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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему Исследование применимости генетического алгоритма для решения поисковых задач

Студент Р. И. Семенов

Руководитель О. В. Лелонд

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

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

Институт **математики, физики и информационных технологий**
Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Семенов Р.И. ПМИб-1201
1. Тема Исследование применимости генетического алгоритма для решения поисковых задач
2. Срок сдачи студентом законченной выпускной квалификационной работы 15.06.2016
3. Исходные данные к выпускной квалификационной работе: классический генетический алгоритм, примеры задач.
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов): необходимые теоретические сведения, NP-полные задачи и их решение, генетический алгоритм как способ решения NP-полных задач, вариативный генетический алгоритм, многомерная оценка, вариативная фитнесс-функция, устройство вариативного генетического алгоритма, реализация генетических алгоритмов, особенности реализации вариативного генетического алгоритма, достоинства применения вариативного генетического алгоритма, сравнительный анализ производительностей классического и вариативного генетический алгоритмов.
5. Ориентировочный перечень графического и иллюстративного материала: схемы алгоритмов, презентация.
6. Дата выдачи задания «11» января 2016 г.

Руководитель выпускной
квалификационной работы _____ О.В. Лелонд

Задание принял к исполнению _____ Р.И. Семенов

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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Семенова Р.И.
по теме Исследование применимости генетического алгоритма для решения поисковых задач

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Выбор и утверждение темы ВКР	13.01.2016	13.01.2016	выполнено	
Поиск литературы	16.01.2016	16.01.2016	выполнено	
Анализ задач	25.01.2016	25.01.2016	выполнено	
Оформление пояснительной записки ВКР	01.06.2016	01.06.2016	выполнено	
Введение	15.02.2016	15.02.2016	выполнено	
1. Необходимые теоретические сведения	01.03.2016	01.03.2016	выполнено	
1.1 NP-полные задачи и их решение	20.02.2016	20.02.2016	выполнено	
1.2 Генетический алгоритм как способ решения NP-полных задач	25.02.3016	25.02.3016	выполнено	
1.3 Трудности применения генетического алгоритма	01.03.2016	01.03.2016	выполнено	
2. Генетический алгоритм с вариативными компонентами	05.03.2016	05.03.2016	выполнено	
2.1 Описание генетического алгоритма с вариативными	02.03.2016	02.03.2016	выполнено	

компонентами				
2.2 Многомерная оценка	03.03.2016	03.03.2016	выполнено	
2.3 Вариативная фитнесс-функция	04.03.2016	04.03.2016	выполнено	
2.4 Иерархия фитнесс-функций	05.03.2016	05.03.2016	выполнено	
3. Разработка требований к реализации вариативного генетического алгоритма	15.03.2016	15.03.2016	выполнено	
3.1 Разнообразие искомых объектов и атрибутов	10.03.2016	10.03.2016	выполнено	
3.2 Трудности практического применения генетического алгоритма	12.03.2016	12.03.2016	выполнено	
3.3 Общие требования к вариативному генетическому алгоритму	14.03.2016	14.03.2016	выполнено	
3.4 Описание требований к реализации вариативного генетического алгоритма	15.03.2016	15.03.2016	выполнено	
4. Процесс реализации вариативного генетического алгоритма	22.03.2016	22.03.2016	выполнено	
4.1 Каркасная архитектура вариативного генетического алгоритма	17.03.2016	17.03.2016	выполнено	
4.2 Варианты популяций вариативного генетического алгоритма	18.03.2016	18.03.2016	выполнено	
4.3 Очередь подзадач вариативного генетического алгоритма	19.03.2016	19.03.2016	выполнено	
4.4 Конфигурация вариативного генетического алгоритма	20.03.2016	20.03.2016	выполнено	
4.5 Параллельные операции вариативного генетического алгоритма	21.03.2016	21.03.2016	выполнено	
4.6 Выбор индивидов из популяции	22.03.2016	22.03.2016	выполнено	
5 Вариант реализации вариативного генетического алгоритма на Java	01.04.2016	01.04.2016	выполнено	
5.1 Вариант реализации каркаса вариативного генетического алгоритма	25.03.2016	25.03.2016	выполнено	
5.2 Вариант реализации вариативного генетического алгоритма для поиска точек минимума функции на заданном отрезке	28.03.2016	28.03.2016	выполнено	

5.3 Вариант реализации вариативного генетического алгоритма для поиска минимального вершинного покрытия графа	01.04.2016	01.04.2016	выполнено	
Разработка презентации к защите	10.04.2016	10.04.2016	выполнено	
Проверка ВКР на антиплагиат	10.04.2016	10.04.2016	выполнено	
Предзащита ВКР	05.06.2016	31.05.2016	выполнено	
Корректировка ВКР	10.06.2016	10.06.2016	выполнено	
Оформление ВКР и презентации	10.06.2016	10.06.2016	выполнено	
Сдача пояснительной записки	15.06.2016	15.06.2016	выполнено	

Руководитель выпускной квалификационной работы _____ О.В. Лелонд

Задание принял к исполнению _____ Р.И. Семенов

Аннотация

На бакалаврскую работу студента Семенова Р.И. на тему «Исследование применимости генетического алгоритма для решения поисковых задач».

Объектом исследования бакалаврской работы является генетический алгоритм и его модификации.

Предмет исследования – особенности работы генетического алгоритма, находящего решение поисковой задачи.

Бакалаврская работа состоит из введения, пяти глав, заключения, списка литературных источников и приложения.

Во введении обосновывается актуальность существующих проблем в сфере применения генетических алгоритмов, ставится цель, определяются задачи.

В первой главе изучаются вопросы, связанные с задачами, которые решает генетический алгоритм, а также указываются трудности применения классического генетического алгоритма.

Во второй главе предлагается модифицированная версия генетического алгоритма, призванная расширить сферу применения генетических алгоритмов.

В третьей главе рассматриваются факторы, ограничивающие практическое применение генетического алгоритма. Также приводятся требования к модифицированной версии генетического алгоритма.

В четвертой главе рассматриваются способы реализации модифицированной версии генетического алгоритма, а также описывается метод удобной конфигурации модифицированного генетического алгоритма.

В пятой главе приводится описание реализации модифицированной версии генетического алгоритма на языке Java, а также рассматриваются примеры и результаты практического применения данного алгоритма.

В заключении формулируются основные результаты исследования, делаются окончательные выводы.

В списке источников перечисляются используемые материалы не ранее 2007 года.

В приложении приводится диаграмма классов, реализующих базовую часть модифицированной версии алгоритма на языке Java.

Объем бакалаврской работы – 45 страниц, она содержит 5 рисунков, 22 источника литературы.

Оглавление

Введение

Широкое применение информационных технологий подтолкнуло к их значительному развитию. Современные вычислительные системы позволяют автоматизировать множество процессов, создавая, трансформируя и передавая различные данные. Одной из наиболее популярных задач, поставленных перед информационными системами, является получение новых данных, основанных на анализе уже имеющейся информации, либо сгенерированных по определенным условиям. Мощность современной вычислительной техники позволяет производить моделирование различных сложных процессов (например, моделирование возможных климатических изменений для метеопрогнозов), избавляя от необходимости создания специальных методов решения и формул. Несмотря на наличие мощного математического аппарата, до сих пор остаются задачи, *точное* решение которых осуществимо лишь при полном переборе всех возможных ответов с последующим отсевом неправильных вариантов, например, именно таким образом происходит решение NP-полных задач. С другой стороны, точное решение таких задач требуется не всегда, а потому производится поиск *подходящего* ответа. В данной работе рассмотрен один вариант генетического алгоритма, позволяющего создавать требуемые структуры данных на основе информации о свойствах этих структур.

Цель работы: исследовать применимость генетического алгоритма для решения поисковых задач, найти способы преодоления препятствий, затрудняющих применение генетического алгоритма как средства решения поисковых задач.

Объект исследования: генетический алгоритм и его модификации.

Предмет исследования: особенности работы генетического алгоритма, находящего решение поисковой задачи.

Глава 1 Необходимые теоретические сведения

1.1 NP-полные задачи и их решение

Представим процесс решения NP-полной задачи как процесс поиска. Решение NP-полной задачи с данной точки зрения представляет собой выполнение цикла из двух операций: генерации ответа и проверки сгенерированной структуры данных на пригодность в качестве ответа. Как правило, такое решение основано на полном переборе, когда попытка генерации ответа представляет собой создание очередной комбинации значений атрибутов искомого объекта. В случае, когда требуется отыскать один ответ, процесс поиска завершается после нахождения первого подходящего решения, иначе поиск продолжается до тех пор, пока не закончатся комбинации всех возможных решений.

Рассмотрим процесс поиска решения задачи о минимальном вершинном покрытии графа (поиске множества вершин, охватывающих все ребра графа). Данная задача является NP-полной, то есть на данный момент неизвестны алгоритмы, способные дать точное решение поставленной задачи за полиномиальное время. Это означает, что для нахождения минимального вершинного покрытия придется найти все варианты вершинных покрытий и выбрать из них наименьшее.

Однако стоит обратить внимание на тот факт, что конкретная NP-полная задача может иметь сразу несколько верных ответов. К тому же точное решение NP-полной задачи требуется не всегда: иногда пользователю достаточно частичного выполнения условий решения. Например, требуется поиск как можно меньшего вершинного покрытия за ограниченное время, где решающим фактором является время работы алгоритма, а не качество полученного решения [14]. То есть в некоторых ситуациях мы по условию имеем возможность пожертвовать качеством полученного решения с целью уменьшения временных и аппаратных затрат. Также может возникнуть ситуация, когда пользователь сам бы хотел контролировать зависимость между

качеством решения и затраченным на его получение время. Например, если пользователь желает получить качественный результат, то дает достаточное время на поиск.

Таким образом, NP-полную задачу можно решить двумя способами: точно методом полного перебора и неточно альтернативными методами (предполагается, что альтернативный метод будет работать быстрее, чем полный перебор, но в ущерб точности решения). В рамках данной работы в качестве альтернативного метода предлагается генетический алгоритм.

1.2 Генетический алгоритм как способ неточного решения NP-полных задач

Генетический алгоритм – алгоритм, моделирующий генетический эволюционный процесс. Генетический алгоритм осуществляет *направленный* поиск, то есть он способен *оценить* качество текущих результатов, чтобы выполнить коррекцию критериев поиска. Таким образом, генетический алгоритм позволяет быстро найти неточное решение NP-полной задачи.

Классический генетический алгоритм включает в себя следующие операции [2]:

- создание начальной популяции;
- отбор;
- выбор родителей;
- скрещивание;
- мутации.

Начальная популяция генетического алгоритма, как правило, генерируется при помощи известных или упрощенных алгоритмов. Наиболее известной реализацией генератора стартовых индивидов является алгоритм, выставляющий атрибуты нового объекта случайным образом. Очевидно, что такие индивиды могут не подходить в качестве решения задачи (правда, стоит отметить, что существует некоторая вероятность *угадать* подходящее решение уже на этом этапе). Начальная популяция создается один раз в начале работы

генетического алгоритма с расчетом на то, что со временем ее заменят более приспособленные индивиды.

Отбор удаляет из памяти наименее подходящие в качестве решения элементы («убивает» наименее приспособленные индивиды), тем самым обеспечивая сходимость поиска решения. Корректно реализованный оператор отбора обеспечивает приемлемый расход памяти для хранения индивидов в популяции, при этом сохраняя необходимое генное разнообразие популяции [1], без которого работа генетического алгоритма не приведет к получению качественных результатов.

Оператор выбора родителей осуществляет определенным образом выборку из популяции таких индивидов, на основе которых будет создан новый индивид-потомок. Выбор наиболее приспособленных родителей обеспечивает определенную направленность поиска решения. При этом следует учитывать, что частый выбор одних и тех же индивидов в качестве родителей приведет к обеднению генного разнообразия популяции.

Скрещивание осуществляет сборку нового элемента популяции на основе выбранных родителей. Предполагается, что дочерний индивид примет положительные черты своих родителей и станет адаптированным лучше своих родителей. Механизм скрещивания должен обеспечивать передачу генетической информации от всех родителей, чтобы задействовать наибольшее доступное генетическое разнообразие популяции. Стоит заметить, что число родителей может быть любым, хотя зачастую применяется схема из двух родителей.

Мутации служат для компенсации явления локальной сходимости, осуществляя в ходе работы случайное изменение некоторых свойств потомка. Такие случайные изменения позволяют скомпенсировать некоторые недостатки в реализации операторов выбора и скрещивания, повышая генетическое разнообразие популяции. К тому же имеется вероятность получить полезную мутацию, повысив приспособленность созданного индивида.

Рассмотрим устройство классического однопоточного генетического алгоритма:

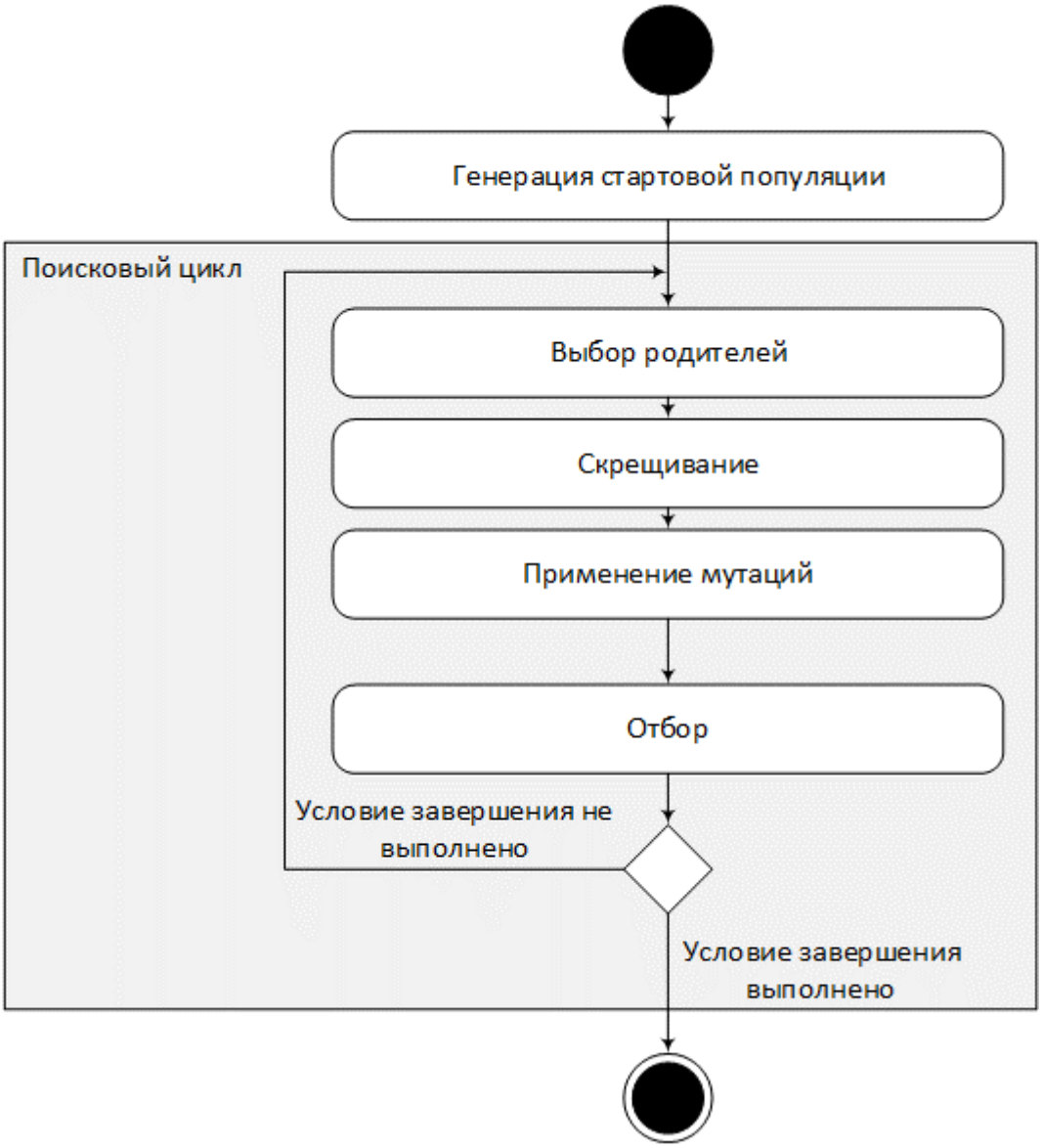

Рисунок 1.1 – описание классического генетического алгоритма

Из рисунка 1.1 можно видеть, что что основная работа генетического алгоритма будет заключаться в повторении цикла

… > **«выбор родителей»** > **«скрещивание»** > **«мутации»** > **«отбор»** > **…**

до тех пор, пока не будет получено подходящее решение.

Зачастую указанные выше функции реализуются при поддержке генератора псевдослучайных величин [3]. Так, выбрать родителей можно

случайным образом, скрещивание осуществить в случайных пропорциях, а мутациям подвергнуть случайные свойства. При этом имеется возможность контроля действий в зависимости от случайной величины, например, панмиксия позволяет каждому индивиду стать родителем с равной вероятностью, в то время как метод рулетки попытается выбрать наиболее приспособленных родителей. На рисунке 1.2 продемонстрированы примеры распределения вероятностей в зависимости от выбранного метода:

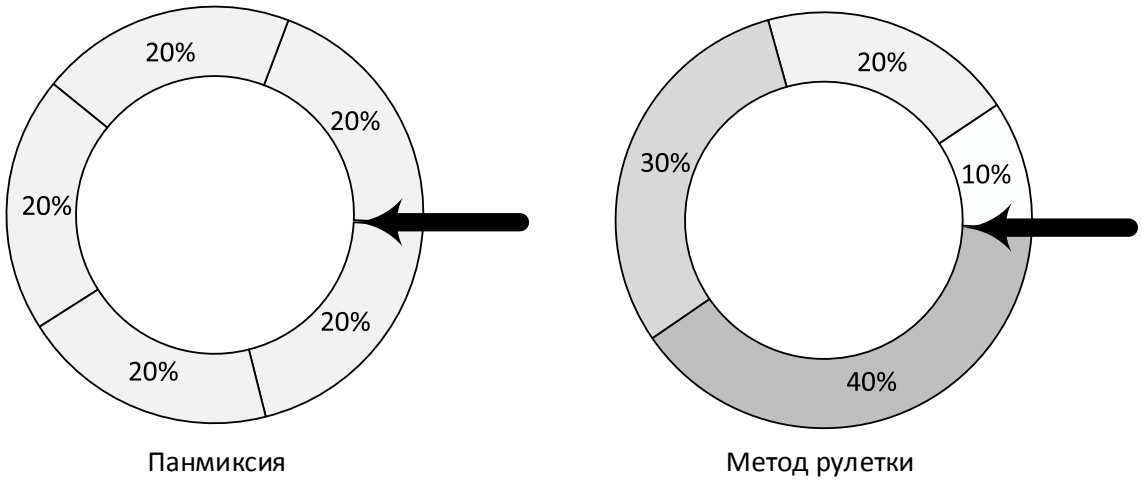

Рисунок 1.2 – способы распределения вероятности выбора индивидов

Таким образом, основная задача генетического алгоритма – найти решение задачи, постепенно улучшая его в ходе работы, проходя путь поиска от менее качественного до более качественного решения.

1.3 Трудности применения генетического алгоритма

Использование генетического алгоритма сопряжено с рядом проблем, влияющих на качество и скорость работы алгоритма. Не существует полной универсальной конфигурации генетического алгоритма, способного одинаково быстро находить наилучшие решения любой переборной задачи. Как правило, генетический алгоритм не сможет создавать качественные решения без предварительной конфигурации и специальной настройки.

В первую очередь необходимо выбрать функцию, которая давала бы оценку степени приспособленности индивида. Данная функция называется

фитнесс-функцией (от англ. fitness - пригодность). Фитнесс-функция – решающий параметр качества генерируемых решений, так как именно она задает направление поиска генетического алгоритма [5]. На основе значений фитнесс-функции идет отбор наименее приспособленных индивидов, а также может осуществляться выбор родителей (стоит заметить, что выбор родителей может быть организован случайным образом по всей популяции, а может принимать вид функции, отдающей предпочтение, например, более приспособленным особям). Если в некоторых задачах, основанных исключительно на математических операциях (например, интерполяция функций или поиск минимума на отрезке), разработка фитнесс-функции может быть сведена к решению математической задачи, то в прикладных задачах зачастую требуется оценка индивидов через методы нечеткой логики, где фитнесс-функцию зачастую приходится подбирать в ходе разработки [15].

Таким образом, фитнесс-функция должна учитывать случайную природу генетического алгоритма, а также обеспечивать наиболее возможную стабильность поиска. Стоит отметить, что функции, обеспечивающие выбор родителя, выбор точки скрещивания и выбор атрибутов для мутации, также позволяют ускорить поиск решения [4], но их выбор зависит в большей степени от конкретной задачи и структур данных, с которыми будет работать алгоритм.

Заметным недостатком генетического алгоритма является его сходимость к локальному оптимуму. Генетический алгоритм не имеет готовых механизмов полного подавления выбора кратковременного улучшения в пользу долговременного улучшения. Так как генетический алгоритм осуществляет направленный поиск, стремящийся постоянно улучшать популяцию, возникает серьезная проблема, связанная с получением глобально оптимальных решений, связанных с кратковременным ухудшением индивидов. Из-за этого генетическому алгоритму зачастую ставится упрощенная версия исходной поисковой задачи (например, вместо разработки целой системы алгоритму дают разрабатывать ее отдельные части). Стоит отметить, что решение упрощенной версии задачи позволяет привести время получения необходимого

ответа к приемлемой величине, к тому же появляется возможность значительно сократить затрачиваемые на решение аппаратные ресурсы. На рисунке 1.3 показана ситуация, при которой индивиды, поступая слева, сосредоточились на локальном максимуме, не достигнув глобального.

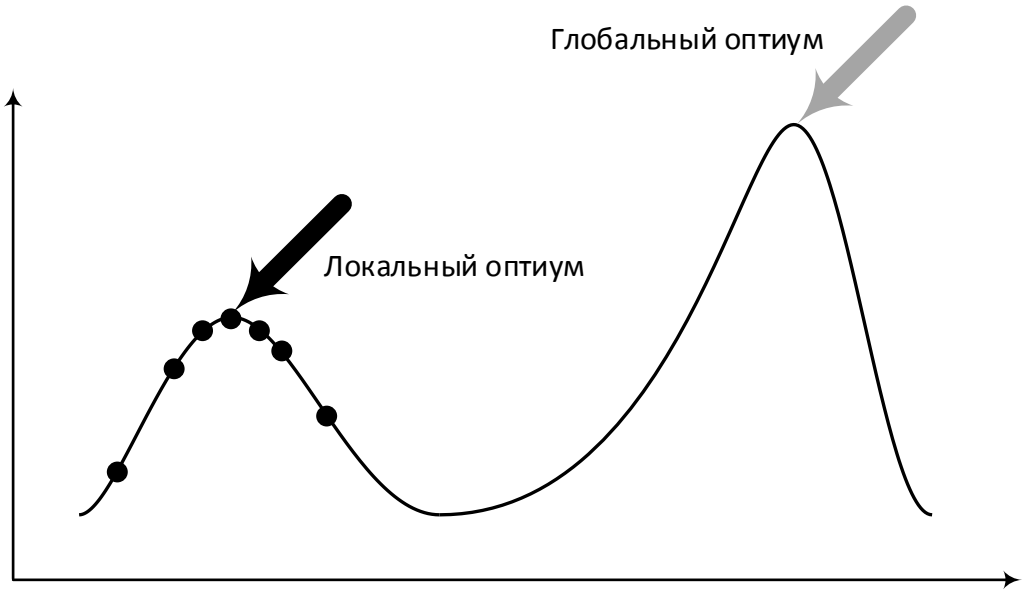

Рисунок 1.3 – пример локальной сходимости при поиске максимума

Также стоит отметить, что генетический алгоритм крайне чувствителен к числу искомых атрибутов. Классическая версия генетического алгоритма, стремящаяся охватить все атрибуты, может столкнуться с ситуацией, когда нахождение значения некоторого атрибута станет крайне маловероятным [6]. Это означает, что генетический алгоритм требует разбиения одной большой задачи на ряд подзадач, работающих с допустимым числом искомых атрибутов.

При практическом применении генетического алгоритма достаточно остро встает проблема потребления памяти, выделенной для хранения индивидов в популяции. Для реализации решения приемлемого качества пользователю алгоритма необходимо найти компромисс между допустимым размером популяции и приемлемым генетическим разнообразием [16]. Это означает, что для работы генетического алгоритма требуются индивиды с различными хромосомами (значениями атрибутов). Когда хромосомы будут

слишком похожи друг на друга, произойдет преждевременное завершение поиска. Если это произойдет слишком рано, то велика вероятность получить некачественное решение. Но для хранения индивидов с различными хромосомами требуется большая популяция с относительно нестрогим отбором, которая расходует больше памяти.

В данной работе предлагается способ конфигурации генетического алгоритма на основе многоуровневого оценивания получаемых индивидов. Это означает, что общая задача поиска будет разбита на ряд подзадач, а общая фитнесс-функция будет представлена в виде последовательности функций, оценивающих отдельные атрибуты получаемых объектов. Предполагается, что в ходе работы такого алгоритма искомый объект будет составляться по частям, что позволит уменьшить расход памяти для популяции и расход процессорного времени на поиск решения. Правильно подобранный набор функций позволит обеспечить наиболее доступную скорость поиска, а также упростит дальнейшее сопровождение реализации алгоритма.

Глава 2 Генетический алгоритм с вариативными компонентами

2.1 Описание генетического алгоритма с вариативными компонентами

Классическая схема генетического алгоритма предполагает поиск готового решения. Это означает, что классический алгоритм стремится отыскать готовое решение, сколь сложным оно бы ни было. Очевидно, что «угадать» все свойства искомого решения получится далеко не сразу, при этом возрастает нагрузка на единственную доступную фитнесс-функцию, что легко приводит либо к ошибкам в оценивании отдельных свойств результата, либо к преждевременной сходимости к заведомо неверному ответу [17].

Для упрощения применения генетического алгоритма автор предлагает заменить генерацию всего объекта сразу на ряд последовательных задач по получению отдельных свойств искомого объекта. Это означает, что над данной популяцией будут последовательно работать *несколько* классических генетических алгоритмов со своими наборами оценивающих, отсеивающих, скрещивающих и производящих мутации функций. Такое разбиение на этапы продемонстрировано на рисунке 2.1, оно дает возможность наглядно контролировать направление поиска, а также позволяет очевидным образом обеспечивать иерархию важности атрибутов искомого объекта.

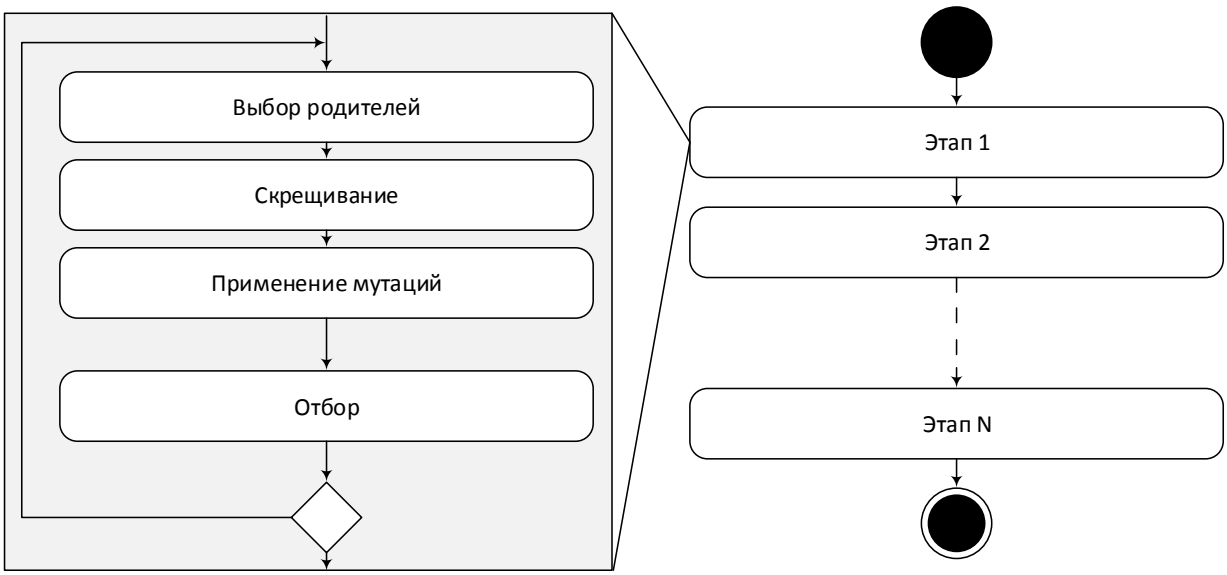

Рисунок 2.1 – этапы работы вариативного генетического алгоритма

Например, при приближенном поиске минимального вершинного покрытия в первую очередь в качестве ответа ожидается корректный набор вершин, действительно образующих вершинное покрытие. После того, как популяция наполнится корректными данными и выполнит необходимое начальное условие, ее можно качественно улучшить, наполнив корректными вершинными покрытиями меньшего размера. Такой подход обеспечивает получение неточного, но корректного ответа в условиях ограниченного времени. По описанному выше принципу разбиения данная задача решается при помощи двух классических генетических алгоритмов: первый образует корректные вершинные покрытия, а второй постепенно уменьшает размер корректных вершинных покрытий, причем работа второго алгоритма может быть остановлена по желанию пользователя (например, при окончании времени на решение).

2.2 Многомерная оценка

Оценка объектов в популяции отражает близость имеющихся вариантов ответа к оптимальному. Как правило, оценка является числом, которое сравнивают либо с другими оценками, либо с заранее определенными константами. Корректный выбор оценки является основополагающим действием, обеспечивающим базовые принципы работы генетического алгоритма. Когда классический генетический алгоритм получит задачу на поиск большого числа атрибутов, фитнесс-функции придется строить одну единственную оценку на основе слишком большого множества атрибутов [18]. В таком случае крайне велика вероятность потерять смысл оценки, когда одну и ту же оценку можно будет получить путем достижения различных наборов значений атрибутов. То есть такая сложная фитнесс-функция не будет обеспечивать наглядную сходимость поиска решения. К тому же усложнение структуры фитнесс-функции легко приведет к ошибкам при разработке, а также серьезному усложнению отладки и тестирования процесса поиска.

Для решения данной проблемы предлагается применение многомерной оценки, являющейся последовательностью оценок отдельных атрибутов. Когда общее условие пройдет через процесс разбиения на подзадачи, будет сформирован ряд отдельных оценок, связанных с определенными подмножествами атрибутов. Такое представление оценки обеспечивает высокую степень гибкости в настройке генетического алгоритма, а также позволяет сохранить смысл оценок. Стоит отметить, что многоуровневая оценка дает основу для принципа модульности, являющегося желательным для современного программного обеспечения.

Таким образом, многомерная оценка является модифицированной версией обыкновенной оценки, приспособленной для решения упрощенных задач поиска. Многомерная оценка является неким каркасом решения, основанного на применении вариативного генетического алгоритма.

2.3 Вариативная фитнесс-функция

Задача вариативной фитнесс-функции – обеспечение иерархии важности различных свойств атрибутов, а также поддержка корректного направления поиска. Непосредственная реализация фитнесс-функции сильно зависит от конкретной структуры искомого объекта.

Основной особенностью вариативной фитнесс-функции является способность в определенный момент изменить направление поиска. Применение вариативной фитнесс-функции предполагает наличие сформированной качественной архитектуры системы оценок, где набор фитнесс-функций должен зависеть от выбранных методов оценивания. Стоит обратить внимание, что фитнесс-функция – крайне часто используемый компонент алгоритма, а потому слишком сложный алгоритм фитнесс-функции приведет к нежелательно низкой скорости работы генетического алгоритма.

Автор предлагает следующие архитектуры наборов фитнесс-функций для данной многомерной оценки:

- базисная (каждой данной фитнесс-функции соответсвует определенная размерность оценки);
- ступенчатая (размерность оценки постепенно возрастает для последующей фитнесс-функции);
- смешанная (оценку формирует последовательность непересекающихся ступенчатых структур).

На рисунке 2.2 показаны варианты соответствия между обрабатываемыми атрибутами (буквы) и этапами (числа):

Рисунок 2.2 – виды оценок вариативного генетического алгоритма

Базисная архитектура является эталонной. Предполагается, что наибольшая производительность вариативного генетического алгоритма наступает тогда, когда алгоритм ищет не всю совокупность атрибутов сразу, а поочередно концентрируется на каждом из них. Базисная архитектура обеспечивает наилучшее быстродействие, но ее реальное применение маловероятно: как правило, между свойствами искомого объекта имеется зависимость, не дающая получить качественное решение путем поиска отдельных атрибутов.

Ступенчатая архитектура наиболее проста в понимании и реализации. Такая архитектура постепенно расширяет зону внимания вариативного генетического алгоритма, постепенно увеличивая размер совокупности

искомых атрибутов. Ступенчатая архитектура имеет наименьшее быстродействие, но она позволяет быстро сконфигурировать направление поиска генетического алгоритма. Рекомендуется применять данную архитектуру только для решений, не приемлющих иных методов организации многомерной оценки (например, для небольших задач с незначительным числом искомых атрибутов).

Смешанная архитектура позволяет достичь компромисса между сложностью конфигурации и получаемой производительностью. Смешанная архитектура позволяет концентрировать работу вариативного генетического алгоритма на различных совокупностях искомых атрибутов. Данная архитектура рассчитана на организацию решения объемных задач с большим числом атрибутов, когда имеется возможность разбить множество всех атрибутов на независимые подмножества.

2.4 Иерархия фитнесс-функций

Вернемся к ранее рассмотренной задаче о поиске вершинного покрытия. Примем за индивид набор вершин, образующих вершинное покрытие. Оценка решения будет производиться по двум параметрам: по полноте охвата ребер графа (то есть вершинное покрытие действительно охватывает все ребра графа) и по размеру вершинного покрытия (чем оно меньше, тем лучше).

Очевидно, что при построении начальной популяции на основе случайных величин возникнут наборы вершин, образующих неполное вершинное покрытие, то есть такие варианты вершинного покрытия не будут охватывать все вершины данного графа и подобные ответы станут заведомо неверными. Это означает, что основной и первой по важности оценкой будет степень полноты охвата ребер графа. Например, в качестве оценки будет выступать количество неохваченных ребер.

После того, как популяция будет наполнена корректными вершинными покрытиями, алгоритм сможет приступить к поиску наименьшего вершинного покрытия, тем самым усовершенствовав первоначальные результаты. Для этого

фитнесс-функция будет оценивать и полноту охвата ребер графа, и размер вершинного покрытия. При этом фитнесс-функция должна давать значительное преимущество таким решениям, которые соответствуют ранее указанному основному критерию. Например, такая фитнесс-функция будет сначала проверять корректность решения и, если оно заведомо некорректно, ставить плохую оценку (число вершин во всем графе), либо ставить значащую оценку, равную мощности данного вершинного покрытия.

Таким образом, степень корректности вершинного покрытия будет играть роль необходимого условия, а размер вершинного покрытия – роль желательного условия.

Глава 3 Разработка требований к реализации вариативного генетического алгоритма

3.1 Разнообразие искомых объектов и атрибутов

При разработке варианта генетического алгоритма следует учитывать возможность применения данного алгоритма для разнообразных задач. Это означает, что один и тот же метод будет использоваться для создания различных объектов, составленных из множества разнообразных атрибутов. А потому в первую очередь в алгоритм требуется заложить возможность абстрагирования от конкретно поставленной поисковой задачи.

Обратим внимание на тот факт, что генетическому алгоритму чаще требуется только оценка приспособленности конкретного объекта, нежели знание о том, как этот объект устроен [6]. Это означает, что часть генетического алгоритма будет работать с всегда известными и неизменными структурами данных, являющимися оценками объектов, а другая часть будет выполнять операции над конкретными объектами, учитывая их внутреннее строение. Например, для операции поиска родителей нужны только оценки индивидов, хотя для получения потомков из выбранных родителей требуется знать строение этих родителей.

Автор предлагает разделить процесс реализации готового алгоритма на два этапа: реализацию общего шаблона генетического алгоритма, применимого к широкому спектру задач и реализацию конкретных алгоритмов, наиболее приближенных к обрабатываемым объектам. Предполагается, что один и тот же шаблон сможет использовать различные конкретизированные алгоритмы, в зависимости от устройства которых будет происходить определенная работа над объектами.

Устройство общего шаблона генетического алгоритма должно поддерживать возможность абстрагирования от выполнения и реализации однотипных задач. Например, каждый генетический алгоритм реализует одинаковую последовательность операций. Но при этом данные операции

отличаются лишь конкретными действиями, основанными на строении искомых объектов [7]. Работа с популяцией также ограничена узким набором операций, включающих поиск родителей, поиск жертв отбора, добавление, чтение и удаление индивидов. При этом желательно избавить конечного пользователя от необходимости управления популяцией и прочими одинаковыми для каждого генетического алгоритма процессами. Наибольшее внимание пользователя должно быть сосредоточено на реализации операций, обрабатывающих искомые объекты. Зачастую именно малопроизводительная реализация данных операторов приводит к неприемлемо медленной работе алгоритма, основанного на частом получении новых данных [8].

Таким образом, задача по реализации генетического алгоритма разбивается на два этапа: поиск шаблонной структуры генетического алгоритма, моделирующей общие принцип работы генетического алгоритма, и привязку базовых операций генетического алгоритма к определенным действиям над конкретными объектами. Данный подход к проектированию позволит реализовать модульную архитектуру вариативного генетического алгоритма. С точки зрения разработки данный подход позволяет приступить к созданию одного поискового алгоритма сразу нескольким группам разработчиков, где каждая группа сможет специализироваться на небольшом наборе операций поискового алгоритма.

3.2 Трудности практического применения генетического алгоритма

Ранее мы рассмотрели трудности применения генетического алгоритма, сопряженные с самой математической моделью генетического алгоритма и его базовым описанием. К сожалению, список препятствий, затрудняющих применение генетического алгоритма как способа решения поисковой задачи, можно дополнить возможной неоптимальной реализацией данного алгоритма и ограничениями аппаратной части [19].

Решение поисковой задачи генетическим алгоритмом базируется на математической модели, позволяющей повысить вероятность нахождения

решения за счет неограниченно большого числа хранимых объектов. Операция отбора в рассматриваемом алгоритме присутствует скорее для обеспечения направления поиска, а не для оптимизации потребления памяти [9]. Очевидно, что данный подход малопригоден для работы на устройствах с ограниченными объемами памяти, обрекая пользователя на отказ от применения классической версии генетического алгоритма как способа решения имеющихся задач.

Также математическая модель генетического алгоритма не учитывает скорость обработки имеющихся в популяции объектов. Медленные реализации основных операций генетического алгоритма приведут к неприемлемо долгому поиску решения [10]. При этом большая часть процессорного времени будет выделена для выполнения именно определенных пользователем операций над объектами. Таким образом, при практическом применении генетического алгоритма следует обратить серьезное внимание на решение вопроса по уменьшению как расхода памяти, так и расхода процессорного времени. Алгоритм, потребляющий недопустимо большое количество ресурсов, навряд ли сможет быть применен на практике, особенно в современных информационных системах, и без того высоко нагруженных.

Для решения данных проблем автор предлагает применение такой схемы вариативного генетического алгоритма, при которой каждому этапу работы поискового алгоритма будет достаточно памяти для работы. К тому же уменьшение числа обрабатываемых атрибутов приведет к значительному ускорению работы алгоритма. К сожалению, на реализацию конкретных алгоритмов, выполняющих операции над конкретными объектами, применение принципа вариативного генетического алгоритма влияет лишь косвенно, оставляя решение вопроса о производительном оперировании объектами для каждой конкретной задачи на рассмотрение пользователем вариативного генетического алгоритма.

3.3 Общие требования к вариативному генетическому алгоритму

Примем во внимание перечисленные выше трудности, сопряженные с применением генетического алгоритма для решения поисковых задач, и определим основные требования к такому алгоритму.

В первую очередь вариативный генетический алгоритм должен поддерживать принцип модульности. Суть вариативного генетического алгоритма базируется на понятиях модульности и масштабируемости. Поступающая общая задача будет разбита пользователем на ряд подзадач. Это означает, что пользователь каркасной части вариативного генетического алгоритма должен иметь возможность добавлять нужное количество этапов поиска необходимого объекта, при этом испытывая минимум затруднений.

В понятие модульности вариативного генетического алгоритма также входит возможность конфигурации этапов поиска. Так как каждый этап является классическим генетическим алгоритмом, то его конфигурация будет заключаться в предоставлении набора реализаций основных операторов генетического алгоритма (оценивания, скрещивания, мутации и т.д.).

Предполагается, что задача по организации популяции будет выполняться вариативным генетическим алгоритмом. Это означает, что вариативный генетический алгоритм будет использовать систему многомерных оценок, организовывая популяцию по указанному пользователем методу.

Стоит обратить внимание на возможность создания вариативного генетического алгоритма, поддающегося распараллеливанию. Данная особенность позволит успешно применять данный алгоритм на современных многоядерных устройствах.

3.4 Описание требований к реализации вариативного генетического алгоритма

Вариативный генетический алгоритм состоит из последовательности классических генетических алгоритмов. Предполагается, что базовая структура классического генетического алгоритма не будет изменяться. При этом

обработка последовательности классических генетических алгоритмов также будет постоянным элементом, не изменяющимся в зависимости от конкретной задачи.

Таким образом, разработка готового вариативного генетического алгоритма разделяется на создание общей структуры вариативного генетического алгоритма, выполняющей набор операций классического генетического алгоритма, абстрагируясь от их реализации, и на создание компонентов классического генетического алгоритма, которые будут применены при обработке конкретных структур данных.

При этом пользователь должен иметь возможность конфигурировать этапы решения поисковой задачи путем внедрения конкретных реализаций операторов классического генетического алгоритма в данный этап. В настройку этапа входит также реализация фитнесс-функции, выставляющей оценку индивидам на данном этапе.

Вариативный генетический алгоритм требуется реализовать таким образом, чтобы была возможность параллельного наполнения популяции новыми индивидами и параллельного удаления плохо приспособленных индивидов. То есть операции в этапе разделяются по принципу записи/удаления, и выполняются таким образом, что в определенный момент времени алгоритм либо только добавляет новые индивиды, либо только удаляет их.

Вариативный генетический алгоритм организует работу с памятью, записывая индивиды в ассоциативный массив с ассоциацией «оценка индивида – индивид». Это позволит находить необходимые индивиды сразу по их оценкам, ведь генетическому алгоритму зачастую чаще требуется оценка индивида, а не его содержимое. При этом если диапазон оценок мал, а данная ассоциация приводит к малому числу видов в популяции (происходит перезапись, а не добавление), недостаточному для работы генетического алгоритма, то допустимо применение ассоциации вида «оценка индивида – список индивидов». Как правило, индивиды с малым диапазоном оценок не

занимают много места в памяти, делая такую ассоциацию применимой на практике.

Вариативный генетический алгоритм предполагается применять с числовыми оценками, как правило, целочисленными. Оператор оценивания индивида реализуется непосредственно для каждого этапа, давая возможность оценивать один и тот же объект по-разному, чтобы обработать его различные атрибуты.

Также следует учесть, что вариативный генетический алгоритм может быть остановлен при переходе от этапа к этапу. Это позволит контролировать ход работы поисковой задачи. При переходе от этапа к этапу полученные ранее индивиды сохраняются, но обретают новые оценки в соответствии с алгоритмом оценивания, установленным для данного этапа.

Глава 4 Процесс реализации вариативного генетического алгоритма

4.1 Каркасная архитектура вариативного генетического алгоритма

Ранее упоминалось требование к реализации вариативного генетического алгоритма, позволяющее провести разделение на абстрактную и конкретизированную части. Абстрактная часть будет организовывать вызов операторов генетического алгоритма в нужной последовательности, а также организовывать популяцию в памяти. Конкретизированная часть будет реализовывать определенные действия над известными структурами данных. То есть абстрактная часть неизменна для любых искомых объектов, а конкретизированная часть будет изменяться в зависимости от структуры искомого объекта. Таким образом, вариативный генетический алгоритм будет разработан с применением каркасного подхода, где в качестве каркаса выступит абстрактная часть алгоритма, а в качестве заменяемых модулей – реализации основных операторов генетического алгоритма.

Обозначим изменяемые модули: ими являются любые действия над индивидами, зависящие от строения индивидов. К строению индивидов привязаны операторы скрещивания, мутации и оценки индивидов, данные операторы реализует пользователь базовой части вариативного генетического алгоритма. Чтобы предоставить пользователю возможность регулировать направление поиска, автор предлагает избавить пользователя от поиска индивидов в популяции, дав возможность работать непосредственно с оценками имеющихся индивидов.

Можно видеть, что при этом операторы выбора конкретных родителей и жертв селекции, работающие на основе оценок индивидов, не знают о строении индивидов, а потому могут быть реализованы в абстрактной части. К тому же в абстрактную часть можно заложить последовательность выполняемых операций, так она останется неизменной для любого вариативного генетического алгоритма.

4.2 Варианты популяций вариативного генетического алгоритма

Упомянутое выше разделение операторов может быть описано разделением на две группы вида «работает с оценкой» и «работает с атрибутами». Операторы, работающие с оценкой индивида, реализуются в базовой части вариативного генетического алгоритма. Экспериментальным путем было установлено, что для организации популяции подходит ассоциативный массив, привязывающий уникальное значение оценки к одному индивиду. При этом если в популяцию добавляется индивид, а его оценка не является уникальной (то есть в популяции уже есть индивид с данной оценкой), то старый индивид заменяется новым. Такая организация памяти позволяет замедлить рост популяции, производя экономный расход памяти. Возможны ситуации, что с такой популяцией не придется использовать операцию селекции (наступает момент, когда диапазон и плотность оценок стабилизировались), что может положительно сказаться на скорости работы алгоритма.

К сожалению, у описанной ранее автоселекции есть серьезный недостаток: при узком диапазоне оценок в популяции будет храниться слишком мало индивидов, в то время как генетическому алгоритму для работы требуется генетическое разнообразие [11]. В таком случае допустима привязка уникальной оценки к списку индивидов (список может быть ограниченной или неограниченной величины). То есть при добавлении индивида с неуникальной оценкой удаления старых данных не произойдет. Автор рекомендует применять данную структуру популяции для малогабаритных (в плане занимаемой памяти) объектов, небольшой диапазон оценок которых обусловлен малым объемом носимых данных, а размер занимаемой памяти остается приемлемым даже при большом количестве таких объектов в популяции.

Таким образом, пользователю будет предоставлен выбор из двух каркасов, реализующих две модели популяции, где в одной происходит ассоциация уникальной оценки с одним объектом, а в другой – ассоциация уникальной оценки со списком объектов.

4.3 Очередь подзадач вариативного генетического алгоритма

Разделение единой поисковой задачи на последовательность подзадач обусловлено желанием сократить расход аппаратных ресурсов и времени за счет небольшого усложнения структуры поискового алгоритма. Предполагается, что разбиение на подзадачи позволит направить процесс поиска для вычисления узкого ряда атрибутов, давая возможность задействовать меньше памяти для хранения популяции и времени для поиска нужных значений атрибутов.

Данная концепция реализуется за счет того, что вариативный генетический алгоритм производит обработку единой популяции последовательностью классических генетических алгоритмов. Каждый классический генетический алгоритм, являющийся этапом решения общей поисковой задачи, обладает собственным набором реализаций операторов скрещивания, мутации и оценивания. Так, на каждом этапе происходит определенная пользователем обработка объектов в популяции по указанным алгоритмам.

Этап завершает свою работу, когда популяция начинает удовлетворять условиям, указанным пользователем (например, достигнута определенная концентрация видов с некоторыми оценками [12]). При переходе от этапа к этапу индивиды в популяции сохраняются, но оценки индивидов при этом устаревают. Это означает, что при переносе популяции должно происходить повторное оценивание индивидов, вычисляющее степень адаптации к новым условиям отбора.

4.4 Конфигурация вариативного генетического алгоритма

На основе указанных решений сформируем алгоритм использования каркаса вариативного генетического алгоритма. Например, у нас имеется объемная поисковая задача, где требуется отыскать значения для множества некоторых переменных. Постараемся разбить данную объемную задачу на ряд подзадач поменьше. В первую очередь следует обратить внимание на

возможность разбиения атрибутов на независимые группы. То есть значения атрибутов одной независимой группы можно найти отдельно от значений атрибутов другой независимой группы. Например, генетический алгоритм должен найти два пути на двух разных локациях. Очевидно, что оценка «пути найдены»/«пути не найдены» приведет к решению задачи поиска двух путей сразу, тогда как достаточно всего лишь найти сначала один путь, а затем другой.

Описанное выше разбиение формирует базисные оценки вариативного генетического алгоритма. Базисные оценки не влияют друг на друга, то есть пожертвовать одной базисной оценкой для улучшения другой базисной оценки не получится, причем каждую из таких оценок можно улучшить одновременно. К сожалению, выделить базисные оценки удается не всегда, и часто возникают ситуации, когда значения одних атрибутов влияют на значения других атрибутов. Например, задача найти такой путь, который был бы и кратчайшим, и с наименьшим числом поворотов. В данном случае все зависит от того, какие приоритеты расставит пользователь, какое качество пути его интересует больше. Например, в первую очередь нужен кратчайший путь, а наличие поворотов желательно сделать как можно меньшим. В таком случае составляется ступенчатая оценка, где сначала происходит заполнение популяции короткими путями с любым количеством поворотов, а затем происходит завершающее улучшение популяции, которую пытаются заполнить кратчайшими путями с наименьшим числом поворотов. То есть сначала улучшаются самые приоритетные качества, а затем к ним добавляются менее важные [20].

На основе полученной схемы оценок создается последовательность этапов. Предполагается, что наиболее приоритетные оценки будут улучшаться до менее приоритетных. Такое разбиение позволяет распределить условия на необходимые и достаточные условия и дает возможность досрочно остановить поисковый алгоритм, например, из-за нехватки времени.

Конфигурация каждого этапа заключается в предоставлении наборов реализаций операторов скрещивания, мутации и оценки, а также в выборе функции, отвечающей за выбор родителей и за выбор удаляемых объектов. Например, можно выбирать из популяции случайных родителей с равной вероятностью для каждого из них, либо отдавать предпочтение тем особям, которые имеют наивысшие оценки.

4.5 Параллельные операции вариативного генетического алгоритма

Для повышения производительности алгоритма на многоядерных аппаратных устройствах разумно использовать распараллеленную версию генетического алгоритма. Автор предлагает выделить операции, работающие с популяцией, и распределить их по двум группам: «чтение-запись» в популяцию и «чтение-удаление» из популяции.

К операциям, добавляющим в популяцию новые индивиды, можно отнести последовательность операций выбора родителей, скрещивания и мутации. Работа с популяцией происходит только в начале указанной последовательности, где происходит считывание и выбор видов, и в конце указанной последовательности, где происходит добавление потомка в популяцию. Каждая операция в последовательности не является неприемлемо медленной, но поочередное выполнение большого числа таких последовательностей в одном потоке занимает ощутимое время, а потому имеет смысл разбить одну большую очередь на ряд параллельных очередей поменьше, которые будут обрабатываться каждым ядром системы.

Операцией, удаляющей из популяции индивиды (предполагается, что наименее приспособленные), является операция селекции. Время работы операции селекции в значительной степени зависит от реализации ассоциативного массива, в котором хранятся индивиды. Очередь селекции также поддается разбиению на ряд очередей поменьше для выполнения данного оператора несколькими ядрами одновременно.

С целью упрощения параллельного доступа к данным, а также требования получить предсказуемую работу модели генетического алгоритма автор предлагает не выполнять операции записи и удаления одновременно. То есть при работе параллельной версии вариативного генетического алгоритма в определенный момент времени будет происходить либо только запись в популяцию, либо только удаление из популяции.

4.6 Выбор индивидов из популяции

Одной из наиболее частых операций, производимых при работе с популяцией, является выбор ключа (оценки индивида), с которым ассоциирована искомая структура данных. Генетический алгоритм работает на основе выбора случайных индивидов из популяции. Выбор индивида из популяции нужен для выполнения операции поиска родителей и для операции удаления малоприспособленных особей, бесполезно занимающих память.

Выбор индивида может осуществляться при помощи панмиксии, когда у каждой особи в популяции равные шансы стать родителем, либо при помощи специальных функций, например, дающих больше шансов более приспособленным особям [13]. Ранее указывалось, что в вариативном генетическом алгоритме выбор основывается на оценочном показателе индивида, а потому список ключей в ассоциативном массиве индивидов будет постоянно изменяться. К сожалению, получение актуального списка ключей для каждой попытки выбрать индивид приводит к неприемлемо долгой работе алгоритма. Для ускорения процесса поиска автор предлагает применять кэшированный список ключей: перед каждым параллельным участком алгоритма происходит обновление кэша ключей, с которыми алгоритм может быстро работать.

Можно видеть, что добавление новых индивидов не вызывает каких-либо затруднений и осуществимо на основе кэшированных данных, так как кэшированных ключей будет либо меньше, либо столько же, сколько действительно есть в популяции. Но стоит обратить внимание на то, что при

удалении индивидов возникнет ситуация, когда кэш будет содержать больше ключей, чем их есть на самом деле. Для такого случая в алгоритме должна присутствовать проверка на наличие ключа в популяции перед удалением его содержимого.

Глава 5 Вариант реализации вариативного генетического алгоритма на Java

5.1 Вариант реализации каркаса вариативного генетического алгоритма

Автор предлагает реализацию вариативного генетического алгоритма в виде многофункционального и удобного API. Для его использования требуются лишь базовые знания того, как работает классический генетический алгоритм. Предполагается, что таких знаний достаточно для исполнения корректных реализаций обработки пользовательских объектов. Параллельное выполнение процессов моделирования, а также работа с популяцией выполняются на уровне каркаса.

Предлагаемый каркас реализует два типа популяций, предоставляющих ассоциации «оценка – индивид» и «оценка – список индивидов». На основе данных популяций предоставляется два варианта соответствующих реализаций абстрактных этапов. Для конфигурации этапа пользователь задает реализации операторов скрещивания, мутации и оценивания, а также предоставляет функции поиска родителей и жертв отбора.

Основной класс вариативного генетического алгоритма получает от пользователя реализацию генератора стартовой популяции, а также количество потоков, параллельно выполняющих операции генерации, размножения и отбора. Порядок выполнения этапов задается пользователем, при этом имеется возможность получить результат работы этапа. Основной класс сохраняет полученные между этапами объекты, передавая их следующему этапу.

С целью проверки того, насколько вариативный генетический алгоритм поддается реализации при помощи современных технологий, было решено активно использовать ООП для реализации каркаса. Для этого реализации операторов генетического алгоритма было решено предоставлять через интерфейсы (Java interface), а типизацию данных организовать через шаблоны (Java generic). Предполагается, что такая реализация каркаса вариативного генетического алгоритма позволит упростить процесс ознакомления с API, а

также будет подталкивать пользователя к верным шагам, ведущим к корректному использованию алгоритма.

Таким образом, предлагаемая реализация каркаса получает от пользователя только готовые методы работы с конкретными объектами, при этом избавляя его от решения вопросов по организации памяти и по определению последовательности выполняемых операций. Диаграмма классов каркаса изображена в приложении А.

Классы *Abstract Single Association Step* и *Abstract Multiple Association Step* предоставляют операции над индивидами и популяцией. *Abstract Single Association Step* реализует ассоциацию одного индивида с уникальной оценкой, а *Abstract Multiple Association Step* позволяет хранить несколько индивидов под одной уникальной оценкой (работает медленнее, чем *Abstract Single Association Step*, и используется в тех случаях, когда *Abstract Multiple Association Step* неспособен справиться с решением задачи).

Приведем список интерфейсов, позволяющих производить конфигурацию этапов.

1. Интерфейс *Abstract Evaluator* указывает генетическому алгоритму способ оценки добавляемых в популяцию индивидов.
2. Интерфейс *Abstract Breeder* указывает алгоритму, как именно производить скрещивание индивидов.
3. Интерфейс *Abstract Mutator* указывает алгоритму набор действий, необходимых для применения определенных пользователем мутаций к полученному потомку.
4. Интерфейс *Abstract Parent Selector* указывает генетическому алгоритму способ выбора родителей на основе их оценок приспособленности.
5. Интерфейс *Abstract Victim Selector* указывает генетическому алгоритму способ выбора удаляемых из популяции индивидов на основе их оценок приспособленности.

6. Интерфейсы *Abstract Single Condition* и *Abstract Multiple Condition* обозначают генетическому алгоритму условие выполнения этапа на основе множества оценок приспособленности и количества индивидов с данными оценками.

Класс *Abstract Search Process* руководит процессом выполнения операций генетического алгоритма и последовательности ступеней решения поисковой задачи. Данный класс получает экземпляр *Abstract Generator*, а также размер стартовой популяции, количество размножаемых индивидов, количество удаляемых индивидов и количество потоков, по которым будут распределено параллельное выполнение некоторых операций (рекомендуется приравнивать к числу используемых ядер процессора).

Интерфейс *Abstract Generator* является основой для генератора стартовой популяции. Его реализация указывает генетическому алгоритму способ создания объектов начальной популяции.

Использование вариативного генетического алгоритма сводится к созданию реализаций указанных выше интерфейсов, с помощью которых конфигурируются этапы алгоритма. Затем пользователь предоставляет конфигурацию основных параметров *Abstract Search Process,* после чего формирует очередь выполнения этапов. При этом пользователь имеет возможность получать промежуточные результаты работы, предоставляемые после завершения каждого этапа.

Таким образом, произведена попытка предоставить пользователю понятный и легко осваиваемый шаблон, по которому можно быстро создать реализацию конкретного поискового алгоритма. Можно видеть, что вариативный генетический алгоритм успешно поддается реализации на основе современных технологий разработки.

5.2 Вариант реализации вариативного генетического алгоритма для поиска точек минимума функции на заданном отрезке

Рассмотри процесс решения классической математической задачи о поиске точек минимума функции на отрезке при помощи генетического алгоритма. Очевидно, что данная задача имеет быстрые способы решения, связанные, например, с анализом критических точек, однако такие методы предполагают дальнейшую работу над условием конкретной задачи. То есть берется заданная условием функция, у которой затем производят поиск частных производных, и анализ уже осуществляется по отношению к производным данной функции. Особенность вариативного генетического алгоритма состоит в том, что он позволяет решить задачу без изменения условий или превращения одной задачи в другую, совершенно непохожую на исходную.

Определим условие задачи: найти значения аргумента функции $\sin(x)$ на отрезке $[-5; 5]$, в которых функция принимает наименьшие значения. Можно видеть, что данная задача имеет вполне очевидное решение: достаточно решить уравнение вида $\sin(x) = -1$ и получить множество значений x, описанное уравнением $x = \frac{3}{2}\pi + 2\pi k$, где $k \in \mathbb{Z}$. Подходящих значений на заданном отрезке обнаружится два: $x = \{-\frac{1}{2}\pi; \frac{3}{2}\pi\}$ (в десятичной форме минус 1.57079 и плюс 4.71238 соответственно). Так как нам заранее известно верное решение задачи, то используем ее для тестирования вариативного генетического алгоритма и сравним аналитически полученные ответы со сгенерированными.

Рассмотрим процесс конфигурации вариативного генетического алгоритма. В качестве типа искомых объектов выберем число с плавающей точкой (Java float). Оценкой приспособленности данного числа будет соответствующее значение функции, округленное с необходимой точностью. Известно, что $\sin(x)$ – периодическая функция, а потому необходимо применять популяцию с возможностью ассоциации нескольких индивидов с одной уникальной оценкой.

С целью проверки способности алгоритма находить верный ответ с помощью свойства направленности поиска генератор первоначальной популяции исполнен как генератор случайных чисел с плавающей точкой, равномерно распределенных на отрезке $[-5; 5]$. Такой подход позволит наполнить большую часть популяции непригодными в качестве ответа индивидами.

Скрещивание индивидов будет происходить на основе двух родителей. Из выбранных родителей вычисляется их среднее арифметическое. Данный способ скрещивания позволит обеспечить необходимую для направленного поиска сходимость, когда при большом количестве выполнения подобных действий произойдет постепенное приближение к искомому результату.

Случайная мутация потомка выполнена в виде внесения небольшого случайного изменения в значение потомка. Данный подход позволит сделать популяцию богаче, при этом не сильно влияя на значения получаемых результатов. Также возможна ситуация, когда случайная мутация приведет к получению более точного ответа, повысив качество популяции.

Направленность поиска в данной реализации будет осуществляться через реализацию операторов выбора родителей и жертв отбора. Для этого применим упрощенный вариант метода рулетки, где половина индивидов в популяции, чьи ключи относятся к более приспособленным оценкам, получает равные шансы стать родителем, а индивиды в половине менее приспособленных оценок не получают возможности стать родителями вообще. При этом малоприспособленные индивиды имеют равные шансы стать жертвами отбора и быть удаленными из популяции, в то время как более приспособленные индивиды освобождаются от отбора. Такое поведение операторов отбора будет постепенно смещать популяцию к точкам минимума.

Заключение о том, является ли данный этап выполненным, будет базироваться на плотности сгущения точек. Это означает, что поиск будет завершен, когда на определенный ключ начнет поступать достаточно много индивидов. В качестве коэффициента плотности результатов было решено

использовать показатель, равный 20, как приводящий к верному результату в приемлемые сроки.

Рассмотрим усредненные результаты 100 запусков приложения на Java, реализующего поиск минимумов $\sin(x)$. Алгоритм был настроен производить генерацию стартовой популяции размером 200 видов (в среднем изначальная базовая точность решения $\frac{200}{5-(-5)} = 0.5$, такое значение наглядно продемонстрирует способность алгоритма улучшать начальные показатели), производить разведение 200 потомков между обновлениями кэша ключей, производить удаление 150 потомков между обновлениями кэша ключей. Точность решения 10^{-5} по значению функции. В ходе работы алгоритм смог отыскать оба искомых минимума с точностью 10^{-3}. В среднем работа осуществлялась за 12 итераций поискового цикла и требовала 100мс процессорного времени на аппаратной платформе с двухъядерным процессором с частотой каждого ядра 3ГГц.

Экспериментально было проверено влияние частоты обновления кэша на производительность алгоритма. Так, при обновлении кэша через каждые 2 разведенных и 2 удаленных индивида полученный ранее результат достигался в среднем за 900 итераций поискового цикла, а алгоритм работал 1000мс. При уменьшении частоты обновления кэша также наблюдается падение производительности: при разведении 1000 индивидов и удалении 750 алгоритм осуществляет в среднем 5 итераций поискового цикла и работает в среднем 150мс.

Таким образом, вариативный генетический алгоритм смог осуществить решение задачи по поиску минимумов функции $\sin(x)$ на отрезке $[-5; 5]$. При этом исходная функция не изменялась. Также в ходе применения были получены такие значения настроек алгоритма, которые позволяют находить необходимое решение за приемлемое время.

5.3 Вариант реализации вариативного генетического алгоритма для поиска минимального вершинного покрытия графа

Задача о поиске минимального вершинного покрытия уже рассматривалась в данной работе как один из вариантов поисковой задачи. Необходимо заметить, что точное решение данной задачи генетическим алгоритмом не гарантируется. С другой стороны, имеется возможность получать качественные результаты с приемлемыми отличиями от наилучшего результата, и все зависит от того, насколько корректно настроен вариативный генетический алгоритм.

Рассмотрим процесс конфигурации элементов алгоритма для решения поставленной задачи. Сначала определим условие: необходимо отыскать приближенную версию минимального вершинного покрытия данного графа. На вход алгоритму будет подаваться неориентированный связный граф, а на выходе будем ожидать множество вершин, предположительно образующих минимальное вершинное покрытие. В качестве основы для поиска воспользуемся вполне очевидными свойствами минимального вершинного покрытия: оно содержит небольшое количество вершин (не более половины от количества вершин в графе), а также охватывает все ребра графа (все ребра в графе будут инцидентными к корректному вершинному покрытию).

В качестве искомой структуры оставим само вершинное покрытие графа. Представим искомые объекты в виде кортежей из вершин, а все операции генетического алгоритма будем производить над ними. Данный подход позволит проверить то, насколько близко к искомой структуре можно отнести работу вариативного генетического алгоритма.

Ранее рассматривался вариант алгоритма, формирующий искомое решение задачи за два этапа, когда сначала алгоритм пытается получить корректные вершинные покрытия, а затем из корректных вершинных покрытий образуется наименьшее возможное вершинное покрытие. В данном случае формируется общая популяция, которую по очереди используют различные этапы.

Представим, что нам неизвестны методы построения минимальных вершинных покрытий, а потому воспользуемся свойством генетического алгоритма улучшать имеющиеся индивиды. В данной ситуации генератор стартовой популяции должен генерировать варианты вершинных покрытий любого качества. Выполним реализацию такого генератора таким образом, чтобы он составлял случайные последовательности вершин. Пусть генератор будет добавлять каждую вершину в графе с вероятностью 0.5. Так мы получим стартовую популяцию, соответствующую приближенному решению задачи о построении вершинного покрытия, когда размер вершинного покрытия не превосходит половины числа вершин в графе.

Реализуем скрещивание, основанное на двух родителях. Представим последовательности вершин родителей в виде хромосом, которые будем скрещивать по методу одноточечного кроссинговера. Рассмотрим процесс скрещивания подробнее. Точкой кроссинговера называется место в хромосомах, в котором они будут разделены и склеены между собой. По отношению к будущим вершинным покрытиям произойдет выбор номера вершины, относительного которого будет происходить кроссинговер: вершины, чей номер меньше выбранной точки кроссинговера, будут выбираться из одного родителя, а чей номер больше – из другого. Такой метод создания потомков обеспечит передачу необходимой информации от родителей, обеспечивая нужное направление поиска.

В качестве оператора мутации возьмем реализацию, основанную на внесении случайных изменений в потомка. Оператор мутации будет работать по следующему алгоритму: сначала выбирается случайный номер вершины, находящейся в графе, затем происходит проверка на наличие данной вершины в последовательности, образующей вершинное покрытие. Если такая вершина в покрытии имеется, то она удаляется из потомка, иначе она добавляется в потомка. Данный способ приведет к поступлению новых вариантов генов, образованных в ходе работы алгоритма.

Поддержим направление поиска через реализацию операторов выбора родителей и жертв отбора. Для этого применим ранее рассмотренный упрощенный вариант метода рулетки, где половина индивидов в популяции, чьи ключи относятся к более приспособленным оценкам, получает равные шансы стать родителем, а индивиды в половине менее приспособленных оценок не получают возможности стать родителями вообще. При этом малоприспособленные индивиды имеют равные шансы стать жертвами отбора и быть удаленными из популяции в то время, как более приспособленные индивиды освобождаются от отбора.

Рассмотрим работу операторов оценивания: их потребуется два, так как алгоритм будет основан на двух этапах обработки популяции. Так как первый этап призван сгенерировать корректные вершинные покрытия, то в качестве оценки применим количество неохваченных данным вариантом вершинного покрытия ребер в графе. Второй этап будет генерировать уменьшенные корректные вершинные покрытия, а потому введем комплексную оценку: алгоритм сначала будет оценивать корректность варианта вершинного покрытия, а затем его размер. Если вариант вершинного покрытия некорректный, то он получает наихудшую оценку, равную числу вершин в графе, иначе он получает оценку, равную имеющемуся количеству вершин.

Заключение о том, является ли каждый этап выполненным, будет базироваться на плотности получаемых результатов. В случае с первым этапом необходимо определенное наличие корректных вершинных покрытий. В данном случае проходным коэффициентом плотности будут являться 100 вариантов корректных вершинных покрытий. Недостаточный проходной коэффициент приведет к бедности генов для обработки следующим этапом, а потому иногда необходимо экспериментально определять размер подобных коэффициентов. Для второго этапа необходимо определять точку сгущения в неизвестном нам ключе, то есть изначально мы не знаем, какое именно вершинное покрытие у графа наименьшее. Экспериментально было

установлено, что коэффициент сгущения у первого (наилучшего) ключа, равный 10, приводил к получению корректных результатов.

Рассмотрим усредненные результаты 100 запусков приложения на Java, реализующего поиск минимального вершинного покрытия данного графа. Алгоритм был настроен производить генерацию стартовой популяции размером 100 индивидов, производить разведение 100 потомков между обновлениями кэша ключей и производить удаление 80 потомков между обновлениями кэша ключей. Для проверки алгоритма были использованы случайно сгенерированные графы с количеством вершин от 10 до 50. При обработке графа с 10 вершинами в среднем требовалось 6 итераций первого поискового и 2 итерации второго поискового циклов; алгоритм стабильно находил точное решение задачи о поиске минимального вершинного покрытия данного графа. При обработке графа с 50 вершинами в среднем требовалось 30 итераций первого и 7 итераций второго поискового циклов. Алгоритм не всегда приходил к точному решению, выводя в качестве ответа вершинное покрытие, большее минимального на 1-3 вершины с вероятностью 0.7. Стоит отметить, что была обнаружена закономерность, заключающаяся в том, что достижение точного решения задачи требовало в среднем меньше итераций второго поискового цикла.

Таким образом, вариативный генетический алгоритм был испытан в качестве средства для решения NP-полной задачи. Алгоритм действительно давал неточные результаты, как было указано ранее, а потому его применение для получения точного решения ограничено (ведь алгоритм ошибался не всегда, иногда стабильно выводя правильное решение). Возможно, это связано с постоянством настроечных коэффициентов, не зависящих от размера графа, что открывает перспективное направление по автоматической настройке вариативного генетического алгоритма.

Заключение

В ходе работы была рассмотрена возможность применения генетического алгоритма как средства решения поисковых задач. Были проанализированы достоинства и недостатки классического генетического алгоритма, связанные со свойствами его математической модели, а также связанные с его реальным применением. С целью повышения эффективности генетического алгоритма, а также расширения круга решаемых задач была разработана модификация генетического алгоритма, позволяющая производить быструю и удобную настройку компонентов алгоритма, а также дающая возможность обойти ресурсные ограничения аппаратной платформы. В работе продемонстрирован вариант каркаса вариативного генетического алгоритма, на основе которого были разработаны алгоритмы, осуществляющие решение двух поисковых задач: задача о поиске минимума некоторой функции на отрезке и задача о приближенном поиске минимального вершинного покрытия. Разработанные алгоритмы успешно находят приближенные решения поставленных задач, демонстрируя определенную применимость для решения поисковых задач.

Список используемой литературы

1. Гладков, Л.А. Биоинспирированные методы в оптимизации / Л.А. Гладков, В.В Курейчик, В.М. Курейчик и др. — М.: Физматлит, 2009. — С. 384.
2. Гладков, Л.А. Генетические алгоритмы / Л.А. Гладков, В.В. Курейчик, ред. В.М. Курейчик - 2-е изд., испр. и доп. - М.: ФИЗМАТЛИТ, 2006. — 320 с.
3. Данилов, В.Р. Метод генетического программирования для генерации автоматов, представленных деревьями решений // Научно-программное обеспечение в образовании и научных исследованиях. 2008. - С. 171-174.
4. Данилов, В.Р. Метод представления автоматов деревьями решений для использования в генетическом программировании // Научно-технический вестник, №53, 2008. - С. 103-108.
5. Данилов, В.Р. Представление функции переходов линейными бинарными графами при генерации управляющих автоматов с помощью генетического программирования // Доклады научно-практической конференции студентов, аспирантов, молодых ученых и специалистов "Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте", 2007. - С. 109-115.
6. Егоров, К.В. Совместное применение генетического программирования и верификации моделей для построения автоматов управления системами со сложным поведением // Научно-технический вестник, №5, 2010. - С. 81-89.
7. Куцый, Н.Н. Применение генетического алгоритма для оптимизации автоматических систем с ПИД-регулятором // Вестник Иркутского государственного технического университета, №6, 2012. – С. 6-10.
8. Мандриков, Е.А. Генерация числовых последовательностей, описывающих движение человекоподобного робота, при помощи генетических алгоритмов // Научные доклады научно-практической конференции студентов, аспирантов, молодых ученых и специалистов "Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте", 2009. – Т. 2. – С. 181-188.
9. Мандриков, Е.А. Построение автоматов с помощью генетических алгоритмов для решения задачи о "флибах" // Сборник докладов X Международной конференции по мягким вычислениям и измерениям, 2007. – С. 293-296.
10. Мандриков, Е.А. Применение распределённых вычислений для автоматической генерации конечных автоматов с использованием генетических алгоритмов // Сборник докладов XI Международной конференции по мягким вычислениям и измерениям, 2008. – С. 255-260.

11. Мандриков, Е.А. Разработка инструментального средства для автоматической генерации конечных автоматов с ипользованием генетических алгоритмов // Научно-технический вестник, №53, 2008. – С. 100-102.
12. Поликарпова, Н.И. Применение генетического программирования для генерации автоматов с большим числом входных // Научно-технический вестник, №53, 2008. - Вып. 53. – С. 24-42.
13. Поликарпова, Н.И. Разработка библиотеки для генерации управляющих автоматов методом генетического программирования // Сборник докладов X Международной конференции по мягким вычислениям и измерениям, 2007. – С. 84-87.
14. Тяхтин, А.С. Виртуальная лаборатория для обучения методам искусственного интеллекта для генерации управляющих конечных автоматов // Сборник докладов IV Международной научно-практической конференции "Современные информационные технологии и IT-образование", 2009. – С. 222-229.
15. Царев, Ф.Н. Совместное применение генетического программирования, конечных автоматов и искусственных нейронных сетей для построения системы управления беспилотным летательным аппаратом // Научно-технический вестник, №53, 2008. – С. 42-60.
16. Abdullah K., David W.C., Alice E.S., Multi-objective optimization using genetic algorithms: A tutorial // Reliability Engineering and System Safety. 2006. Vol. 91. P. 992-1007.
17. Brunauer R., Locker A., Mayer H.A., Mitterlechner G., Payer H. Evolution of iterated prisoner's dilemma strategies with different history lengths in static and cultural environments. 2007.
18. Civicioglu P., "Transforming Geocentric Cartesian Coordinates to Geodetic Coordinates by Using Differential Search Algorithm" // Computers &Geosciences. 2012. Vol. 46. P. 229-247.
19. Hamed S., The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm // International Journal of Bio-Inspired Computation (IJBIC). 2009. Vol. 1.
20. Steven S., The Algorithm Design Manual (2nd ed.). // Springer Science+Business Media. 2010.
21. Tomoiaga B., Chindris M., Sumper A., Sudria-Andreu A., Villafafila-Robles R.P. Optimal Reconfiguration of Power Distribution Systems Using a Genetic Algorithm Based on NSGA-II. // Energies. 2013. Vol. 6(3). P. 1439-1455.
22. Ziarati Z., A multilevel evolutionary algorithm for optimizing numerical functions // IJIEC. 2010. Vol. 2. P. 419-430.

Приложение А

Диаграмма классов каркаса вариативного генетического алгоритма