сценариям из 35 элементов.
3. Выполнено сравнение генетических алгоритмов и эволюционных методов на задаче о часах с будильником и задачах, полученных случайным образом. Сделаны выводы об эффективности данных методов применительно к реальным задачам.
4. Предложен и реализован метод многокритериальной оптимизации для использования совместно с эволюционными алгоритмами. Проведено исследование влияния многокритериальной оптимизации на эффективность работы эволюционных алгоритмов, использующихся для построения автоматов по сценариям работы.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены существующие на данный момент методы машинного обучения для построения управляющих конечных автоматов. В результате анализа существующих подходов были выявлены их достоинства и недостатки с учетом оценки критериев эффективности применения методов для различного набора задач, времени работы алгоритмов и сложности их реализации.

Предложен новый метод построения управляющих конечных автоматов по сценариям работы. Данный метод позволяет использовать уже существующие эволюционные алгоритмы, для которых также предложен способ реализации основных алгоритмических этапов

Для апробации метода были отобран и реализованы генетический алгоритм и эволюционная стратегия, сравнение которых проводилось на двух задачах. Оба алгоритма успешно находят решения рассмотренных задач. Лучшие результаты с точки зрения найденного решения и времени выполнения показал генетический алгоритм. По сравнению с существующими методами построения конечных автоматов предложенный метод, основанный на использовании сценариев работы проигрывает по показателям времени работы из-за значительных затрат, требующихся на вычисление функции приспособленности. Как следствие, в качестве направления для дальнейшего исследования можно выделить работу над оптимизацией вычисления функции приспособленности и, как следствие, увеличения эффективности работы метода. Указанный недостаток компенсируется относительной простотой реализации метода, что делает его использование оправданным для задач, требующих больших вычислительных ресурсов для моделирования функции приспособленности.

Также был предложен и реализован алгоритм, использующий многокритериальную оптимизацию для построения автоматов с помощью эволюционных алгоритмов. Для некоторых частных случаев задачи – при

наличии большого числа переходов в эталонном автомате, для методов с большим процентом мутирующих особей, применение многокритериальной оптимизации является целесообразным.

Таким образом, поставленные в ходе работы задачи были полностью реализованы.

ИСТОЧНИКИ

1. *Бедный Ю.Д., Шалыто А.А.* Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/_ant.pdf
2. *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы: Учебное пособие. М.: Физматлит, 2006.
3. *Поликарпова Н. И., Шалыто А. А*. Автоматное программирование. СПб. Питер, 2009.
4. *Рассел С.*, Норвиг П. Искусственный интеллект. Современный подход. М.: Вильямс, 2006.
5. *Соколов Д.О., Давыдов А.А., Царев Ф.Н. и др.* Виртуальная лаборатория обучения генетическому программированию для генерации управляющих конечных автоматов. М.: МАКС Пресс, 2008, http://is.ifmo.ru/works/_2_93_davidov_sokolov.pdf
6. *Тяхти А.С.* Сравнение генетических алгоритмов и методов имитации отжига на примере построения управляющих конечных автоматов с использованием виртуальной лаборатории. СПбГУ ИТМО, 2010.
7. *Царев Ф.Н.* Разработка метода совместного применения генетического и автоматного программирования. СПбГУ ИТМО, 2009.
8. *Цветков А.А.* Сравнение поведенческих и эволюционных алгоритмов построения управляющих конечных автоматов. СПбГУ ИТМО, 2010.
9. *Шалыто А. А.* SWITCH технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998, http://is.ifmo.ru/books/switch/1
10. *Ульянцев В.И.* Применение методов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов по сценариям работы, СПбГУ ИТМО, 2011.
11. *Angeline P. J., Pollack J.* Evolutionary Module Acquisition Proceedings of the Second Annual Conference on Evolutionary Programming, 1993. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
12. *Beck K.,* Test-Driven Development: By Example. Addison-Wesley. Winner of the Jolt Productivity Award, 2002.
13. *Brockhoff D., Friendrich T., Neumann F.,* Analyzing Hypervolume Indicator Based Algorithms. Springer-Verlag Berlin, Heidelberg, 2008.

14. *Eduardo G. Carrano, Ricardo H. C. Takahashi, Elizabeth F. Wanner*. An Enhanced Statistical Approach for Evolutionary Algorithm Comparison. ACM. 2008.
15. *Hans-Georg Beyer, Hans-Paul Schwefel.* Evolution strategies. A comprehensive introduction. Kluwer Academic Publishers, 2002.
16. *Ingber A.L.* Simulating Annealing: Practice versus theory. Mathl. Comput. Modelling, 1993.
17. *Koza J.* Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, 1992.
18. *Koza J.* Genetic Programming II: Automatic discovery of Reusable Programs, California, Massachusetts Institute of Technology, 1994.
19. *Lawrence J. Fogel, Peter J. Angeline, David B. Fogel.* An Evolutionary Programming Approach to Self-Adaptation on Finite State Machines.
20. *Metropolis,N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller,* "Equation of State Calculations by Fast Computing Machines", J. Chem. Phys., 1953
21. *Kirkpatrick, S., C. D. Gelatt Jr., M. P. Vecchi,* "Optimization by Simulated Annealing",*Science,* 1983.
22. *Knowles J., Watson R., Corne D.* Reducing Local Optima in Single-Objective Problems by Multi-objectivization, Brandeis University, Waltham, MA 02454, USA, 2001

ПРИЛОЖЕНИЕ. ИСХОДНЫЕ КОДЫ МЕТОДА ГЛАВНЫХ КОМПОНЕНТ НА ЯЗЫКЕ ПАКЕТА MAPLE

```
with(Statistics);
with(linalg);
with(LinearAlgebra);

#Загрузка исходных данных
A := Matrix(<<input data>>);

#Нахождение среднего для критериев
A2 :=Matrix(rowdim(A),coldim(A));
x := Vector(coldim(A));
for i to coldim(A) do
     x[i] := Mean(col(A, i))
end do;

#Преобразование исходных данных с учетом найденных значений
#средних
for i to rowdim(A) do
     for j to coldim(A) do
          A2[i, j] := A[i, j]-x[j]
     end do
end do;

#Вычисление матрицы ковариации
A3:=Transpose(A2);
covAtest :=(1/rowdim(A))*MatrixMatrixMultiply(A3,A2);
covA := CovarianceMatrix(A2,ignore);

#Нахождение собственных векторов матрицы ковариации
ev,EV:=Eigenvectors(covAtest);
EV:=simplify(EV);
```

```
#Получение набора данных с нулевой корелляцией
s:=(-1)*MatrixMatrixMultiply(A,EV);

#количество выборок с повторениями
selCount := 5000;
#количество элементов, участвующих в одной выборке
inSelCount := 200;

#Осуществление выборок с повторением для построения функции
#плотности вероятности
res:=Matrix(selCount,2);
for i from 1 to selCount do
  for j from 1 to inSelCount do
     res[i,1]:=res[i,1]
     +s[RandomTools[Generate](integer(range = 1 .. rowdim(s))),1];
     res[i,2]:=res[i,2]
     +s[RandomTools[Generate](integer(range = 1 .. rowdim(s))),2];
  end do;
  res[i,1]:=res[i,1]/inSelCount;
  res[i,2]:=res[i,2]/inSelCount;
end do;
```