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

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

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

01.04.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему «Исследование вариантов улучшения конфигурации генетического алгоритма»

Студент Р.И. Семенов _____
(И.О. Фамилия) (личная подпись)

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

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

Допустить к защите

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

Тольятти 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Существует множество задач, решение которых связано с перебором всех возможных вариантов решения. При решении такой задачи каждый сгенерированный вариант решения оценивается, а после завершения перебора из полученного множества решений выбирается самое пригодное в качестве ответа. Особенностью таких задач является то, что имеющихся знаний для ее решения хватает только на генерацию очередного варианта решения и его оценку. Под оценкой в данном случае понимается некоторая величина, позволяющая сравнить два варианта между собой, чтобы затем объявить один из них лучшим (или оба равносильными). При этом из оценки не может следовать вариант решения, равно как ответ на задачу не может следовать из условия рассматриваемой задачи.

Одной из математических моделей для решения таких задач является генетический алгоритм. Данный алгоритм относится к алгоритмам направленного поиска. Как правило, в его реализации участвуют случайные величины, поэтому его можно также называть стохастическим алгоритмом направленного поиска. Генетический алгоритм моделирует естественный процесс эволюции. Данный алгоритм является эффективным способом оптимизации решений с большим числом параметров. Наиболее частым промышленным применением генетического алгоритма является составление расписаний, так как данный алгоритм успешно справляется с задачами планирования и составления расписаний.

Помимо составления расписаний, генетический алгоритм может быть использован для решения различных комбинаторных задач, к которым относятся задачи компоновки, задачи на графах и прочие. В связи с ростом производительности вычислительных устройств и последующим увеличением популярности алгоритмов искусственного интеллекта вообще, генетический алгоритм теперь применяется и для ускорения настройки нейронных сетей.

Исследования, связанные с повышением эффективности генетического алгоритма, являются крайне актуальными по двум причинам. Во-первых, генетический алгоритм можно легко реализовать для решения той или иной оптимизационной задачи. Например, для обнаружения экстремума некоторой функции, алгоритму понадобятся только сама функция, область поиска и тип экстремума (минимум или максимум). Во-вторых, на текущем этапе развития теории данный вариант генетического алгоритма способен эффективно решать только конкретную задачу, под которую он приспособлен. То есть не существует единого варианта генетического алгоритма, способного одинаково эффективно решать любую задачу поиска.

Научная проблема заключается в том, что получение качественного генетического алгоритма является долгим и трудоемким процессом, обращающим обыкновенную разработку в научную работу с применением эвристик. Разработки, позволяющие упростить процесс создания качественного вариант генетического алгоритма, могут быть полезны во всех областях, связанных с применением данного алгоритма. Расширение знаний о работе генетического алгоритма позволит проще и быстрее решать поисковые задачи с большим числом параметров.

Объект исследования – генетический алгоритм и ход его работы.

Предмет исследования – высокая сложность разработки генетических алгоритмов.

Цель исследования – выявить закономерности между различными настройками генетического алгоритма и качеством его работы, найти способы модификации генетического алгоритма, приводящие к повышению качества его работы, найти способы оптимизации процесса разработки генетического алгоритма.

Гипотеза – подробный анализ хода работы различных генетических алгоритмов на различных задачах позволит выявить полезные закономерности, на которые можно будет опереться при разработке генетического алгоритма для некоторой задачи.

Задачи исследования:

1. Изучить классический генетический алгоритм и его операторы, составить необходимые математические модели.

2. Выявить трудности практического применения классического генетического алгоритма и определить критерии, оценивающие качество работы генетического алгоритма.

3. Реализовать такой вариант генетического алгоритма, который бы позволял легко и быстро тестировать различные конфигурации генетического алгоритма.

4. Реализовать инструмент для испытания и исследования конфигураций генетического алгоритма, позволяющий хранить ход работы алгоритма при данной конфигурации, предоставлять данные, полученные в ходе испытания конфигураций генетического алгоритма, в удобном для последующего анализа виде.

5. Разработать и испытать различные варианты генетических операторов путем составления из них конфигураций генетического алгоритма, найти закономерности между настройками данных операторов и качеством получаемого результата.

6. Провести анализ собранных данных, выявить наиболее успешные варианты конфигурации генетического алгоритма, найти закономерности между конфигурациями генетического алгоритма и качеством получаемого результата, разработать рекомендации по созданию генетического алгоритма с качественной конфигурацией.

7. Оформить выводы и результаты исследования.

Текущий процесс разработки генетических алгоритмов строится на верном и подтвержденном практикой предположении, что алгоритм следует приспосабливать к конкретной задачи. Независимая разработка нового генетического алгоритма при помощи перебора различных конфигураций была единственным доступным способом получения эффективных алгоритмов на

заре развития генетических алгоритмов. Сейчас, когда уже разработано множество генетических алгоритмов для решения широкого спектра задач и есть возможность использования ранее полученных достижений, поэтому необходимо развивать знания о процессе получения производительных конфигураций. Источники [3-6,10-17] акцентируют внимание на готовых конфигурациях, игнорируя процесс их получения. Из-за такого взгляда на генетические алгоритмы возникает недостаток знаний о способах получения производительных конфигураций.

В данной работе предлагается подход к решению возникшей проблемы, основанный на проведении абстрактных (от реально возникающих на практике) испытаний генетического алгоритма на тестовых задачах для методов оптимизации. Полученные эмпирические выборки, описывающие ход работы алгоритма, позволят выявить полезные опорные явления, на которые может ориентироваться разработчик генетических алгоритмов. В основу данного исследования положена визуализация полученных данных в виде эволюционных траекторий, подробно описывающих состояние алгоритма в различные моменты времени.

Научная новизна данной работы заключается в уточнении данных о разработке новых генетических алгоритмов. Предлагаются правила выбора генетических операторов, позволяющие сократить время подбора эффективной конфигурации. Предлагается способ прогнозирования работы генетического алгоритма, также позволяющий существенно сократить время подбора качественной конфигурации генетического алгоритма.

Установлено, что размер популяции следует подбирать в зависимости от сложности задачи. Разработано правило выбора размера популяции в зависимости от сложности решаемой задачи. Установлено, что оператор выбора родителей следует подбирать в зависимости от сложности задачи. Разработано правило выбора оператора выбора родителей в зависимости от сложности решаемой задачи, разработан вариант оператора выбора родителей. Установлено, что мощность мутации следует подбирать в зависимости от

сложности задачи. Разработано правило выбора мощности мутации в зависимости от сложности задачи. Установлена обратная зависимость между скоростью получения решения и его точностью, для борьбы с данной особенностью разработан адаптивный вариант оператора мутации. Разработана функция, чья форма схожа с эволюционной траекторией, разработан способ ее получения по первоначальному участку эволюционной траектории.

Объем и структура диссертации: диссертационное исследование состоит из введения, 5 глав, заключения, библиографии (30 наименований). Работа изложена на 94 страницах, содержит 26 рисунков.

Глава 1 Введение в генетический алгоритм

1.1 Назначение генетического алгоритма

Развитие вычислительной техники привело к значительному росту ее производительности. Это привело к повсеместному распространению моделирующих алгоритмов. Одним из таких алгоритмов, имеющих моделирующую природу, является генетический алгоритм. Данный алгоритм моделирует процесс естественной эволюции с точки зрения генетики. Генетический алгоритм относится к алгоритмам искусственного интеллекта: он способен оценить имеющееся решение и внести в него необходимые поправки в надежде на улучшение [1]. Данное свойство позволяет генетическому алгоритму решать оптимизационные задачи, путем оценки и сравнения решения, чтобы попытаться провести улучшение.

Основной причиной применения генетического алгоритма является отсутствие в задаче очевидной зависимости оптимального решения от условия задачи. В качестве примера рассмотрим одну из наиболее известных задач для генетического алгоритма – генерацию расписаний. Например, стоит задача составить учебное расписание для университета, где в качестве ограничений имеются множества из обязательных и желательных ограничений. Вариант обязательного ограничения: проводить пару только в аудитории, в которой поместятся все студенты. Вариант желательного ограничения: отсутствие пустых пар между учебными парами. В данном случае мы имеем возможность пожертвовать выполнением необязательных ограничений в пользу выполнения обязательных.

Очевидно, что среди множества всех вариантов расписаний существует такое, которое наилучшим образом выполняет указанные ограничения. Однако в общем случае его можно найти только полным перебором всех вариантов расписаний, их оценкой, сравнением и объявлением лучшего варианта решения. То есть не существует общего правила, позволяющего получить

оптимальный вариант расписания без предварительного сравнения этого расписания с другими вариантами расписаний.

Однако, способностей создавать варианты расписаний и сравнивать их достаточно для реализации генетического алгоритма. Это означает, что мы можем создавать варианты расписаний до тех пор, пока один из этих вариантов не будет принят нами как подходящий. При этом необходимо найти подходящее решение за приемлемое время, без полного перебора. Для этого следует использовать алгоритм направленного поиска, способный оценивать имеющиеся решения и проводить перебор решений не на всем множестве, а на его небольших подмножествах.

1.2 Понятие генетического алгоритма

Генетический алгоритм является алгоритмом направленного поиска. В отличие от алгоритма полного перебора, рассматривающего все возможные варианты решения, алгоритм направленного поиска стремится отыскать окрестность оптимального решения. Предполагается, что в случае направленного поиска решение будет найдено заметно быстрее, чем при полном переборе, так как проверке подвергнется значительно меньшее множество вариантов решения. На практике решение NP-полных задач при помощи алгоритмов направленного поиска оказывается крайне непростой задачей из-за стремления алгоритма направленного поиска к локально оптимальному решению. Это связано с тем, что алгоритм не имеет информации о локальности или глобальности обнаруженного оптимума, из-за чего алгоритмы направленного поиска не гарантируют нахождение глобально оптимального решения [2].

Работа генетического алгоритма заключается в последовательном исполнении операторов генетического алгоритма, в которые входят:

- оператор выбора родителей;
- оператор скрещивания;
- оператор мутации;

- оператор селекции.

Перечисленные операторы производят манипуляции с данными таким образом, чтобы имеющиеся решения постепенно улучшались. Как только алгоритм получит решение необходимого качества, он остановится и вернет результат работы пользователю.

1.3 Представление данных в генетическом алгоритме

Каждый отдельный вариант решения, сохраненный в памяти генетического алгоритма, является индивидом. Множество решений, хранящихся в памяти генетического алгоритма, образует популяцию. Так генетический алгоритм оперирует индивидами в популяции. Каждый индивид имеет степень приспособленности. Данный показатель определяет пригодность индивида в качестве решения задачи. Генетический алгоритм не накладывает ограничений на реализацию степени пригодности. Как правило, она выражается целым или действительным числом, однако в иных случаях допустимо применение более сложных вариантов.

Математическая модель генетического алгоритма не накладывает ограничений на размер популяции, однако популяция неограниченного размера невыполнима на реальном вычислительном устройстве. Существует множество вариантов сокращения численности индивидов до приемлемых значений. Например, можно сохранять постоянную численность при помощи оператора селекции, удаляющего из популяции столько же индивидов, сколько их было добавлено в данном поколении [3]. Другим простым способом является ассоциация ограниченного числа индивидов с некоторой степенью приспособленности, однако такой подход рассчитан на узкий спектр значений степени приспособленности, встречающийся в комбинаторных задачах с дискретным множеством решений.

Последовательное изменение состояния популяции и содержащихся в ней индивидов формирует последовательность поколений. Предполагается, что в ходе работы генетического алгоритма будет происходить улучшение

популяции, где индивиды в каждом новом поколении окажутся приспособленными лучше по сравнению с индивидами из прошлых поколений. Чем меньше поколений требуется генетическому алгоритму для получения решения необходимого качества, тем лучше и быстрее он работает.

Индивид состоит из хромосом, которые содержат гены. Данная организация позволяет легко представить вариант решения некоторой задачи в виде, приемлемом для генетического алгоритма. Если операторы, связанные с выбором индивидов из популяции, оперируют целыми индивидами и не изменяют их содержимого, то операторы скрещивания и мутации должны редактировать содержимое генов.

Содержимое генов и их набор в хромосоме сильно зависят от задачи, тем не менее, существуют три основных типа генов: действительный, целочисленный и логический. Действительный ген реализован одним действительным числом, целочисленный – целым числом, а логический – логической переменной. Генетический алгоритм допускает применение собственных генов, однако необходимо предоставить реализации операторов скрещивания и мутации, способные обрабатывать данный вид генов.

1.4 Генерация начальной популяции

Генетический алгоритм оперирует последовательностью поколений, образуя новое улучшенное поколение индивидов на основе индивидов из прошлого поколения. Однако перед этим требуется создать начальную популяцию, которая опишет самое первое поколение.

Полезной особенностью генератора начальной популяции является допустимость генерации индивидов любого качества. Предполагается, что улучшением индивидов из начальной популяции займутся другие операторы генетического алгоритма [4]. Основным назначением генератора начальной популяции является создание таких индивидов, с которыми смогут корректно работать остальные операторы.

Стоит заметить, что генератор начальной популяции допускает наличие эвристик, потенциально приводящих к ускорению процесса поиска решения приемлемого качества. Так при наличии эффективных алгоритмов, позволяющих приемлемо быстро отсеивать потенциально непригодные решения, генератор начальной популяции может сразу генерировать улучшенную начальную популяцию [2,5,6]. С другой стороны, применение эвристик, корректирующих результат работы генератора начальной популяции и слишком сильно сужающих спектр генерируемых индивидов, отрицательно скажется на генном богатстве популяции и с большой вероятностью приведет к получению лишь локально оптимального решения.

1.5 Оценка индивидов функцией приспособленности

В основе генетического алгоритма лежит принцип сравнения имеющихся решений. Для применения генетического алгоритма необходимо уметь составлять варианты решения поставленной задачи и сравнивать их между собой. Выгоднее всего применять данный алгоритм по отношению к таким задачам, у которых нет способа нахождения оптимального решения без использования сравнений среди множества вариантов решений. Оценка генетического алгоритма имеет сравнительный характер, ему неизвестна близость данного решения к оптимальному. Генетическому алгоритму известно только то, что некоторый индивид лучше или хуже другого индивида. В зависимости от того, насколько индивиды оказались лучше или хуже, их называют хорошо приспособленными или слабо приспособленными. Приспособленность реализуется при помощи степени приспособленности (пригодности) и зависит от генов данного индивида.

Функция приспособленности (также фитнесс-функция, от англ. fitness - приспособленность) показывает степень приспособленности данного индивида в качестве ответа на поставленную задачу. При помощи фитнесс-функции происходит расчет степени приспособленности данного индивида. Реализация функции приспособленности сильно зависит от конкретной задачи.

Генетический алгоритм использует значение данной функции, чтобы оценить поступающий в популяцию индивид. Так он отделяет пригодные решения от непригодных за счет сравнения значений степени приспособленности.

Одним из удобных вариантов организации функции приспособленности $fitness$, часто встречающимся на практике, является применение численной степени приспособленности, в определенном смысле показывающей близость некоторого варианта решения к оптимальному. Функция приспособленности строится таким образом, что $fitness(i)$ при оптимальном решении i принимает максимальное или минимальное значение. Стоит заметить, что само значение $fitness(i)$ может оставаться неизвестным, так как разработчику генетического алгоритма необходимо только поведение данной функции. Иными словами, задача поиска оптимального решения на некотором множестве формулируется как задача поиска минимума или максимума функции приспособленности.

1.6 Оператор выбора родителей и оператор селекции

Операторы выбора родителей и селекции управляют множеством индивидов, переходящих из одного поколения в следующее. Так на основе выбранных родителей будут созданы потомки, а выбранные в ходе селекции жертвы будут удалены из популяции. Предполагается, что родителями выбираются такие индивиды, которые потенциально приведут к образованию лучше приспособленного потомка, а жертвами – потенциально тормозящие развитие популяции.

Оператор выбора родителей выбирает несколько индивидов в количестве, необходимом для работы оператора скрещивания. Как правило, оператор скрещивания работает с двумя родителями, однако это число зависит только от конкретной задачи и выбранной архитектуры генетического алгоритма. В ходе изучения данного алгоритма было разработано множество вариантов реализации операторов выбора родителей. Как правило, в них применяются случайные величины, закон распределения которых определяется конкретной задачей.

Основной сложностью разработки оператора выбора родителей является незнание генетическим алгоритмом таких индивидов, которые наилучшим образом подошли бы в качестве родителей [6]. Понять, что данный индивид или группа индивидов подходят наилучшим образом в качестве родителей, приводящих к образованию улучшенного потомка, можно только в сравнении с другими вариантами родителей. В основе большинства операторов выбора родителей стоит гипотеза о том, что хорошо приспособленные родители приведут к образованию еще лучше приспособленного потомка с большей вероятностью по сравнению с малоприспособленными родителями. Однако разработчик при необходимости может исполнить иной вариант оператора выбора родителей.

Похожим по устройству и противоположным по назначению является оператор селекции. Данный оператор может быть описан в двух формулировках: как оператор выбора жертв и как оператор селекции. Оператор выбора жертв выбирает индивиды, которые необходимо удалить из популяции, после чего удаляет их. Оператор селекции выбирает индивиды, которые необходимо перенести в новое поколение, он каждый раз заново составляет очередное поколение. В данной работе будет рассмотрена работа оператора выбора жертв, так как он нагляднее оператора селекции.

Оператор выбора жертв имеет те же трудности, что и оператор выбора родителей: индивиды, которые не приведут к улучшению приспособленности, изначально неизвестны. В качестве решения данной проблемы используется гипотеза о том, что малоприспособленные индивиды приведут к улучшению популяции с меньшей вероятностью, чем хорошо приспособленные индивиды. При необходимости в основу оператора выбора жертв может быть заложена иная более подходящая гипотеза.

1.7 Оператор скрещивания

Оператор скрещивания служит для обработки индивидов-родителей и создания на их основе потомков. Целью оператора скрещивания является

создание таких потомков, которые бы превосходили своих родителей в степени приспособленности. Оператор скрещивания осуществляет работу с генами и может редактировать их значения. Это означает, что реализация оператора скрещивания сильно зависит от конкретной задачи. Тем не менее, разработано множество операторов скрещивания для различных генов, так как на крайне широком спектре задач возможно применение только действительных, целочисленных и логических генов.

Создание подходящего оператора выбора скрещивания является сложной задачей, так как заранее неизвестен алгоритм обработки генов, гарантированно приводящий к улучшению приспособленности [7]. Качество работы данного оператора определяется в сравнении с другими вариантами настройки оператора скрещивания. При работе с данным оператором принимается гипотеза о том, что если оператор выбора родителей удачно выбрал родителей, то и любой обмен генами потенциально приведет к созданию лучше приспособленного потомка.

Существуют различные варианты операторов скрещивания. Самым простым вариантом данного оператора является алгоритм, образующий одного потомка из случайных генов родителей. Однако в таком случае вероятность совершить неоптимальные преобразования в общем случае максимальна. Для повышения эффективности получаемых решений применяется усложненный оператор скрещивания, создающий несколько вариантов потомков. В таком случае происходит сравнение различных потомков между собой, после чего из них выбирается наилучший. С другой стороны, применение такого оператора скрещивания может значительно замелить алгоритм в случае работы с медленной функцией приспособленности.

1.8 Оператор мутации

Работа оператора мутации заключается в редактировании значений генов у создаваемых потомков. Данный оператор служит для внесения в популяцию новых вариантов значений генов. Если бы данный оператор отсутствовал, то

новые гены создавались бы только в результате генерации начальной популяции или при некоторых вариантах реализации оператора скрещивания, что привело бы к неприемлемому сужению области поиска и выводу в качестве решения заведомо некачественного результата. Реализация данного оператора сильно зависит от конкретной задачи, так как она связана с редактированием содержимого генов [8]. Как правило, в основе оператора мутации лежит случайная величина, определяющая степень изменения полученных после скрещивания значений. Чем сильнее оператор мутации изменяет содержимое генов, тем он считается мощнее.

Мощность оператора мутации является величиной, познаваемой в сравнении с другими вариантами настройки оператора мутации. Мощность мутации ограничивается сверху и снизу двумя условиями: слишком слабая мутация недостаточно влияет на поддержку генного богатства популяции, а слишком сильная мутация ухудшает результат работы операторов выбора родителей и скрещивания.

Оператор мутации может образовывать полезные и вредные мутации, однако это происходит случайно и не является основным назначением оператора мутации. Оператор мутации служит для сохранения генного богатства популяции. Генное богатство популяции – это характеристика того, насколько различные варианты значения генов находятся в популяции. Чем генное богатство выше, тем шире область поиска, на которой работает генетический алгоритм. С другой стороны, слишком богатая генами популяция расходует неприемлемо большой объем памяти и может быть засорена генами, приводящими к слабой приспособленности.

1.9 Остановка генетического алгоритма

После получения алгоритмом решения приемлемого качества, необходимо остановить работу поискового цикла. Условие, при котором популяция начинает обладать необходимым решением, называется условием

остановки генетического алгоритма. Существуют различные признаки, на которые может опираться условие остановки:

- достижение необходимой точности решения;
- уменьшение изменения состояния популяции;
- превышение лимита количества поколений;
- превышение лимита затраченного времени.

В случае нахождения приемлемого решения с заданной точностью, работа алгоритма считается успешно выполненной, однако для применения такого условия необходимо знать окрестность глобально оптимального решения. Условие уменьшения прогресса между поколениями является универсальным, для его использования не требуется знание окрестности глобально оптимального решения [9,11]. С другой стороны, данное условие будет останавливать неправильно настроенный алгоритм в окрестности локального минимума.

Время работы алгоритма можно измерить двумя путями: количеством обработанных поколений и количеством затраченного астрономического времени. Первый вариант предпочтительнее, так как он позволяет сравнивать качество генетических алгоритмов без учета производительности вычислительного устройства, использованного во время опыта. Данные ограничения чаще применяются для исследования самого генетического алгоритма, нежели на практике, так как они позволяют легко организовать сравнение производительности различных конфигураций генетического алгоритма.

1.10 Трудности разработки генетического алгоритма

Генетический алгоритм, являясь мощным средством решения оптимизационных задач, имеет множество трудностей в настройке. При разработке генетического алгоритма существует серьезная проблема, сложность решения которой растет по мере использовании данного алгоритма. Она заключается в том, что получение эффективной конфигурации

генетического алгоритма намного сложнее разработки первых версий такого алгоритма. Генетический алгоритм позволяет очень быстро начать решение задачи, однако эффективная конфигурация данного алгоритма может быть получена после продолжительных экспериментов в условиях конкретной задачи. Перед разработчиком стоит множество вопросов, связанных с выбором функции приспособленности, отображения вариантов решений в виде генов, генетических операторов и иных особенностей архитектуры генетического алгоритма.

Таким образом, генетический алгоритм имеет большой потенциал, который трудно раскрыть, так как данные алгоритмы невзаимозаменяемые. Конфигурация, пригодная для решения одной задачи, оказывается непригодной для решения другой задачи. Иными словами, каждая попытка применить генетический алгоритм к некоторой проблеме является лишь очередным вариантом приспособления генетического алгоритма к конкретной задаче, а достижения в решении одной задачи, скорее всего, окажутся непригодными для решения другой задачи.

Глава 2 Математическая модель элементов генетического алгоритма

2.1 Популяция

Популяции в генетическом алгоритме – это множество хранящихся в памяти индивидов. В рамках данного исследования популяция рассматривается в качестве функционального хранилища, поддерживающего добавление и удаление индивидов, а также имеющего определенную структуру. В популяции хранятся индивиды только с известным показателем приспособленности.

Самым простым вариантом популяции является популяция неограниченного размера. Размер такой популяции не убывает, поэтому генетический алгоритм хранит все результаты своей работы. Имея неограниченную популяцию, генетический алгоритм будет владеть большим числом вариантов индивидов, что в некоторой степени положительно скажется на качестве получаемого решения. Однако неограниченная популяция невыполнима на реальных устройствах и ее размер необходимо контролировать [10].

Применимым на практике вариантом популяции является популяция постоянного размера. В ней на место старых индивидов ставятся новые. Так расход памяти стабилизируется за счет удаления предыдущих результатов работы алгоритма. Однако перед разработчиком встает вопрос об определении размера популяции. Необходимо выбрать популяцию такого размера, чтобы она занимала разумное пространство, но при этом не приводила бы к генному обеднению популяции.

Одним из способов ограничить размер популяции и при этом сохранить запас генного богатства является применение популяции с ассоциацией по значению степени приспособленности. В такой популяции задается два ограничения: наибольшее число уникальных значений степени приспособленности и наибольшее число индивидов, имеющих одинаковую степень приспособленности. Данный подход позволяет оперировать

индивидами внутри уменьшенной группы и не влиять на всю популяцию в целом.

В рамках теории генетического алгоритма популяция не предполагает наличие какой-либо сортировки. Однако применение сортировки по степени приспособленности на практике оказывается крайне удобным решением. Наличие заранее отсортированной по степени приспособленности популяции позволяет существенно сократить объем работы, выполняемой некоторыми реализациями операторов выбора родителей и жертв. Более того, описание условий поиска у данных операторов получает необходимую наглядность.

Стоит заметить, что отсортированная популяция проще поддается анализу: аналитик сразу видит качество индивидов и может приступать к сравнению работы различных конфигураций генетического алгоритма без предварительных действий с данными. Так следить за прогрессом поиска можно при помощи составления списка из самого приспособленного индивида в очередном поколении, при этом в отсортированной популяции он займет крайнее положение.

2.2 Генератор начальной популяции

Назначение генератора начальной популяции заключается в создании некоторого числа индивидов, на основе которых будет проводиться последующее улучшение популяции. На генератор начальной популяции не возлагается задача по созданию качественных индивидов, однако применение эвристик, приводящих к генерации хороших индивидов, может заметно ускорить нахождение улучшенного решения. Так при известной конечной области с оптимальным решением имеет смысл вести поиск только на этой области, не растрачивая ресурсы на обработку заведомо неподходящих решений [11].

От содержимого начальной популяции зависит быстрота и вероятность нахождения приемлемого решения. Однако, настройку хорошего генератора начальной популяции можно получить путем сравнения одних вариантов

генератора начальной популяции с другими вариантами. Получение хорошего генетического алгоритма затруднено связностью его компонент. Не существует гарантии, что если находить по очереди наилучшую реализацию операторов генетического алгоритма, то в итоге получится лучший вариант генетического алгоритма.

Рассмотрим генерацию вещественного, целочисленного и логического генов. Как правило, для вещественного и целочисленного генов необходимы ограничения, задающие область значений D_i, которые может принимать ген i. Так генератор начальной популяции образует индивиды, у которых значения генов $g_i \in D_i$.

По способу выбора значения генов g_i генераторы начальной популяции можно разделить на детерминированные и недетерминированные. Детерминированные генераторы начальной популяции образуют индивиды по известному алгоритму. Достоинствами таких генераторов является простота и предсказуемость, однако детерминированный генератор индивидов может исказить процесс поиска конфигурации других операторов генетического алгоритма [8,14]. Детерминированный генератор начальной популяции легко поддается настройке, а генерируемые значения явно задаются разработчиком. Применение такого генератора необходимо в случае гарантированного сохранения генного богатства популяции.

Недетерминированные генераторы начальной популяции образуют индивиды со значениями генов g_i, выбранными из D_i на основе некоторого закона распределения. Достоинством недетерминированного генератора начальной популяции является возможность проверки работы алгоритма на различных вариантах начальной популяции. С другой стороны обращение с недетерминированным генератором начальной популяции сложнее из-за его меньшей предсказуемости. Настройка недетерминированного генератора начальной популяции заключается в выборе такого распределения F_i,

применение которого не приведет к уменьшению генного богатства популяции и позволит быстрее улучшать индивиды.

Таким образом, генератор начальной популяции является первым способом задания направления поиска. При помощи генератора начальной популяции разработчик генетического алгоритма может задавать область поиска и помогать алгоритму быстрее находить приемлемое значение.

2.3 Операторы выбора родителей и жертв

2.3.1 Панмиксия и метод ранжирования

Простейшим вариантом выбора индивида из популяции является панмиксия. Использование панмиксии дает каждому индивиду в популяции равные шансы быть выбранным в качестве родителя. Так в популяции из N индивидов у каждого индивида вероятность p_i быть выбранным равна $\frac{1}{N}$. Польза применения данного оператора зависит от числа потенциально хороших G и плохих B индивидов в данной популяции. Так, вероятность выбрать потенциально хороший индивид $p(G)$ равна $\frac{G}{B}$. Если данная вероятность крайне низка, из-за чего генетический алгоритм неприемлемо слабо улучшает индивиды, то следует отказаться от использования панмиксии. Тем не менее панмиксия часто применяется на практике в качестве тестового варианта оператора выбора родителей или жертв и позволяет быстрее приступить к отладке алгоритма.

В случае, когда потенциально полезных индивидов в популяции недостаточно, необходимо отдавать им предпочтение чаще, чем другим индивидам. Это означает, что хороший индивид необходимо чаще выбирать в качестве родителя, а плохой индивид – в качестве жертвы. Отдать предпочтение отдельным индивидам позволяет метод ранжирования. Метод ранжирования применяется на отсортированных по степени приспособленности популяциях и отдает предпочтение индивидам на основе их положения.

Вероятность выбора индивида p_i при использовании метода ранжирования равна $\frac{2}{N}(1-\frac{i-1}{N-1})$. Так $p_1 = \frac{2}{N}$, а $p_N = 0$ (изображено на рисунке 2.1). Так метод ранжирования позволяет компенсировать малое число хорошо приспособленных индивидов за счет неравномерной частоты выбора хороших и плохих индивидов.

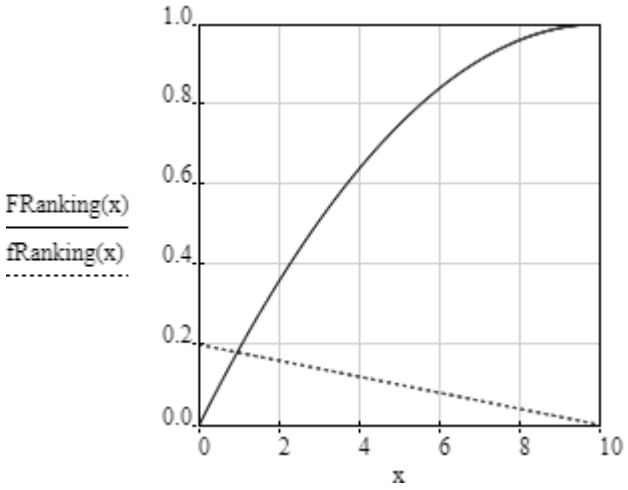

Рисунок 2.1 – Функция (сплошная линия) и плотность (пунктирная линия) распределения метода ранжирования

В связи с тем, что функции распределения данных методов выбора индивидов не изменяются в зависимости от состояния популяции, алгоритм может работать неэффективно. В таком случае необходимо применять иные способы выбора индивидов, чья работа основана на более детальном учете значений степени приспособленности и генов.

2.3.2 Метод рулетки

Дальнейшим развитием идеи перераспределения вероятностей, заложенной в метод ранжирования, является метод рулетки. Для понимания сущности метода рулетки достаточно представить вращающийся барабан с несколькими секторами некоторого размера и неподвижный указатель, который при остановке барабана укажет на случайный сектор. Вероятность того, что данный сектор остановится под указателем, пропорциональна размеру (длине

дуги) данного сектора. На рисунке 2.2 изображена модель устройства, выбирающего случайный сектор.

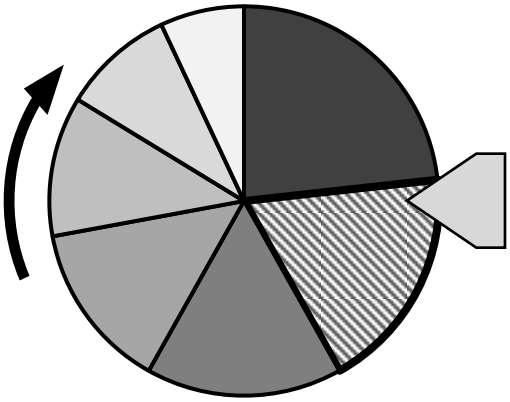

Рисунок 2.2 – Рулетка с секторами разного размера

Для того чтобы применить метод рулетки к выбору индивидов, необходимо ввести зависимость размера сектора от некоторой величины, описывающей индивид. Как правило, данной величиной является степень приспособленности. Так, если нам потребуется чаще выбирать хорошо приспособленных индивидов, то за такими индивидами необходимо закреплять более крупные секторы [15].

Рассмотрим пример, в котором степень приспособленности выражена некоторым число. Чем больше значение степени приспособленности, тем лучше индивид приспособлен. Вычисление размера секторов для учащенного выбора хорошо приспособленных индивидов будет производиться как $l_i = \frac{f(i)}{S}$, где $f(i)$ – степень приспособленности индивида i, а S – сумма всех степеней приспособленности в популяции. Можно видеть, что $\sum_{i=1}^{N} l_i = 1$, а потому при выборе равномерно распределенного случайного числа на промежутке от $[0;1]$ с вероятностью $p = l_i$ будет выбран индивид i.

Достоинством метода рулетки является необязательность сортировки популяции. Более того, разработчик может самостоятельно реализовать вычисление величины l_i наиболее подходящим для данной задачи образом. Недостатком метода рулетки является необходимость сравнения одних

варианты конфигурации данного оператора с другими, чтобы получать хорошие варианты конфигурации генетического алгоритма.

2.3.3 Турнирный отбор

Основной идеей турнирного отбора является распределение индивидов в популяции в небольшие группы, из которых затем и будут отбираться индивиды. Например, нам необходимо отобрать M индивидов, подходящих по некоторому критерию. Для этого из популяции размером в N индивидов необходимо сформировать M подгрупп некоторого размера (как правило размер группы ограничивается 2-3 индивидами). При этом допустимо добавление индивида сразу в несколько подгрупп. Затем из каждой подгруппы по заранее определенному критерию выбирается наиболее подходящий индивид, побеждающий в турнире.

Критерий выбора победителя в турнире может быть детерминированным или основанным на случайной величине. Детерминированный выбор победителя описывается правилом, по которому некоторый индивид будет признан лучше остальных только на основе его характеристик (например, выбор индивида с наибольшей степенью приспособленности). Отбор победителя на основе случайной величины заключается в совмещении характеристик индивида и значения случайной величины (например, выбор индивида с наибольшей суммой значения степени приспособленности и случайного числа).

Основной задачей турнирного отбора является подавление негативного явления, характерного для методов ранжирования и рулетки: применение данных методов делает выбор слабо приспособленных индивидов на практике неприемлемо редким. Это губительно влияет на генное богатство популяции и приводит к решениям низкого качества. Турнирный отбор обладает возможностью собрать подгруппу из слабо приспособленных индивидов и вернуть слабо приспособленный индивид в качестве результата. Таким образом, турнирный отбор позволяет избежать неприемлемого обеднения

генного богатства популяции, но при этом он не теряет способности отбирать наиболее подходящие индивиды.

2.3.4 Инбридинг и аутбридинг

Идея создания таких генетических операторов, чья работа контролировалась бы автоматически самим алгоритмом, стала основой для создания операторов инбридинга и аутбридинга. Сущность инбридинга и аутбридинга заключается в выборе индивидов не только по степени приспособленности, но и по степени схожести. Оператор инбридинга ищет родителей так, чтобы они были похожи друг на друга. Оператор аутбридинга наоборот ищет наиболее отличных родителей.

Такой подход позволяет алгоритму лучше подстраиваться под состояние популяции, что делает генетический алгоритм адаптивным. С другой стороны, данные операторы сложнее. Во-первых, необходимо ввести некоторую функцию $\rho(i, j)$ для оценки различий между индивидами i и j. Во-вторых, необходим алгоритм поиска родителей, соответствующих необходимым критериям схожести или различия. Функция оценки различий может быть исполнена в двух вариантах, именуемых фенотипным и генотипным. Фенотипная функция оценки различий использует степень приспособленности имеющихся индивидов, в то время как генотипная функция оценки различий устроена сложнее и работает с генами имеющихся индивидов.

Использование данного оператора выбора родителей предполагает наличие другого алгоритма, выбирающего первого родителя. Перед формированием множества родителей из популяции некоторым образом выбирается индивид, именуемый первым родителем. Так выбор наиболее или наименее похожего индивида осуществляется относительно первого родителя. В итоге получается оператор, имеющий множество затруднительных для анализа компонентов, его настройка дополнительно усложнит процесс разработки генетического алгоритма.

Таким образом, операторы выбора родителей на основе инбридинга и аутбридинга позволяют формировать множество родителей на основании анализа содержимого популяции. Такой оператор хорошо адаптируется под состояние популяции. С другой стороны, сложность настройки генетического алгоритма с применением такого оператора возрастет.

2.4 Оператор скрещивания

2.4.1 Кроссинговер

Реализация оператора скрещивания на основе кроссинговера является моделью естественного кроссинговера. Во время кроссинговера осуществляется распределение родительских генов таким образом, чтобы потомок унаследовал гены обоих родителей. Существуют различные варианты кроссинговера, различающиеся числом точек кроссинговера.

Представим гены, содержащиеся в хромосоме индивида в виде строки. Так хромосома с N генами может быть представлена в виде последовательности $g_1, g_2, \ldots, g_N$. Точка кроссинговера – это целое число из диапазона $[1; N]$, обозначающее изменение родителя, из которого следует брать следующие гены.

Рассмотрим пример работы одноточечного кроссинговера. На рисунке 2.3 изображены входные данные для оператора одноточечного кроссинговера и результат его работы.

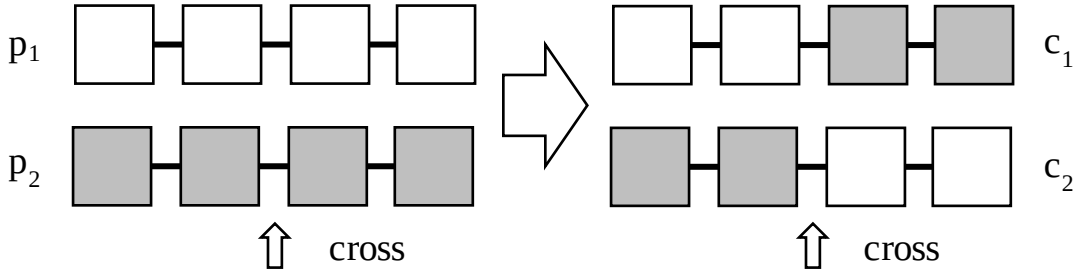

Рисунок 2.3 – Иллюстрация работы одноточечного кроссинговера

На вход оператору скрещивания от оператора выбора родителей поступают два индивида-родителя p_1 и p_2 с одной хромосомой, в которой

содержатся N генов. Оператор скрещивания должен создать потомка с одной хромосомой, в которой также будет N генов. Случайным образом выберем точку кроссинговера $cross \in [1; N]$. Составим хромосому потомка из генов обоих родителей таким образом, чтобы первые $g_1, g_2, \dots, g_{cross}$ гены были взяты от первого родителя p_1, а оставшиеся $g_{cross+1}, \dots, g_N$ гены были взяты от второго родителя p_2. В результате образуется пара потомков c_1 и c_2, которые будут содержать в себе перераспределенные гены обоих родителей p_1 и p_2. В случае, когда имеется несколько хромосом, данные действия исполняются для каждой хромосомы.

Данный оператор скрещивания легко поддается распараллеливанию. Параллельная версия оператора одноточечного кроссинговера будет отличаться от однопоточной версии только заменой циклов, перебирающих каждую хромосому и ген, на отдельные потоки, наполняющие потомков генами из нужных родителей.

Дальнейшим развитием оператора скрещивания на основе кроссинговера стал многоточечный кроссинговер. В отличие от предыдущего случая, выбираются сразу несколько точек кроссинговера. Рассмотрим применение -точечного кроссинговера на двух родителях с одной хромосомой из N генов. В данном случае будет сгенерирована последовательность из M случайных точек кроссинговера $cross_i \in [1; N]$, отсортированных по возрастанию. Так, первые $g_1, \dots, g_{cross_1}$ гены будут взяты из первого родителя p_1, следующие $g_{cross_1+1}, \dots, g_{cross_2}$ гены – из второго родителя p_2, а гены $g_{cross_2+1}, \dots, g_{cross_3}$ взяты снова из первого родителя p_1 и так далее. На рисунке 2.4 изображены входные данные для оператора двухточечного кроссинговера и результат его работы.

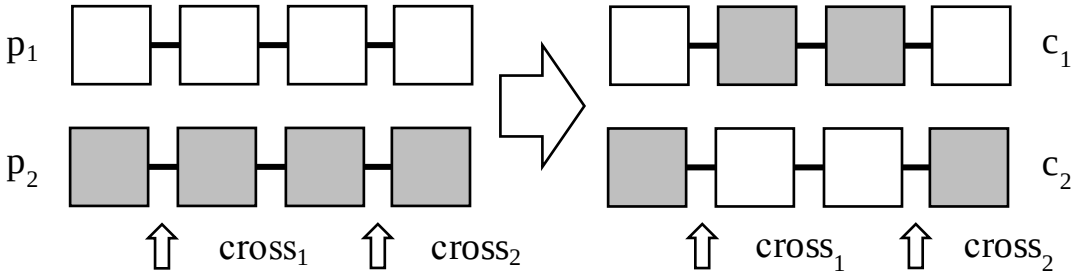

Рисунок 2.4 – Иллюстрация работы двухточечного кроссинговера

Основным отличием многоточечного кроссинговера от одноточечного кроссинговера является увеличенное число изменений, которое оператор скрещивания вносит в хромосомы. Обработка родительских индивидов при помощи одноточечного кроссинговера образует группы сцепленных генов, попадающих из поколения в поколение. Многоточечный кроссинговер имеет большую вероятность разрушить эту группу и создать новый индивид с уникальными свойствами.

2.4.2 Дискретная рекомбинация

Дискретная рекомбинация является частным случаем многоточечного кроссинговера, где каждый ген является точкой кроссинговера. Данный вариант скрещивания не имеет ограничений, связанных с типом гена. Алгоритм дискретной рекомбинации также легко поддается распараллеливанию. В случае дискретной рекомбинации не требуется выбирать точки кроссинговера, вместо этого при составлении генов потомка следует выбирать родителя, от которого будет унаследован очередной ген. Работа дискретной рекомбинации изображена на рисунке 2.5.

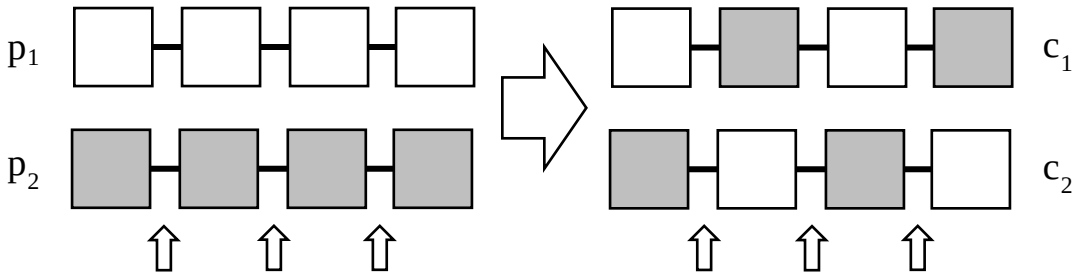

Рисунок 2.5 – Иллюстрация работы дискретной рекомбинации

В упрощенном варианте данного оператора каждый родитель имеет равные шансы передать свои гены потомку и после скрещивания образуется один потомок. Дискретная рекомбинация при необходимости может быть модифицирована путем введения эвристик, связанных с локальным поиском. Например, вероятность выбора гена у некоторого родителя может определяться на основе его степени приспособленности. Также число генерируемых потомков может быть увеличено для того, чтобы провести на множестве сгенерированных потомков поиск наилучшего варианта.

2.5 Оператор мутации

Реализация оператора мутации зависит от типа генов, к которым они применяется. В общем случае операторы мутации делятся на дискретные и непрерывные. Непрерывный оператор мутации работает на непрерывном множестве значений, как правило, применяется к вещественным генам. Дискретный оператор мутации оперирует на дискретных множествах значений. Работа оператора мутации заключается в изменении значения генов на некоторую величину [16]. Существует множество вариантов реализации данного оператора, среди которых как простые и широко распространенные реализации, вносящие случайные мутации в гены потомков, так и эвристические алгоритмы, корректирующие гены потомка с применением локального поиска.

В данной работе будет рассмотрена реализация случайных мутаций. Данный оператор характеризуется двумя величинами: вероятностью применения мутации $p_{mutation}$ и мощностью мутации γ. Вероятность применения мутации описывает то, насколько часто данная мутация будет применяться к генам. Конфигурации генетического алгоритма с $p_{mutation} \leq 0.05$ называются устойчивыми, а с $p_{mutation} > 0.05$ – неустойчивыми. Мощность мутации показывает максимальную величину, на которую данный оператор изменит значение гена: чем больше значение d, тем сильнее влияние оператора мутации.

В случае, когда оператор мутации является единственным источником новых значений генов, его работа может быть описана с применением математической модели случайных блужданий. Так, имея некоторое распределение $F_{mutation}$, описывающее характер выбора изменения $\partial \in [0; d]$, мы можем оценить скорость приближения некоторого гена к окрестности оптимального решения. На дискретном множестве решения такая скорость будет зависеть от общего числа возможных вариантов, а на непрерывном – от точности искомого решения.

В общем случае, работа оператора мутации делится на два этапа: приближение значения гена к окрестности оптимального решения на расстояние одного шага и нахождение оптимального значения в окрестности оптимального решения. В первом случае мощность мутации γ необходимо увеличить, чтобы сократить число совершаемых шагов. Во втором случае – γ наоборот необходимо уменьшить, ведь вероятность попадания в окрестность ε точного решения вычисляется как $p_{solution} = {}^{\varepsilon}\gamma$ (при $F_{mutation}$, являющейся равномерным распределением).

Более сложным вариантом оператора мутации является сборный оператор мутации. Он состоит из отдельных операторов мутации, работающих не с целым индивидом, а с подмножествами его генов. Например, индивид, состоящий из генов различного типа, необходимо обрабатывать при помощи сборного оператора мутации. Также сборный оператор мутации следует применять, когда гены индивида изменяются по разным правилам, например, когда они имеют различную точность и область определения.

Таким образом, перед разработчиком генетического алгоритма возникает задача по настройке оператора мутации, которая включает выбор его архитектуры и настройку управляющих параметров. Наличие у данного оператора свойств, имеющих противоположное влияние на ход работы алгоритма при разных состояниях популяции, приводит к поиску компромиссных решений.

2.6 Моделирование случайной величины

Генетические операторы делятся на детерминированные и недетерминированные. Детерминированные генетические операторы не используют случайные величины, результат их работы зависит только от входных данных. Недетерминированные генетические операторы используют случайные величины, значение которых влияет на результат работы генетического оператора. Такие генетические операторы вносят непредсказуемость в ход работы алгоритма.

С точки зрения реализации случайной величины удобным является решение, заключающееся в последовательном выполнении двух этапов: моделировании равномерно распределенной случайной величины и моделировании неравномерно распределенной случайной величины с использованием значений равномерно распределенной случайной величины. Данный подход также называется методом обратной функции. Сущность метода обратной функции заключается в коррекции равномерно распределенного случайного значения x' при помощи обратной функции F' таким образом, чтобы значение $x = F'(x')$ было распределено по некоторому закону F с плотностью f, отличному от равномерного распределения. Получение обратной функции связано с вычислением интеграла (2.1) от плотности f моделируемого распределения, который может быть найден численно с заданной точностью.

$$x = \int_0^{x'} f\ y\ dy \qquad (2.1)$$

Наличие недетерминированных генетических операторов требует накопления статистики для оценки конфигурации генетического алгоритма. Построение характеристики хода работы алгоритма заключается в запуске алгоритма при данной конфигурации n раз. Затем необходимо построить осредненные результаты работы алгоритма [3,7,13].

Таким образом, реализация случайной величины с задаваемым распределением может быть реализована на основе равномерно

распределенной случайной величины при помощи метода обратной функции. В связи с тем, что ход работы алгоритма будет разным при каждом запуске (из-за применения случайных величин в генетическом операторе), необходимо проводить статистический анализ собранных данных, получая осредненные результаты, к которым будет стремиться алгоритм.

2.7 Проблемы поиска конфигурации генетического алгоритма

Множество генетических операторов, которые включаются в конфигурацию генетического алгоритма, могут быть описаны в виде математической модели. Исследование данных моделей позволяет анализировать их поведение, однако, анализ поведения всего алгоритма – сложная задача. Сами по себе операторы не несут в себе данных о том, как именно они повлияют на систему в целом. Из-за этого, даже владея математическими моделями генетических операторов, разработчик все еще должен многократно запускать множество конфигураций и оценивать их путем сравнительного анализа.

Таким образом, поиск настроек генетических операторов также является трудной задачей, связанных с многочисленными вычислительными экспериментами. Разработка инструкций по применению и настройке данных операторов возможна только с получением данных о том, как шла работа алгоритма с данным оператором на множестве различных задач.

Глава 3 Реализация системы испытания генетического алгоритма

3.1 Описание системы испытания генетического алгоритма

Испытание генетического алгоритма реализуется запуском генетического алгоритма при указанной конфигурации. Во время испытания производится накопление данных о ходе работы алгоритма. К собираемым данным относятся различные показатели, описывающие поведение генетического алгоритма во времени. Назовем ход работы генетического алгоритма динамикой состояния популяции. Состоянием популяции в определенный момент времени называется множество индивидов, входящих в данную популяцию. Состояние популяции описывает ситуацию, над которой генетический алгоритм работает в определенный момент времени. Зная содержимое популяции в последовательные моменты времени, мы построим динамику состояния популяции. В связи с тем, что измерение реального момента времени необходимо только для оценки производительности данного алгоритма на данной системе, за момент времени примем номер поколения. Анализ динамики состояния популяции необходим для детального и эффективного изучения влияния генетических операторов на ход работы алгоритма. Изучение воздействия использованных операторов на динамику состояния популяции дает возможность делать выводы о качестве полученного генетического алгоритма.

Для осуществления анализа работы алгоритма необходимо хранить, обрабатывать и находить данные, связанные с состоянием популяции в некотором поколении. Многочисленные запуски генетического алгоритма с большим числом поколений при различных конфигурациях приведут к накоплению больших объемов данных, которые неудобно обрабатывать вручную. Для организации эффективного исследования необходимо организовать базу данных, хранящую результаты испытаний генетического алгоритма при различных конфигурациях. Однако только хранения данных

недостаточно: для продуктивного исследования необходимо извлекать нужные показатели из множества имеющихся. Более того, необходим простой и понятный интерфейс, позволяющий наглядно выводить результаты испытаний различных конфигураций генетического алгоритма. Быстрая и продуктивная аналитика с автоматизацией как можно большего числа этапов является ключевым моментом успешного исследования генетического алгоритма. Сравнивая между собой различные конфигурации генетического алгоритма за счет оценки смоделированных результатов, мы выявим признаки потенциально успешных элементов алгоритма и найдем такие настройки генетических операторов, которые образуют качественный генетический алгоритм.

Описанные выше задачи реализуемы при помощи системы, состоящей из следующих компонентов:

- база данных заднего плана с редактором данных;
- исполнитель испытаний генетического алгоритма;
- средство составления сводных таблиц и осуществления аналитических функций;

Главной трудностью при разработке данного решения является неопределенность направления дальнейшего развития разрабатываемой системы. В связи с тем, что разрабатываемый инструмент должен быть легко перенастроен под успешные направления исследования, следует оценить заранее все функции, которые потребуются позже. Реализация такого исследовательского продукта предполагает наличие наибольшей возможной гибкости для процесса исследования и простоты для сопровождения системы. Поддержка данного решения включает в себя быстрое и простое изменение структур обрабатываемых данных, элементов пользовательского интерфейса, генетических операторов, аналитических операций, вариантов сводных таблиц. При этом необходима возможность добавления новых функций.

Процесс испытания генетического алгоритма требует значительных вычислительных ресурсов. В связи с тем, что генетический алгоритм содержит

операторы, в основе которых лежат случайные величины, необходимо организовать сбор статистики для получения осредненных данных, то есть необходимо многократно запускать генетический алгоритм с одной и той же конфигурацией. На практике вероятна ситуация, когда функция приспособленности окажется самой долгой и затратной операцией, скорость исследования станет неприемлемо низкой. С другой стороны, отдельные испытания генетического алгоритма не влияют друг на друга, а потому могут исполняться независимо. Для организации таких испытаний необходим способ работы с такими процессами в виде средства исполнения долгих процессов.

Реализуем пользовательский интерфейс при помощи web-страниц. Данный способ исполнения пользовательского интерфейса позволяет единым образом задать интерфейс, чтобы однообразно отобразить его на различных операционных системах и устройствах. Развитие web-браузеров для мобильных платформ позволяет работать с системой испытания генетического алгоритма даже на мобильных устройствах. Пользовательский интерфейс будет включать в себя интерфейс редактора данных в базе данных, интерфейс управления средством исполнения долгих процессов, интерфейс средства составления сводных таблиц и прочие меню для удобного использования системы (например, формы для быстрого создания новых конфигураций генетического алгоритма).

Таким образом, систему испытания генетического алгоритма составят: база данных заднего плана, исполнитель испытаний генетического алгоритма, средство составления сводных таблиц и осуществления аналитических функций и пользовательский интерфейс данных компонентов. Во время разработки решения следует учитывать, что его поддержка будет включать значительные изменения. Требуется обеспечить наибольшую возможную гибкость, позволяющую легко и быстро изменять и расширять разработанную систему.

3.2 Абстрактный генетический алгоритм

Работа исполнителя испытаний генетического алгоритма заключается в получении и реализации конфигурации генетического алгоритма, запуске генетического алгоритма с данной конфигурацией и выводе данных о динамике состояния популяции. Реализация данного компонента предполагает наличие гибкости в рамках поддержки реализаций новых генетических операторов. Несмотря на то, что генетический алгоритм имеет жесткую последовательность выполняемых операций, сами реализации выполняемых операторов могут изменяться. Следовательно, реализация генетического алгоритма должна предполагать работу с любым корректно настроенным генетическим оператором.

В рамках данной работы исследуется генетический алгоритм следующего вида:

1. Генерация начальной популяции.
2. Поисковый цикл (в него входят).
3. Выход из поискового цикла при условии остановки поиска.

В поисковый цикл входят оператор выбора родителей, оператор скрещивания, оператор мутации и оператор выбора жертв. В ходе исполнения поискового цикла происходит образование потомков в необходимом количестве и удаление потенциально непригодных индивидов в количестве, возвращающем увеличенный размер дополненной новыми индивидами популяции к нормальному значению.

Возникающий уровень абстракции позволяет разделить реализацию генетического алгоритма на два части: абстрактную и конкретизированную. В абстрактную войдет та часть генетического алгоритма, которая не зависит от реализации генетических операторов. Она позволит направить разработчика конкретизированного генетического алгоритма, а также избавит его от некоторых деталей углубленной работы алгоритма. Реализация абстрактной части необходима для того, чтобы запуск генетического алгоритма стал возможным сразу после предоставления реализации генетических операторов.

Сущность конкретизации генетического алгоритма описывается множеством реализаций генетических операторов, которые в правильной последовательности исполнит абстрактная часть. В связи с тем, что генетический алгоритм является единым способом решения обширного множества задач, конкретизированная часть необходима для описания таких элементов генетического алгоритма, которые приспособят его для решения конкретной задачи. Следовательно, возникает разделение реализации на абстрактный генетический алгоритм, работающий с абстрактными генетическими операторами, абстрактными индивидами в абстрактной популяции и на конфигурацию генетического алгоритма, конкретизирующую элементы данного алгоритма.

Производительность генетического алгоритма с описанной архитектурой можно дополнительно увеличить при помощи параллельных вычислений. Рассмотрим поисковый цикл подробнее: он состоит из элементов, создающих индивиды и элементов, удаляющих индивиды. Выполнение параллельного добавления и параллельного удаления индивидов не нарушит основную последовательность генетических операторов, где сначала идет добавление потомков, а затем происходит удаление потенциально малопригодных индивидов. Так, заключив последовательность операторов выбора родителей, скрещивания, мутации и определения степени приспособленности в предопределенный процесс создания индивида, а оператор выбора жертв – в предопределенный процесс удаления индивида, мы получим два вида процессов, которые внутри могут выполняться параллельно. Например, в генетическом алгоритме с популяцией некоторого постоянного размера, необходимо добавить и удалить n индивидов. Это означает, что сначала можно параллельно исполнить до n процессов создания индивида, а затем – до n процессов удаления индивида. К сожалению, параллельно и создавать, и удалять индивиды нельзя, так как это нарушит последовательность операций генетического алгоритма. Например, результаты работы нескольких процессов

создания индивида не влияют друг на друга, однако для удаления индивидов необходимо дождаться завершения добавления всех индивидов.

Таким образом, исполнитель испытаний генетического алгоритма получает конфигурацию генетического алгоритма, на основе которой производит реализацию генетического алгоритма и его запуск. Предлагаемая архитектура позволяет многократно ускорить процесс разработки генетических алгоритмов, так как позволяет повторно использовать большие объемы кода (абстрактную часть генетического алгоритма и его операторы). Более того, готовый абстрактный генетический алгоритм позволит сосредоточить усилия исследователей на реализации генетических операторов.

3.3 Элементы генетического алгоритма и его операторы

Множество генетических операторов можно отнести по необходимости работы с генами к двум подмножествам: операторы, работающие с генами индивида, и операторы, не работающие с генами индивида. Операторы, которые работают с генами, как правило, изменяют содержимое индивида и их реализация зависит от конкретной задачи, указывающей структуру индивида. Данными операторами являются операторы скрещивания и мутации. С другой стороны, операторы выбора родителей и жертв часто требуют для работы только степень приспособленности индивидов в популяции, а она предварительно вычисляется при добавлении индивида в популяцию. Исключением являются операторы инбридинга и аутбридинга, которые следят за генотипами выбираемых родителей и производят их сравнение. Однако реализация даже таких операторов поддается абстракции, например, требованием некоторой функции, определяющей разницу между генотипами индивидов, от абстрактного индивида. Таким образом, внутри реализации генетических алгоритмов возникает еще один уровень абстракции, который уже нельзя отнести в абстрактную часть, но уже можно многократно использовать в различных конкретизациях. Данный подход позволит реализовать операторы

выбора индивидов однажды, чтобы затем переносить их между конкретными задачами.

Существуют различные варианты реализации популяции генетического алгоритма. В рамках данного исследования будет использоваться популяция в виде списка, индивиды в котором отсортированы по степени приспособленности. Популяция имеет постоянный размер, однако в момент после добавления потомков, но до запуска оператора выбора жертв, в ней будут временно находиться несколько лишних индивидов, которые будут удалены оператором выбора жертв [20]. В данном исследовании степень приспособленности реализуется действительным значением. При сортировке индивидов наиболее приспособленные индивиды ставятся ближе к началу. Следовательно, в зависимости от реализации функции приспособленности сортировка популяции бывает по убыванию и по возрастанию. Введение обобщенного правила сортировки, описанного выше, необходимо для использования одинаковых реализаций операторов выбора индивидов на различных задачах. Использование отсортированной популяции позволяет значительно облегчить процесс настройки операторов выбора индивидов, сделав ее крайне наглядной [21,22]. Например, стоит задача выбирать наиболее приспособленных родителей в надежде создать еще более приспособленного потомка. При наличии отсортированной популяции для выполнения этой задачи необходимо лишь чаще выбирать индивиды, стоящие в начале отсортированной популяции. Так, мы частично заменяем работу со степенью приспособленности индивидов на работу с их позициями в популяции.

Зачастую работа генетических операторов основана на использовании случайных величин. Например, оператор выбора родителей панкмиксия выбирает случайного индивида, оператор скрещивания дискретная рекомбинация составляет гены потомка из случайных генов родителей, а оператор мутации для вещественных генов вносит случайные изменения в гены потомка. Для реализации распределений сложнее равномерного, необходимо применять методы моделирования случайных величин. В некоторых

операторах предполагается наличие неравномерного закона распределения (например, в методе ранжирования). Более того, исследование может обнаружить несовершенство уже существующих распределений, а потому возникнет необходимость дать возможность описания своих законов распределения случайной величины.

Реализация пользовательских законов распределения случайной величины основана на методе обратной функции. Суть метода заключается в коррекции значений равномерно распределенной случайной величины до значений необходимого распределения при помощи обратной функции. В качестве наглядного способа задания пользовательских законов распределения была выбрана одномерная кривая Безье. Разработчик задает список значений, на основе которых будет построена кривая Безье, описывающая плотность распределения. В связи с дискретностью множеств генов и индивидов, а также возможностью дискретизации кривой Безье, плотность распределения удобнее использовать в процессе интегрирования, необходимом для получения случайной величины из равномерно распределенной случайной величины. Так, разбив плотность данного случайного распределения на взвешенные сегменты и последовательно вычитая их сумму из равномерно распределенной случайной величины, мы выберем номер сегмента, который будет распределен по заданному нами закону распределения.

Рассмотрим операторов выбора индивидов из популяции. Привычный процесс выбора индивидов выглядит следующим образом: оператор выбора индивидов получает содержимое популяции и на основе заданных исследователем алгоритмов выбирает индивид из популяции. Если необходимо выбрать несколько индивидов, то оператор применяется необходимое число раз (например, в случае с двумя родителями, оператор по очереди выберет каждого родителя). В данной работе предлагается иной подход к процессу выбора индивидов. Новый вариант оператора выбора индивидов будет получать на вход множество индивидов и выводить будет также множество индивидов. Например, для выбора двух случайных родителей такой оператор сразу создаст

множество с двумя индивидами. Несмотря на популярное решение применять системы с двумя родителями, описанный подход работы с индивидами необходим на случай дальнейшего развития системы, когда возникнет тестовая задача с переменным числом родителей.

Одним из вариантов реализации операторов выбора индивидов, предлагаемых в данной работе, является составной оператор выбора индивидов. За основу составного оператора была взята сущность механизма Pipe из технологии для разработки пользовательских интерфейсов Angular 2. Назначение данного механизма заключается в реализации некоторого преобразования данных непосредственно перед выводом их пользователю, при этом алгоритмы преобразования можно располагать друг за другом. Например, при помощи Pipe можно фильтровать элементы массива при помощи сразу нескольких фильтров. Похожим образом работает и составной оператор выбора индивидов, который реализован последовательностью других операторов выбора индивидов (при этом допустимо применение другого составного оператора выбора индивидов в качестве элемента такой последовательности). Данный оператор позволяет совместить несколько простых операторов в один сложный. В связи с тем, что оператор выбора индивидов получает на вход множество индивидов (множество индивидов в популяции), а возвращает также множество индивидов (множество родителей, множество удаляемых индивидов), то допустимо представить сложный оператор выбора родителей в виде фильтра, в котором фильтрующими слоями будут являться простые операторы выбора индивидов. Поэтапно обрабатывая исходное множество индивидов в популяции, сложный оператор выбора индивидов отсеет индивиды, не подошедшие под критерии поиска, и вернет те, которые последовательно прошли все этапы отбора.

3.4 Проектирование базы данных

Основной задачей базы данных в системе исследования генетического алгоритма является хранение ассоциаций динамики состояния популяции с

использованной конфигурацией генетического алгоритма. Конфигурация генетического алгоритма состоит из операторов генетического алгоритма, которые необходимо применить. Динамика состояния популяции состоит из последовательности состояний популяции, в которых описывается содержимое популяции в определенный момент времени. Также в базе данных необходимо хранить различные варианты настроек генетических операторов, а также готовые сводные таблицы.

В качестве основы разрабатываемой базы данных взят шаблон Active record. Данный шаблон позволяет организовать объектно-ориентированное представление данных на основе реляционной базы данных. В таком представлении каждый объекты определенной структуры реализованы строками в таблице. В ячейках такой строки хранятся значения переменных у данного объекта. Шаблон описывает способ хранения данных, позволяющий легко добавлять, удалять и редактировать хранящиеся объекты. Далее в данной работе будут применяться следующие термины относительно данных:

- структура (object type) – абстрактный тип данных, описывающий множество переменных, составляющих объект данной структуры;
- переменная (attribute) – составной элемент структуры, описывающий тип хранящихся данных;
- объект (object) – экземпляр структуры, к которому относится данный объект;
- значение переменной (attribute value) – данные, хранящиеся в базе данных с пометкой о принадлежности к переменной некоторого объекта.

Использование шаблона Active record привычным образом без дополнительной доработки невозможно, то есть, сразу описать структуры элементов генетического алгоритма с их переменными в виде отдельных таблиц не получится из-за недостаточной гибкости получаемого решения. Так в случае изменения структуры хранящихся данных появится необходимость менять схему базы данных. Неизбежные действия вроде регистрации нового

генетического оператора, изменения структуры состояния популяции, а также другие изменения, связанные с развитием исследования, приведут к значительным изменениям в схеме базы данных.

В качестве решения данного недостатка предлагается прибегнуть к методам мягкого программирования (от англ. soft code) и организовать базу данных таким образом, чтобы изменение структуры хранящихся данных не требовало изменения схемы базы данных. За основу такого решения предлагается взять несколько ассоциативных массивов, содержимое которых одновременно опишет структуру хранящихся данных и сохранит эти данные. Рассмотрим множества, которые нам для этого потребуются: множество структур, множество переменных, множество объектов и множество переменных. Вместе они способны описать объекты и их значения. Отнеся некоторые переменные к некоторой структуре, мы опишем множество вошедших в нее переменных. Отнеся значения переменных к некоторому объекту, мы опишем содержимое объекта. Отнеся объект к некоторой структуре, мы опишем множество переменных, которыми объект может обладать.

В итоге получится схема, состоящая из следующих таблиц:

- таблица для типов переменных (attr_type);
- таблица для переменных (attr);
- таблица для структур (object_type);
- таблица для описания принадлежности переменных к структурам (object_attr);
- таблица для объектов (object);
- таблица для значений переменных (attr_value);
- таблица для множества структур, на которые может ссылаться данная переменная (reference).

Таблица для структур необходима для реализации множества структур, используемых в системе тестирования генетического алгоритма.

Принадлежность объекта к структуре позволит установить структуру данного объекта. Таблица для переменных необходима для реализации множества переменных, а таблица для типов переменных реализует технические способы работы с указанными переменными. Принадлежность переменной к некоторому типу (текстовому, числовому и т.д.) определит логику работы со значениями данной переменной. Таблицы объектов и значений переменных реализуют хранящиеся в базе данных объекты с их значениями.

Следует рассмотреть случай, когда одна переменная может представлять сразу множество одинаковых по типу значений. Например, индивиды в некоторой популяции могут быть описаны в виде множества индивидов, принадлежащего данной популяции. В связи с этим рассмотрим следующие варианты организации значений относительно множественности: одиночное значение (один объект имеет одно значение), множественное значение (один объект имеет неупорядоченное множество значений) и списочное значение (один объект имеет список упорядоченных значений). В рамках данного исследования одиночное и списочное значения реализуются одной записью в таблице значений переменных, а не требующее сортировки множественное значение – множеством записей.

Типы имеющихся переменных разделим на две группы: базовые и дополнительные. Базовыми являются текстовый, целочисленный, действительный и ссылочный типы. Ссылочный тип позволяет задать некоторый объект (ссылку) в качестве значения. Например, результат работы генетического алгоритма ссылается на конфигурацию, при которой данный алгоритм работал. Дополнительными типами являются те типы, которые связаны непосредственно с исследованием и его сопровождением. Например, переменная с типом «закон распределения случайной величины» является дополнительной.

Отдельно следует отметить переменные ссылочного типа. Данные переменные могут ссылаться на другие объекты, сохраненные в базе данных. Например, конфигурация генетического алгоритма может быть составлена из

ссылок на другие генетические операторы, которыми будет реализована данная конфигурация. С одной стороны, такие значения могут ссылаться не только на одну конкретную структуру. Например, операторы выбора родителей могут быть описаны в разных структурах, но все из них подходят в качестве элемента конфигурации. С другой стороны, необходимо ввести ограничение на множество допустимых значений: например, чтобы предотвратить выбор оператора выбора родителей в качестве оператора мутации. Для этого воспользуемся таблицей reference, описывающей множество доступных для выбора в качестве значения видов структур. Также данная таблица задает переменную, значением которой будет описан некоторый объект. С точки зрения данных для записи значения достаточно только идентификатора объекта, но для пользователя идентификатор не несет необходимого смысла. Для поддержания наглядности пользовательского интерфейса, ссылки на объекты сопровождаются значением описательной переменной. Например, при выборе конфигурации генетического алгоритма пользователь увидит список имен конфигураций генетического алгоритма, а система сама подставит идентификатор объекта на основе принятого пользователем выбора.

3.5 Реализация операций с данными

Ранее упомянутый шаблон Active record предполагает работу с данными при помощи операций CRUD: create (создание), read (чтение), update (обновление) и delete (удаление). Применить Active record в явном виде мы можем только к следующим данным: тип переменной, переменная, структура, объект, значение переменной. Однако в разрабатываемой системе нам необходимо получать хранящиеся в базе объекты привычным образом. Для этого необходимо ввести сервисный слой, обеспечивающий удобную работу с данными как с обычными объектами.

Опишем разрабатываемые способы работы с хранящимися объектами с точки зрения разработчика:

- создание объекта – разработчик создает объект, заполняет его данными и сохраняет в базе данных;
- чтение объекта – разработчик получает объект (например, по идентификатору) и считывает с него данные привычным образом (например, используя геттер);
- обновление объекта – разработчик получает уже имеющийся в базе данных объект и записывает в него данные привычным образом (например, используя сеттер) и сохраняет в базе данных;
- удаление объекта – разработчик удаляет объект из базы данных (например, по его идентификатору, либо предварительно получив в процессе исполнения других задач).

Разделим реализацию данного решения на три уровня: уровень запросов к базе данных, уровень моделей (предметной области) и уровень абстрактных объектов (предметной области). Суть решения заключается в том, что вызов некоторой функции абстрактного объекта (например, update) приводит к созданию необходимого множества моделей, на основе которых построится запрос к базе данных. В рамках предложенного решения такой объект описывается совокупностью структуры и множества значений переменных. Полученный абстрактный объект нужен для предоставления доступа к значениям переменных. В связи с тем, что все значения переменных реализованы текстовой строкой, введем слой редакторов значений. Редакторы значений нужны для преобразования текстовой строки в необходимое значение (например, целочисленное) и обратно. Также редакторы значений преобразуют множественные и списочные значения в коллекции.

Рассмотрим два варианта реализации объектов предметной области в рамках разработки на Java. В первом случае воспользуемся механизмом наследования и применим шаблон Adapter, после чего получим объект, обладающий геттерами и сеттерами с именами переменным. Во втором случае

реализуем создание объектов и заполнение их переменных при помощи механизмов рефлексии и аннотаций.

Адаптер от абстрактного объекта предметной области расширит данный объект геттерами и сеттерами для переменных, которыми данный объект обладает. Имеющиеся редакторы значений позволяют создать однообразный вариант работы с различными переменными, имеющими различные типы и множественность. В связи с тем, что число конечных объектов, задействованных в исследовании, а также их переменных будет расти, необходимо создать средство генерации автокода. Генератор автокода на основе имеющихся структур и их переменных сможет создать множество классов адаптеров, описывающих каждую структуру, задействованную в исследовании. Также данное средство позволяет уменьшить число ошибок, которые может допустить разработчик, и заметно ускоряет процесс сопровождения данного варианта редактора данных. Данный подход рассчитан в первую очередь на небольшие изменения в конкретных переменных. Сгенерированные объекты предоставляют операции, приближенные к CRUD из привычного Active record. С другой стороны, применение таких адаптеров для работы с целыми объектами требует введение многочисленных крупных конвертеров, которые станут непригодными в случае даже небольшого изменения множества структур или переменных. Для удобного чтения и создания новых объектов необходим иной подход, требующий меньше затрат на поддержание в рабочем состоянии.

Как было отмечено ранее, применение шаблона «адаптер» не является удобным способом создания и сохранения полных объектов, так как требует больших объемов однотипного обслуживающего кода. В данном случае необходимо создать такой инструмент, который бы позволял инициализировать объекты предметной области сразу, без использования множеств и последовательностей конвертеров. Для этого введем аннотации, присваивающие отношение некоторого класса Java к структуре в базе данных, а его переменных – к переменным данной структуры. В случае запроса объектов

из базы данных система самостоятельно создаст экземпляры объектов предметной области, а также заполнит переменные данных объектов значениями из базы данных. В случае сохранения новых объектов в базе данных, система по аннотациям определит какой объект необходимо создать, а также какими значениями его необходимо наполнить. Таким образом, разработчик освобождается от необходимости создавать и поддерживать множество конвертеров, что положительно сказывается на скорости и удобстве сопровождения системы исследования генетического алгоритма.

3.6 Средство составления сводных таблиц

Поисковый характер проводимого исследования требует осуществления множества вычислительных экспериментов и накопления значительных объемов данных. Рассмотрим следующий пример: имеется множество пусков генетического алгоритма при различных конфигурациях и необходимо построить осредненную динамику состояния популяции для некоторых конфигураций с целью их сравнения. Для этого необходимо выбрать те запуски, при которых были использованы указанные аналитиком конфигурации, а затем разбить их на подгруппы по признаку принадлежности к некоторой конфигурации и вычислить среднюю динамику состояния популяции для каждой конфигурации. Для эффективной обработки собранных данных необходимо средство для их поиска и удобного отображения.

Одним из таких вариантов отображения является сводная таблица, заполненная необходимыми аналитику данными. За основу данного решения была взята технология OLAP (OnLine Analytical Processing), где имеются операции извлечения срезов данных и консолидации. Действия, описанные в примере, с точки зрения OLAP описываются извлечением среза данных по значениям конфигурации с дальнейшей усредняющей консолидацией по принадлежности к конфигурации. Таким образом, средство составления сводных таблиц должно: выбирать необходимые значения из базы данных,

извлекать срез данных, осуществлять консолидацию, а также производить операции над множественными значениями.

Полученные таблицы могут быть заполнены текстовыми и числовыми данными. В случае, когда необходимо извлечь значение из другого связанного объекта, указывается структура и ее переменная, и тогда данные будут извлекаться уже из нее. В случае, когда переменная может ссылаться на разные структуры, необходимо у каждой ссылки указать переменную, с которой надо будет работать. Особенностью разрабатываемого инструмента является возможность динамического создания новых столбцов в случае работы с множественными значениями. Например, необходимо извлечь последовательность степеней приспособленности лучших индивидов в каждой популяции. Для этого достаточно указать участвующие переменные, в то время как система самостоятельно построит таблицу нужной ширины на основе размера имеющегося множества значений.

Процесс создания формата сводной таблицы заключается в последовательном выборе основной структуры и необходимых переменных, входящих в основную структуру. Далее необходимо задать условия, по которым будет осуществляться срез данных. Если срез не требуется, то такие условия должны отсутствовать. Затем необходимо выбрать переменную, на основе значений которой сформируются консолидированные данные. Например, при выборе осредненных вариантов работы генетического алгоритма необходимо использовать консолидацию по конфигурации генетического алгоритма.

Однако только отображения данных может оказаться недостаточно. Для продуктивной аналитики могут потребоваться функции над множествами, полученными в результате извлечения данных. Например, необходимо получить среднеквадратичное отклонение степени приспособленности в данной популяции. В отличие, например, от наилучшего приспособленного индивида, такие данные не сохраняются отдельно, однако могут быть вычислены в ходе анализа. Для осуществления такого действия необходимо снабдить средство

составления сводных таблиц операциями над полученными в ходе составления таблицы множествами.

В связи с тем, что генетический алгоритм содержит операторы, работа которых основана на использовании случайных величин, необходимо производить множество запусков алгоритма с одной конфигурацией. Далее на основе полученного множества значений необходимо вычислить средний вариант работы данной конфигурации. Для этого необходимо воспользоваться операцией консолидации с осреднением. В рамках данного исследования необходима консолидация относительно уникальных значений. Это означает, что у основной структуры выделяется переменная, значения которой разделят строки неконсолидированной таблицы на подмножества. В случае, когда происходит выбор вариантов работы алгоритма, а консолидирующим параметром выбрана конфигурация генетического алгоритма, то варианты работы будут разделены на подмножества с одинаковыми конфигурациями. Если консолидация является усредняющей, то на основе каждого подмножества будет построена одна строка, содержащая осредненный ход работы алгоритма.

3.7 Исполнитель задач с большой нагрузкой

В связи с тем, что процесс исследования включает в себя исполнение затратных задач в виде моделирования генетического алгоритма или составления сводных таблиц, то необходимо отдельно обеспечить их реализацию. За основу разрабатываемого решения возьмем шаблон Thread pool (пул потоков), описывающий вариант организации многопоточных алгоритмов. С точки зрения разрабатываемой компоненты процессом для данного потока является задача, например, запуск одного генетического алгоритма при данной конфигурации. В случае, когда вычислительное устройство имеет достаточную мощность для обработки сразу нескольких генетических алгоритмов, то они могут быть исполнены одновременно в нескольких независимых потоках. Такой подход позволит ускорить процесс исследования, в котором крайне распространены многочисленные однотипные вычисления.

С точки зрения данных задача описывается совокупностью условия (типа задачи), состояния, результата и времени исполнения. Так задача может находиться в следующих состояниях:

- задача не решалась (задача имеет условие, но решение еще не вычислялось);
- задача в очереди (задача не решалась, но поступила в очередь на исполнение)
- задача исполняется (идет обработка задачи);
- задача решена (решение задачи найдено);
- задача провалена (решение задачи не найдено или найдено частично, однако ход решения был прерван возникновением критической ошибки).

Работа с исполнителем задач выглядит следующим образом: сначала пользователь исследовательской системы формирует условие задачи, затем он составляет запрос на решение задачи, который необходимо исполнить. Данный подход к организации задач позволяет ускорить исследование за счет возможности повторного использования уже имеющихся условий для новых задач, например, для многократного запуска генетического алгоритма с одной и той же конфигурацией. Запрос на исполнение задачи ставит данную задачу в очередь. После прохождения очереди задача исполняется и ей присваивается результат обработки и ставится статус, обозначающий успешность выполнения.

В рамках данной работы исполнение заявок осуществляется при помощи одного исполнителя задач с большой нагрузкой. Такая система может быть смоделирована как одноканальная СМО с конечной очередью. В случае дальнейшего развития системы, связанного с ростом производимых вычислений, текущую реализацию исполнителя задач с большой нагрузкой необходимо расширить до многоканальной системы. В особых случаях допустимо применение облачных вычислений, где исполнитель задач с большой нагрузкой будет расширен до оркестратора. Такая система может быть смоделирована как многоканальная СМО с отказами.

3.8 Пользовательский интерфейс

Реализация пользовательского интерфейса системы тестирования генетического алгоритма исполнена в виде web-приложения. Она состоит из двух основных компонентов: серверной части на JavaEE и клиентской части на Angular 2. В основе архитектуры такой системы лежит схема MVC (Model-View-Controller), у которой модель используется как средство передачи данных, вид описывает отображение данных пользователю на основе данных в модели (упрощенно: клиентская часть), а контроллер реагирует на запросы изменения модели (упрощенно: серверная часть).

Архитектура серверной части может быть разделена на три уровня: транспортный уровень, сервисный уровень и уровень контроллеров. На транспортном уровне происходит получение данных от других элементов системы: базы данных и исполнителя задач с большой нагрузкой. Сервисный уровень содержит основную логику приложения и состоит из сервисов, сервисных моделей и конвертеров. Сервисы позволяют организовать и скрыть от уровня контроллеров сложную обработку данных. Конвертеры необходимы для организации обрабатываемых данных в удобном виде. Уровень контроллеров служит для организации взаимодействия серверной и клиентской частей приложения. Он состоит из контроллеров и моделей представления (он англ. view-model). Контроллер предоставляет клиентской части свой функционал, также называемый API. Стоит заметить, что контроллер должен содержать как можно меньше бизнес-логики, так как она должна исполняться на сервисном уровне. Поддержка серверной части заключается в актуализации и добавлении бизнес-логики и функций API.

Клиентская часть пользовательского интерфейса осуществляет запросы к контроллерам и отображает полученные данные в наглядном виде. Пользовательский интерфейс состоит из трех основных компонентов: редактора базы данных, форм для отображения сводных таблиц и форм для различных автоматизированных операций. В связи с тем, что формат отображаемых данных зависит от структуры данных, которая может легко

меняться, необходимо исполнить пользовательский интерфейс по методам мягкого программирования. Также следует обратить особое внимание на наглядность интерфейса и его удобство, ведь предполагается, что для использования необходимы только знания о генетическом алгоритме.

Таким образом, пользовательский интерфейс должен осуществлять наглядное отображение имеющихся данных, а также предоставлять удобную работу с функциями системы тестирования генетического алгоритма. Более того, реализация данного интерфейса должна поддерживать возможность быстрого и незатруднительного сопровождения относительно изменения имеющихся и добавления новых функций, необходимых для исследования.

3.9 Применение системы испытания генетического алгоритма

Полученное решение позволяет автоматизировать множество рутинных операций, связанных с настройкой и запуском генетического алгоритма. Возможности описанного аналитического инструмента позволят эффективно изучать влияние генетических операторов на ход работы алгоритма. Наглядный пользовательский интерфейс и удобство системы позволят сосредоточить работы исследователя на творческой части, связанной с разработкой новых генетических операторов и поиском свойств имеющихся конфигураций.

Таким образом, описанная система позволит эффективно реализовать исследование алгоритма, основанное на сравнительном анализе различных конфигураций. Данный инструмент позволит исследовать различные явления, возникающие при работе генетического алгоритма.

Глава 4 Тестовые задачи для методов оптимизации

4.1 Задача поиска минимума тестовой функции

Задача поиска экстремумов непрерывной функции является наиболее распространенным способом тестирования методов оптимизации. Сущность данной задачи заключается в нахождении экстремумов некоторой функции с наибольшей возможной точностью за наименьшее возможное время. Сложность данной задачи описывается формой функции и количеством координат у аргумента данной функции. В рамках данного исследования будут рассмотрены две непрерывные функции для тестирования методов оптимизации: функция сферы (4.1) и функция Растригина (4.2), изображенные на рисунке 4.1.

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 \tag{4.1}$$

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i)) \tag{4.2}$$

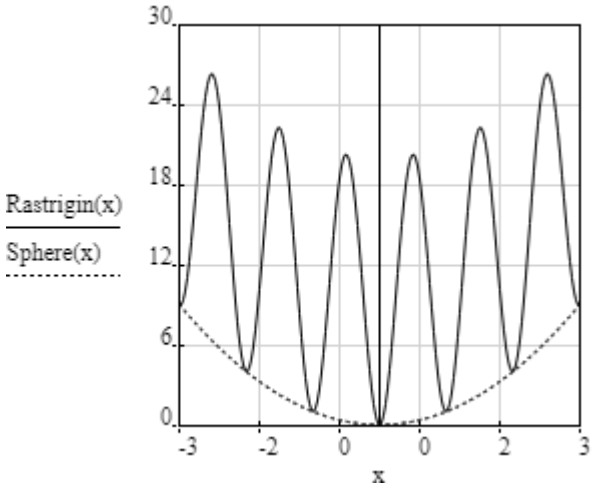

Рисунок 4.1 – Функции сферы (пунктирная линия) и Растригина (сплошная линия) от одной координаты

Глобальный минимум данных функций расположен в точке, у которых все координаты равны нулю. Можно видеть, что функция сферы монотонна

слева и справа от глобального минимума, она не имеет локальных минимумов при неограниченной области определения. В связи с этим функция сферы считается простой для оптимизации: любое улучшение приводит к глобально оптимальному решению. Функция Растригина заметно сложнее для оптимизации, так как она обладает множеством локальных минимумов. Данная функция позволяет выявлять алгоритмы со слишком жадной стратегией направленного поиска, так как в ней не каждое улучшение приводит к глобально оптимальному решению.

Данные тестовые задачи были выбраны как наиболее наглядные. Их оптимальное решение имеет простой вид, позволяющий очевидным образом оценивать близость полученных результатов к искомым. Также данные функции имеют изменяемое число координат, которыми описывается аргумент. Несмотря на большое разнообразие тестовых функций для методов оптимизации, функции сферы и Растригина достаточно наглядно отображают основные негативные явления, возникающие у плохо настроенного генетического алгоритма [23].

Сложность задачи поиска минимума некоторой функции регулируется при помощи количества координат, содержащихся в аргументе, размера области определения и формы тестовой функции. Чем больше количество координат в аргументе, тем больше значений необходимо одновременно найти, а потому такая задача сложнее. Чем шире область определения, тем больше у алгоритма вариантов возможных действий, среди которых могут оказаться неправильные. В случае непрерывной функции необходимо рассматривать отношение окрестности оптимального решения ε (точности) к области определения D. В случае дискретизации необходимо рассматривать отношение мощности множества оптимальных решений S_o к мощности множества всех допустимых значений D. Наиболее значимым критерием сложности является форма функции: чем больше число локально оптимальных решений, тем функция сложнее для оптимизации направленными алгоритмами. Функция со

сложной формой образует множество вариантов, которые алгоритм попытается проверить. В связи с тем, что основная особенность алгоритма направленного поиска – не проверять все имеющиеся варианты, существует вероятность допущения ошибки.

Следует заметить, что применение генетического алгоритма для оптимизации непрерывных функций вообще может оказаться малоэффективным. Это связано с достаточным развитием алгоритмов и методов данной области. Так, для функций, у которых возможно аналитически получить производную на рассматриваемой области определения, выгоднее находить экстремумы при помощи критических точек и проводить поиск на множестве критических точек. Генетический алгоритм эффективен в первую очередь для тех задач, у которых нет специальных (быстрых) методов решения [3,7,13].

Представление задачи поиска минимума тестовой функции в виде конфигурации для генетического алгоритма очень похоже на исходное условие самой задачи. Сначала необходимо выделить свойства искомого результата x, а именно $x: f(x) \to min$. Далее необходимо обозначить правило, по которому один индивид оценивается относительно другого: если $f\ a\ < f(b)$, то a лучше, чем b. Достоинством такого способа отладки генетического алгоритма является то, что тестовую функцию можно без дополнительных преобразований использовать в качестве функции приспособленности. Так как используется отсортированная по ухудшению степени приспособленности популяция, то алгоритм необходимо настроить на сортировку популяции по возрастанию значения приспособленности (в начале списка будут стоять индивиды, у которых наименьшее значение f). Индивид и его гены также мало отличаются от привычного представления аргумента функции: индивид описывает вариант аргумента, а его гены содержат значения координат.

Генерация начальной популяции осуществляется при помощи равномерно распределенной случайной величины. Популяция размера N

состоит из индивидов, описывающих аргументы тестовой функции размерности d. Так для каждого индивида случайным образом выбираются d чисел, лежащих в области определения D, которая находится по каждой координате в отрезке $[-\delta;\ \delta]$. Точность оптимального решения не задается, так как нам необходимо поведение алгоритма в виде эволюционной траектории. Скрещивание двух родителей осуществляется при помощи дискретной рекомбинации. В этом случае значения координат, представленные в виде генов, по отдельности извлекаются из случайного родителя и подставляются в соответствующий ген потомка. Мутация осуществляется при помощи вещественной мутации мощностью γ и вероятностью мутации $p_{mutation} = 1$.

Таким образом, задача поиска минимума тестовой функции является простым и наглядным способом отладки, тестирования и настройки генетического алгоритма. С другой стороны, это не основное назначение генетического алгоритма, а потому полученные на таком типе задач результаты должны быть преобразованы под другие варианты применения данного алгоритма.

4.2 Задача упаковки рюкзака

Задача упаковки рюкзака является задачей оптимизации на дискретном множестве решений. Сущность данной задачи заключается в наполнении некоторого абстрактного рюкзака предметами. Каждый предмет описывается двумя характеристиками – стоимостью c и весом w. Рюкзак при этом имеет ограниченную вместимость W_{max}, в него можно положить только такое подмножество из предметов I, суммарный вес $_I w$ которых не превзойдет ограничение рюкзака W_{max}. Необходимо найти такой вариант наполнения рюкзака I, удовлетворяющего ограничению $_I w \leq W_{max}$, чья стоимость $_I c$ максимальна. На рисунке 4.2 изображен пример решения задачи упаковки рюкзака с учетом формы предметов и ограниченного объема рюкзака. Жирной линией выделена форма рюкзака, внутри которой находятся предметы, чья суммарная стоимость предполагается максимально возможной. Снаружи

изображены предметы, не вошедшие в рюкзак из-за предположения, что их использование препятствует получению максимальной суммарной стоимости.

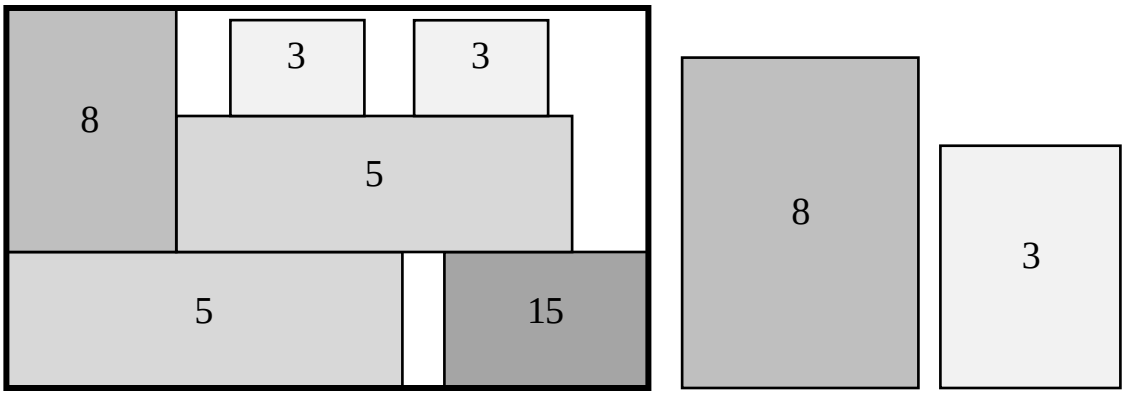

Рисунок 4.2 – Иллюстрация решения задачи упаковки рюкзака с ограничением по объему (числами указана стоимость предметов)

Данная задача похожа на задачу составления расписаний, где требуется найти такую комбинацию событий, которая бы не нарушала обязательные ограничения (не проводить занятия с большими группами в маленьких аудиториях) и при этом стремилась бы удовлетворить желательные условия (отсутствие пустых ячеек между парами). В случае с задачей упаковки рюкзака обязательным ограничением является соблюдение прочности рюкзака $_I w \leq W_{max}$, а желательным условием – максимизация стоимости рюкзака $_I c \rightarrow max$.

Множество условий и ограничений можно учитывать двумя способами: при помощи усложненной функции приспособленности или при помощи составной степени приспособленности. Усложненная функция приспособленности самостоятельно учитывает все имеющиеся ограничения и сразу возвращает значение приспособленности в виде числа. Такой подход удобен при малом количестве ограничений, но с ростом числа имеющихся ограничений неизбежно возникнут конфликты, когда несколько частичных решений будут содержать разные элементы верного решения. В таком случае ряд решений, которые стоило бы объединить для получения приспособленного

потомка, будут восприняты генетическим алгоритмом как похожие решения, слабо отличающиеся друг от друга. Данную проблему может решить применение составной степени приспособленности. Она представляет собой совокупность степеней приспособленности для каждого отдельного свойства искомого решения. В таком случае сохраняется больше данных об индивидах, что дает возможность принимать улучшенные решения, учитывающие больше особенностей состояния популяции.

Сложность задачи упаковки рюкзака регулируется при помощи числа предметов, которые необходимо задействовать, и их характеристик. Чем больше предметов необходимо подбирать, тем больше вариантов решения необходимо рассмотреть алгоритму. Регулировать наличие локально оптимальных решений можно при помощи характеристик имеющихся предметов. Чем меньше предметы различаются по весу и стоимости, тем сложнее алгоритму определить правильное направление поиска. Например, в задаче помимо прочих имеются два вида предметов: A с весом $w_A = 499$ и стоимостью $c_A = 500$, а также B с весом $w_B = 1$ и стоимостью $c_B = 1$. Можно видеть, что 500 предметов B дадут больше стоимости, чем 1 предмет A при одинаковом весе. Чтобы выбрать оптимальный вариант, алгоритму необходимо осуществить изменение в 500 раз большее, по сравнению с изменением, приводящим к похожему локальному решению.

В связи с тем, что генетический алгоритм сравнительно хорошо справляется с комбинаторными задачами, постараемся обнаружить условия, влияющие на скорость поиска. Необходимо найти общие черты в работе алгоритма над непрерывными и дискретными множествами. Анализ обнаруженных свойств позволит проще и быстрее подбирать качественную конфигурацию генетического алгоритма.

Реализация генетического алгоритма, осуществляющего поиск решения для задачи упаковки рюкзака, не требует значительных предварительных исследований и может быть реализована с использованием правил, описанных

в условии данной задачи. К свойствам искомого решения относятся обязательное условие соблюдения вместимости рюкзака $_I w \leq W_{max}$ и желательное условие максимальной цены рюкзака $_I c \rightarrow max$. Для реализации функции приспособленности предлагается подход, оценивающий приоритет имеющихся ограничений. В рамках данного исследования примем, что никакой набор предметов с нарушением вместимости рюкзака не является приемлемым решением задачи. Исходя из этого правила, сформируем функцию приспособленности таким образом, чтобы любой набор предметов, соблюдающий условие вместимости рюкзака, оказался лучше любого несоблюдающего. Данную задачу легко исполнить так, как показано в формуле (4.3).

$$f(I) = \begin{cases} c_I, & w_I \leq W_{max} \\ -w_I, & w_I > W_{max} \end{cases} \quad (4.3)$$

Можно видеть, что данная функция имеет два варианта поведения: на случай нарушенного (сверху) и ненарушенного (снизу) условия вместимости рюкзака. В случае, когда решение корректное и вместимость учтена, решения сравниваются между собой по стоимости. В случае, когда решение не учитывает вместимость рюкзака, сравнение решений осуществляется по их весу. Так при сортировке популяции по убыванию по значению степени приспособленности наилучшие решения окажутся в начале списка индивидов. Данный вариант исполнения функции приспособленности обеспечивает грубое разделение приемлемых и неприемлемых решений: они будут находиться практически в разных участках популяции. Мягкое разделение позволяет неприемлемым индивидам находиться рядом с корректными индивидами. Такой подход позволяет находить приближенные решения с допустимыми нарушениями обязательных ограничений.

Решение задачи упаковки рюкзака описано множеством вошедших в рюкзак предметов. Представим данное множество в виде ассоциативного

массива, у которого ключом является предмет, а значением – число таких предметов в рюкзаке. Так стоимость рюкзака будет вычисляться как $\sum_S c_s n_s$, а вес – как $\sum_S w_s n_s$, где S – множество используемых в задаче предметов (склад), а w_s, c_s и n_s соответственно вес, стоимость и количество предмета $s \in S$.

Генерация начальной популяции осуществляется при помощи равномерно распределенной случайной величины: каждому индивиду присваивается случайное число предметов каждого вида. Разумным в данном случае будет применение множества значений N_S такого, что любое $n \in N_S$ удовлетворяет условию вместимости рюкзака $w_s n_s \leq W_{max}$ и не является отрицательным числом (в рамках данной задаче не рассматриваются мнимые предметы с отрицательными весом и стоимостью). В свою очередь мутация индивидов будет заключаться в изменении числа предметов n_s на некоторую величину из отрезка $[-\gamma; \gamma]$, где γ – мощность мутации. Вероятность мутации $p_{mutation}$ при этом равна 1. Скрещивание родителей осуществляется при помощи дискретной рекомбинации, в ходе работы которой потомок по очереди наследует значение n_s из случайного родителя.

Задача упаковки рюкзака позволит протестировать конфигурацию генетического алгоритма на дискретном множестве решений. Анализ хода работы для поиска решений на множестве с регулируемой мощностью позволит рассмотреть различные стратегии разработки генетического алгоритма. В связи с тем, что сложность задачи существенно влияет на ход работы алгоритма, а также на состав эффективной конфигурации, реализуем данную задачу в вариантах различной сложности.

4.3 Испытание генетического алгоритма

Описанные тестовые задачи позволяют протестировать генетический алгоритм при различных условиях. В связи с тем, что поведение генетического алгоритма зависит от сложности задачи, мощности множества области ее определения и ее масштаба, необходимо использовать задачи с различными перечисленными свойствами. Так задача поиска минимума непрерывной

функции позволяет протестировать алгоритм в случае, когда у него имеется бесконечное множество вариантов решения задачи, в то время как задача упаковки рюкзака в нашем случае имеет конечное число вариантов решения. Описанный в главе инструмент для испытания генетического алгоритма содержит реализации данных задач. Можно видеть, что для решения потребовались относительно несложные алгоритмы генерации варианта решения и его оценки. Очевидно, что для решения данных задач уже разработаны более эффективные по сравнению с генетическим алгоритмом способы решения. Однако, данные задачи хорошо изучены, а потому для них были найдены особенности эффективного решения. В свою очередь генетический алгоритм рассчитан на решение слабо изученных задач, тем не менее, данные тестовые задачи позволят быстро оценить близость генетического алгоритма к оптимальному решению. Таким образом, исследование поведения генетического алгоритма на разнообразных задачах позволит формировать более качественные выводы о работе генетических операторов и о вариантах их настройки.

Глава 5 Варианты улучшения характеристик генетического алгоритма

5.1 Способы оценки генетического алгоритма

Процесс исследования, построенный на сравнении одних конфигураций генетического алгоритма с другими, требует введения оценок, описывающих достижения генетического алгоритма при данной конфигурации. Искомые оценки должны наглядно отражать влияние вносимых в конфигурации различий на ход работы алгоритма.

Существуют варианты оценки генетического алгоритма, основанные на получаемом в ходе поиска результате. К ним относятся такие величины, как вероятность достижения окрестности глобально оптимального решения, равно как и близость к глобально оптимальному решению. Чем чаще алгоритм строит наиболее точное решение, тем лучше он сконфигурирован. Однако такой вариант оценки недостаточно информативен, так как не указывает путь получения сгенерированных величин.

С целью преодоления трудностей, связанных с неполнотой данных о явлениях внутри генетического алгоритма, было предложено построение динамики состояния популяции. В таком случае появляется возможность детальнее рассмотреть ход работы алгоритма и определить характер вносимых изменений [23,27]. Одним из способов наглядного отображения хода работы алгоритма является эволюционная траектория, которая показывает динамику некоторой наблюдаемой в генетическом алгоритме величины во времени. Например, эволюционная траектория может описывать изменение содержимого индивидов или степени приспособленности.

В данной работе предлагаются следующие варианты эволюционных траекторий: степень приспособленности наилучшего индивида в поколении, средняя степень приспособленности в поколении и среднеквадратическое отклонение степеней приспособленности в поколении. Данные показатели могут быть получены построением среза данных на собранных ранее

показателях о динамике состояния популяции. Так процесс получения лучшего результата покажет практическую ценность алгоритма, а такие показатели, как средняя величина степени приспособленности и среднеквадратическое отклонение степеней приспособленности, опишут ситуацию внутри популяции.

При помощи эволюционной траектории можно определить скорость получения результата, более того, можно видеть точность полученного результата и зависимость данной точности от количества потраченного на поиск времени. Наложение нескольких эволюционных траекторий на один график позволяет крайне наглядно сравнивать и оценивать конфигурации генетического алгоритма: мы можем заявлять о превосходстве в скорости и точности, а также выявлять участки наибольшей производительности.

Таким образом, исследование генетических алгоритмов при помощи эволюционных траекторий позволяет анализировать множество свойств имеющихся конфигураций. Выявление положительных и отрицательных черт данных конфигураций позволит создавать новые эффективные способы разработки генетических алгоритмов, объединяющих достоинства ранее исследованных генетических операторов.

5.2 Подбор размера популяции

Размер популяции определяет число индивидов, одновременно хранящихся в памяти алгоритма. Чем больше уникальных индивидов содержится в популяции, тем больше данных о возможных вариантах решения известно алгоритму. В большей популяции имется больше места для потенциально пригодных индивидов, что положительно сказывается на генном богатстве популяции и позволяет алгоритму точнее находить искомое решение. С другой стороны, алгоритм начинает расходовать больше памяти, также для обработки большего числа индивидов требуется больше вычислительной мощности.

Рассмотрим работу алгоритмов с популяциями различного размера над задачей поиска минимума сложной функции Растригина, их эволюционные траектории изображены на рисунке 5.1.

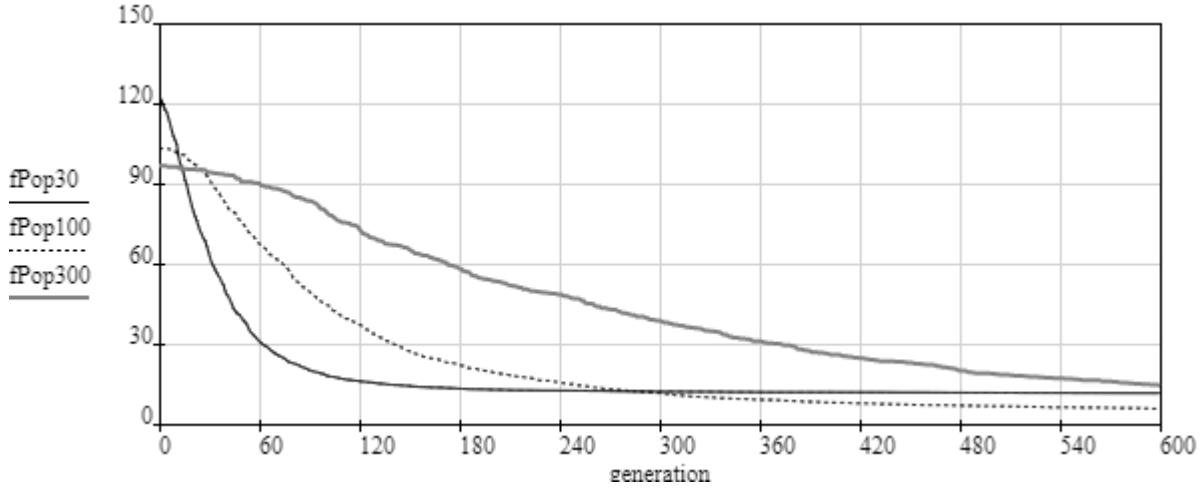

Рисунок 5.1 – Эволюционные траектории у популяций различного размера (решение задачи поиска минимума на функции Растригина)

Первым, на что стоит обратить внимание, является начальное состояние генетического алгоритма. Можно видеть, что у алгоритма с большей популяцией есть больше возможностей получить улучшенный индивид в самом начале. Так увеличение размера популяции позволяет начать поиск с более выгодного состояния. С другой стороны, алгоритм с большой популяцией получает инерцию: можно видеть, как замедляется поиск на начальном этапе. Ограниченная поисковая мощность алгоритма не позволяет сразу обработать всю популяцию и в ней какое-то время содержатся случайно сгенерированные индивиды из начальной популяции.

Следует обратить особое внимание на то, что увеличенная популяция в перспективе приводит к более точному решению. Это возникает в связи с тем, что алгоритм хранит больше данных о проделанной работе и имеет больше возможностей для построения улучшенного потомка. С другой стороны большая популяция существенно замедляет процесс поиска, так как ограниченная поисковая мощность не позволяет одинокого эффективно изменять популяцию любого размера. Так перед разработчиком алгоритма

возникает проблема выбора популяции подходящего размера, ведь необходимо в кратчайшие сроки получить достаточно качественное решение.

На рисунке 5.2 изображены эволюционные траектории алгоритмов, решавших простую задачу поиска минимума функции сферы с использованием траекторий различного размера.

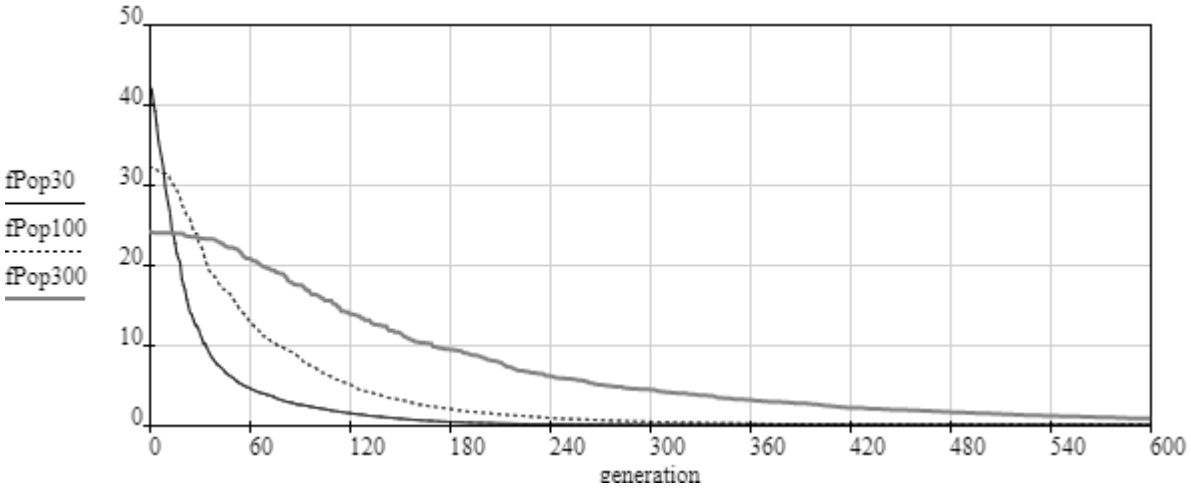

Рисунок 5.2 – Эволюционные траектории у популяций различного размера (решение задачи поиска минимума на функции сферы)

Сравнение рисунков 5.1 и 5.2 показывает, что сложность задачи не влияет на наличие инерции у поискового процесса при использовании большой популяции. Также сложность задачи не влияет на то, что алгоритм с большей популяцией начинает с более выгодного состояния. Однако можно видеть, что в условиях простой задачи крупная популяция лишь тормозит процесс поиска: на малой популяции необходимое решение генерируется существенно быстрее.

Таким образом, значением размера популяции можно задать точность искомого решения и скорость поиска. При выборе данной настройки необходимо учитывать тот факт, что большая популяция способна привести к более точному решению в перспективе. Также следует учитывать сложность решаемой задачи: на простых задачах выгодно использовать небольшие популяции, в то время как на сложных задачах необходимо соблюдать соотношение точности решения и скорости поиска.

5.3 Подбор мощности мутации

Мощность мутации является одной из настроек оператора мутации, которая описывает степень влияния данного оператора на ход работы алгоритма. Мощная мутация позволяет подойти к области шаговой доступности окрестности оптимального решения быстрее. С другой стороны, мощная мутация снижает вероятность достижения окрестности оптимального решения за один шаг.

Рассмотрим влияние мощности мутации на ход решения сложной задачи поиска минимума на функции Растригина, эволюционные траектории которого изображены на рисунке 5.3. Слева изображено полное решение задачи, а справа – фрагмент наиболее активных изменений.

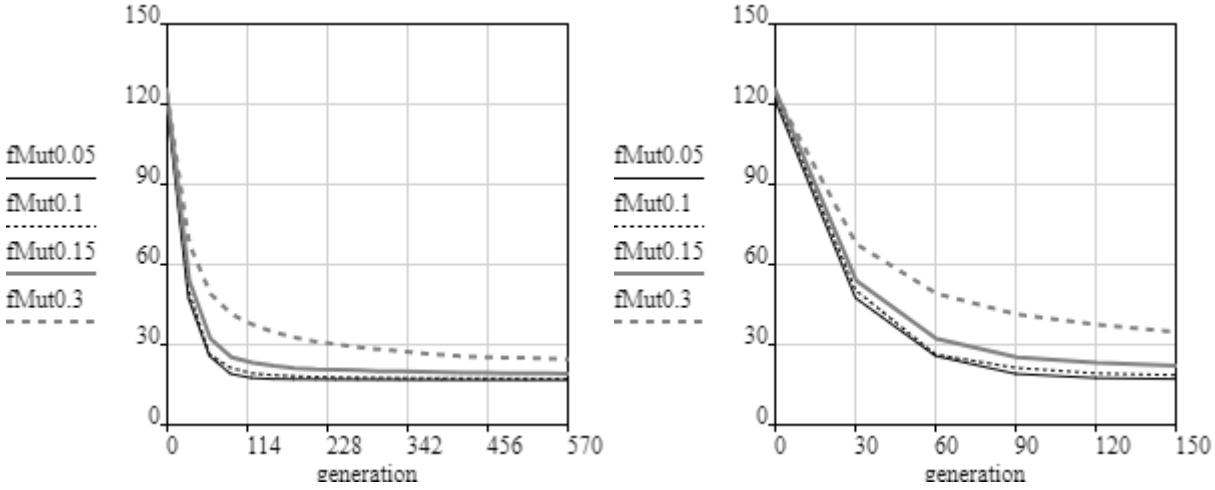

Рисунок 5.3 – Эволюционные траектории у мутаций разной мощности (решение задачи поиска минимума на функции Растригина)

Можно видеть, что слабая мутация быстрее приводит алгоритм к более точному решению. Отсюда можно сделать вывод, что при решении сложной задачи следует пользоваться не самой мощной мутацией. Очевидно, что также существует обратная ситуация, при которой слишком слабая мутация неприемлемо замедлит поиск, но она не изображена на данных графиках.

Рассмотрим влияние мощности мутации на ход решения простой задачи поиска минимума на функции сферы, эволюционные траектории которого

изображены на рисунке 5.4. Слева изображено полное решение задачи, а справа – фрагмент наиболее активных изменений.

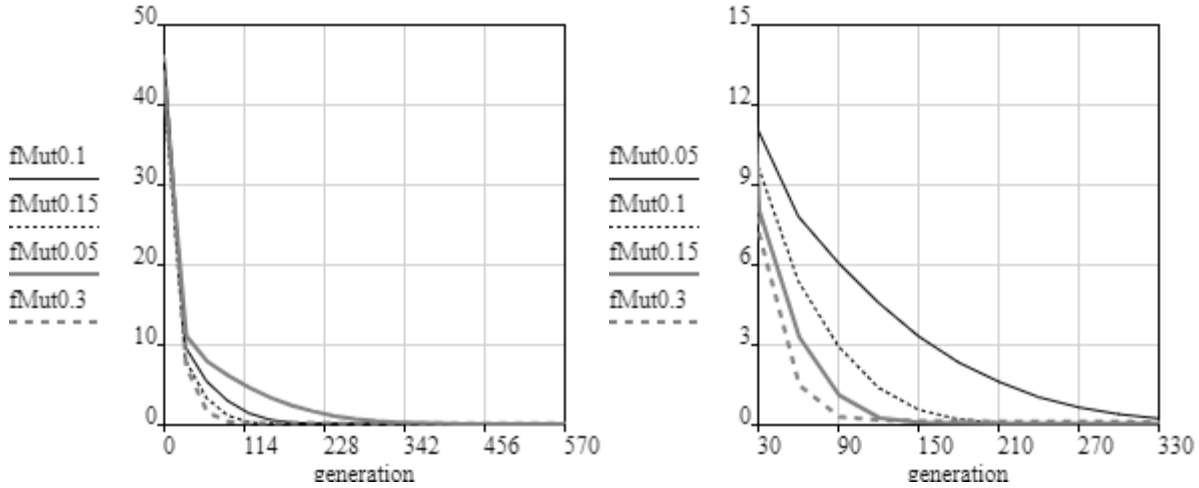

Рисунок 5.4 – Эволюционные траектории у мутаций разной мощности (решение задачи поиска минимума на функции сферы)

Можно видеть, что наибольшая мутация приводит алгоритм к окрестности глобально оптимального решения быстрее, в то время как остальные лишь замедляют поиск. Однако, следует обратить внимание на точность получаемого решения, изображенную на рисунке 5.5. Данный график показывает возникновение ранее предсказанного явления, при котором мощная мутация имеет менее выраженную способность к нахождению решений большой точности.

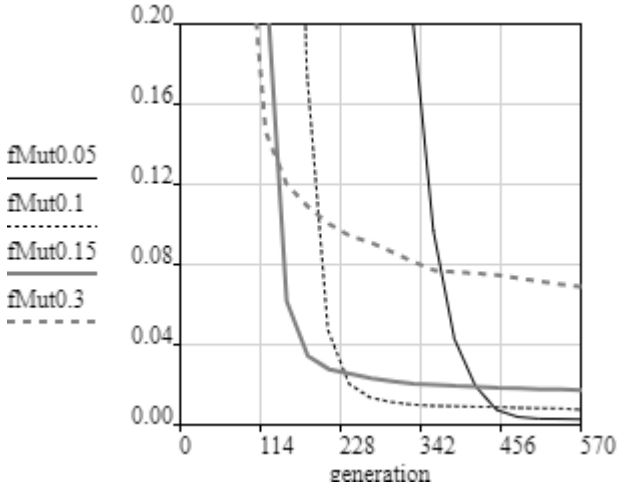

Рисунок 5.5 – Эволюционные траектории у мутаций разной мощности (решение задачи поиска минимума на функции сферы)

С целью обобщения наблюдаемого явления воспользуемся дополнительной тестовой задачей – задачей упаковки рюкзака. На рисунке 5.6 изображены эволюционные траектории сложной задачи упаковки рюкзака.

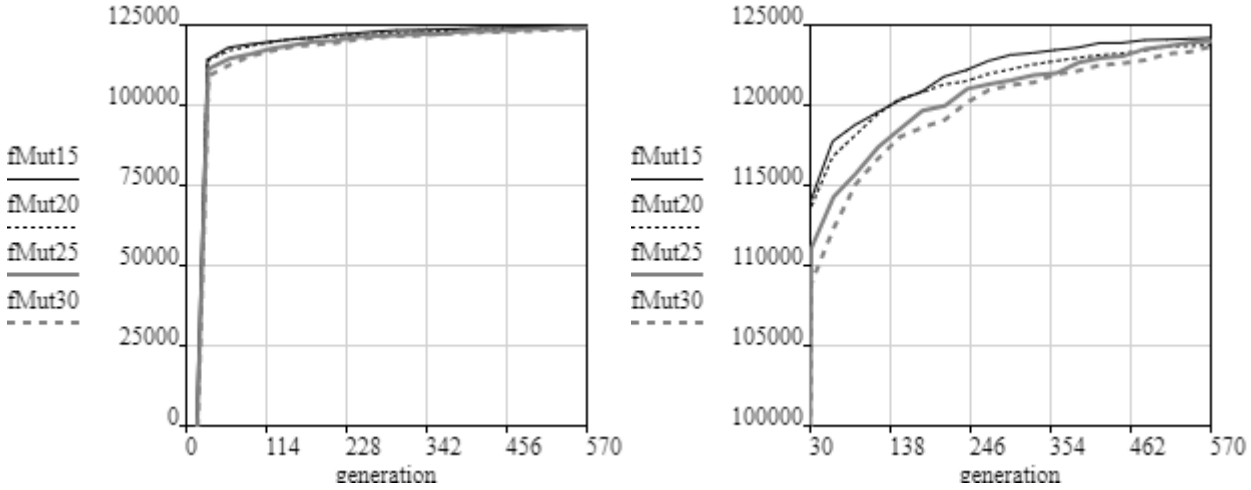

Рисунок 5.6 – Эволюционные траектории у мутаций разной мощности (решение сложной задачи упаковки рюкзака)

Сравнение эволюционных траекторий, изображенных на рисунках 5.3 и 5.6, показывает справедливость гипотезы, утверждающей превосходство менее мощных операторов мутации при решении сложной задачи. Рассмотрим работу различных операторов мутации над простой задачей упаковки рюкзака, соответствующие эволюционные траектории изображены на рисунке 5.7.

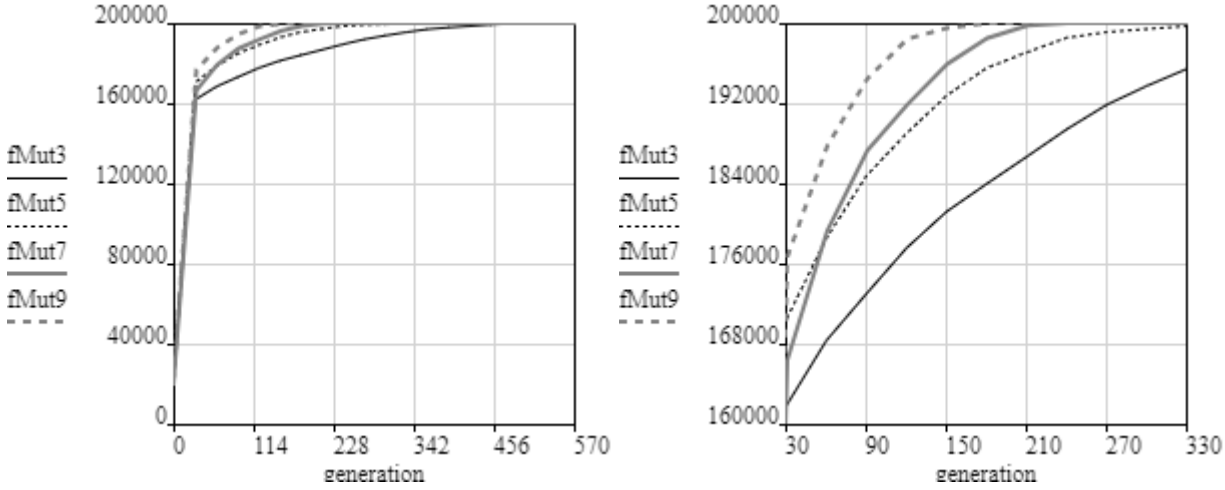

Рисунок 5.7 – Эволюционные траектории у мутаций разной мощности (решение простой задачи упаковки рюкзака)

Сравнение эволюционных траекторий, изображенных на рисунках 5.4 и 5.7, также показывает справедливость гипотезы о превосходстве мощных операторов мутации в рамках скоростного решения задачи. Явление, связанное с обратной зависимостью между точностью генерируемого решения и скоростью его нахождения, выражено слабее в случае задачи упаковки рюкзака и связано с ее дискретной природой.

Таким образом, предлагается стратегия выбора оператора мутации на основе сложности решаемой задачи. Так для простых задач приемлемым является использование мощных операторов мутации (возможно, в ущерб точности решения), в то время как для сложных задач следует искать менее мощные операторы мутации.

5.4 Подбор оператора выбор родителей

Оператор выбора родителей служит для формирования основу генерируемых потомков. Задача данного оператора – выбирать таких индивидов-родителей, которые потенциально дадут улучшенного потомка. Существует множество реализаций данного оператора, однако в рамках данного исследования рассмотрим следующие существующие варианты: панмиксию, метод ранжирования и метод рулетки. Панмиксия позволяет каждому индивиду в популяции стать родителем с равными шансами, в то время как методы ранжирования и рулетки отдают предпочтение наилучшим индивидам.

Также в данной работе предлагается центральный выбор, полученный эмпирическим путем в ходе испытания множества распределений другой формы. Стоит отметить только то, что данный оператор работал существенно эффективнее по сравнению множеством операторов других форм. Были опробованы различные комбинации участков отсортированной по степени приспособленности популяции, индивиды из которых участвовали в выборе, и эффективнее остальных работали те операторы, которые захватывали центральную область популяции. Операторы, которые работали с полюсами

популяции (только лучшие и только худшие индивиды) слабее справлялись с поставленной задачей. Характеристики оператора центрального выбора изображены на рисунке 5.8 и имеют вид (5.1) и (5.2).

$$FCentered\ x = 6\ \frac{x}{N}^{5} - 15\ \frac{x}{N}^{4} + 10\ \frac{x}{N}^{3} \quad (5.1)$$

$$fCentered\ x = 3\ \frac{x}{N}^{4} - 6\ \frac{x}{N}^{3} + 3\ \frac{x}{N}^{2} \quad (5.2)$$

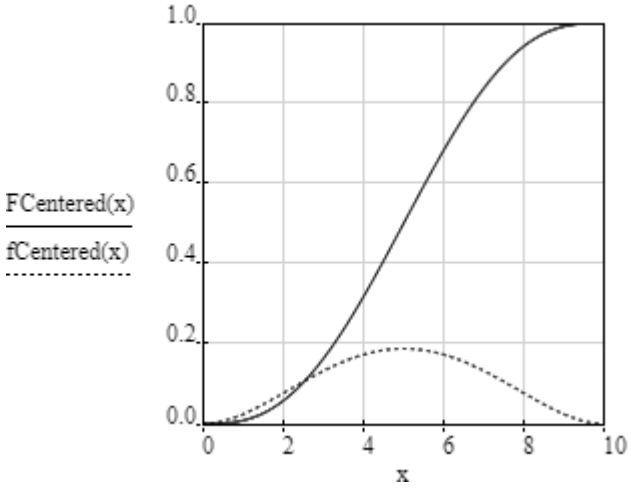

Рисунок 5.8 – Функция (сплошная линия, выражение 5.1)) и плотность (пунктирная линия, выражение 5.2)) распределения центрального выбора

Рассмотрим работу различных операторов выбора родителей над сложной задачей поиска минимума функции Растригина. Эволюционные траектории, описывающие ход решения данной задачи, представлены на рисунке 5.9.

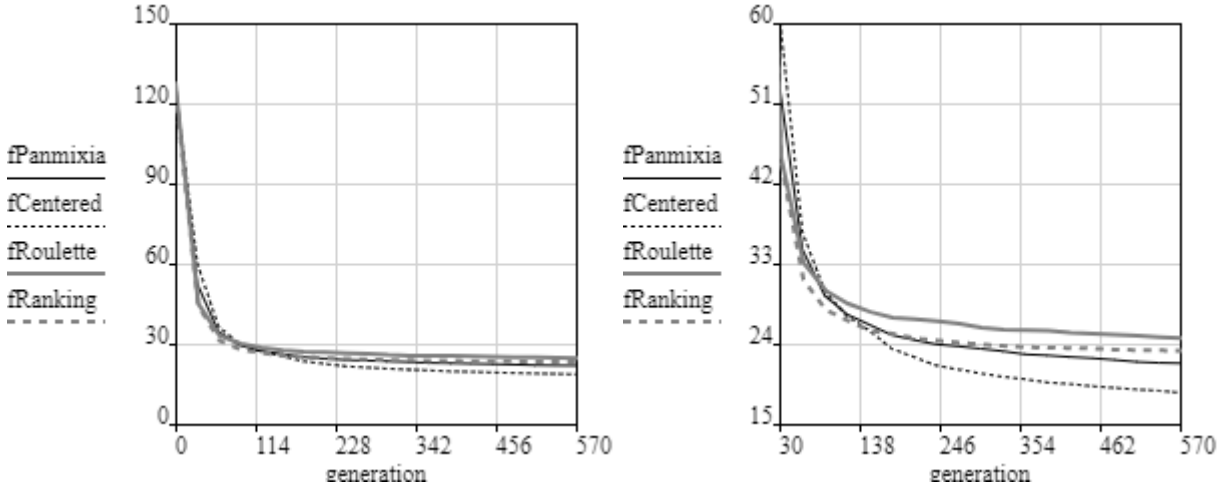

Рисунок 5.9 – Эволюционные траектории у операторов выбора родителей (решение задачи поиска минимума на функции Растригина)

Данные эволюционные траектории показывают, что алгоритмы с центральным выбором и панмиксией лучше справляются со сложной задачей поиска минимума на функции Растригина. При этом операторы выбора родителей на основе методов ранжирования и рулетки приводят алгоритм к менее точному решению. Работа данных операторов выбора родителей над задачей поиска минимума на функции сферы представлена в виде эволюционных траекторий, изображенных на рисунках 5.10.

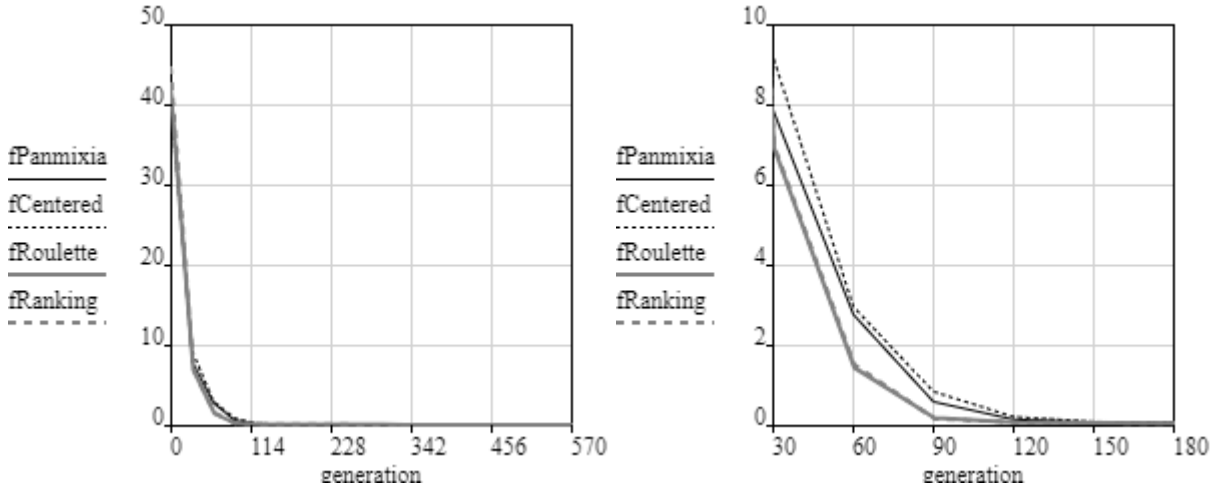

Рисунок 5.10 – Эволюционные траектории у операторов выбора родителей (решение задачи поиска минимума на функции сферы)

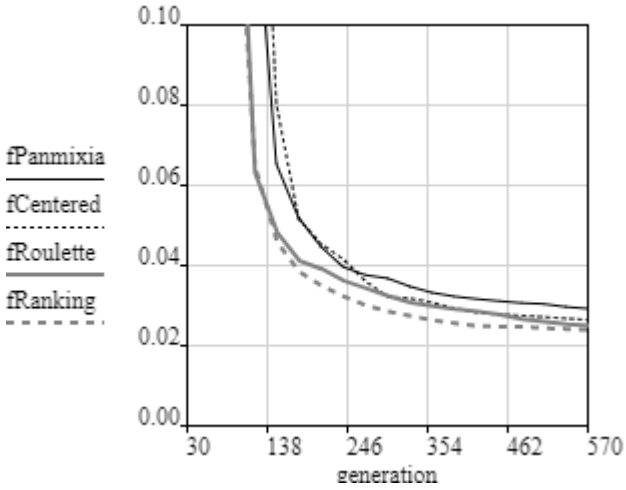

Рисунок 5.11 – Эволюционные траектории у операторов выбора родителей (решение задачи поиска минимума на функции сферы)

Можно видеть, что операторы выбора родителей на основе методов ранжирования и рулетки быстро приводят алгоритм к искомому решению. Точность найденного решения показана на рисунке 5.11.

Назовем методы ранжирования и рулетки жесткими операторами выбора родителей, так как они отдают предпочтение лучшим индивидам популяции. Оставшиеся панмиксию и центральный выбор назовем мягкими операторами выбора родителей, так как они не отдают предпочтение лучшим индивидвам. На основе рассмотренных эволюционных траекторий можно сделать закономерный вывод о том, что сложные задачи необходимо решать алгоритмами с использованием мягких операторов выбора родителей, а простые – с использованием жестких операторов выбора родителей.

Обобщим разработанную гипотезу, рассмотрев работу данных операторов выбора родителей над задачей упаковки рюкзака. Эволюционные траектории, соответствующие сложной задаче упаковки рюкзака представлены на рисунке 5.12.

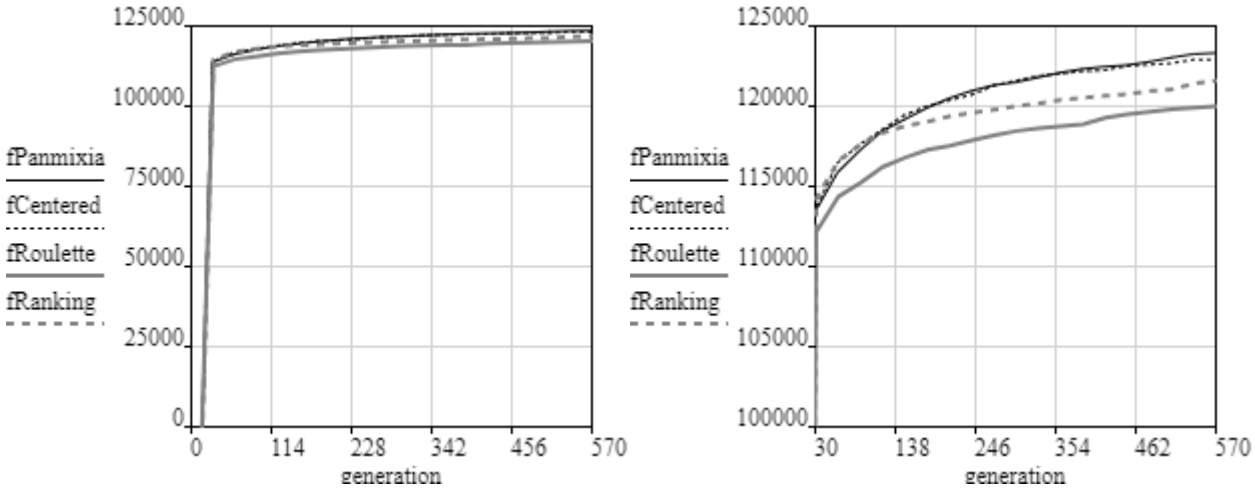

Рисунок 5.12 – Эволюционные траектории у операторов выбора родителей (решение сложной задачи упаковки рюкзака)

Сравнение эволюционных траекторий, изображенных на рисунках 5.9 и 5.12, подтверждает справедливость разработанной гипотезы и на непрерывной задаче поиска минимума функции Растригина, и на дискретной сложной задаче упаковки рюкзака: в обоих случаях с задачей лучше справляются мягкие

операторы выбора родителей. Жесткие операторы выбора родителей наоборот не приводят алгоритм к лучшему результату, хотя и имеют временное преимущество в скорости поиска. Интересным является тот факт, что среднеквадратическое отклонение степеней приспособленности во времени у жестких и мягких операторов не имеет существенной разницы, а потому проблема резкого падения прогресса у жестких операторов выбора родителей не связана с приведением генного богатства популяции в негодность и заключается непосредственно в самом алгоритме выбора индивидов.

Далее необходимо рассмотреть работу операторов выбора родителей над простой задачей упаковки рюкзака, связанные с ней эволюционные траектории изображены на рисунке 5.13.

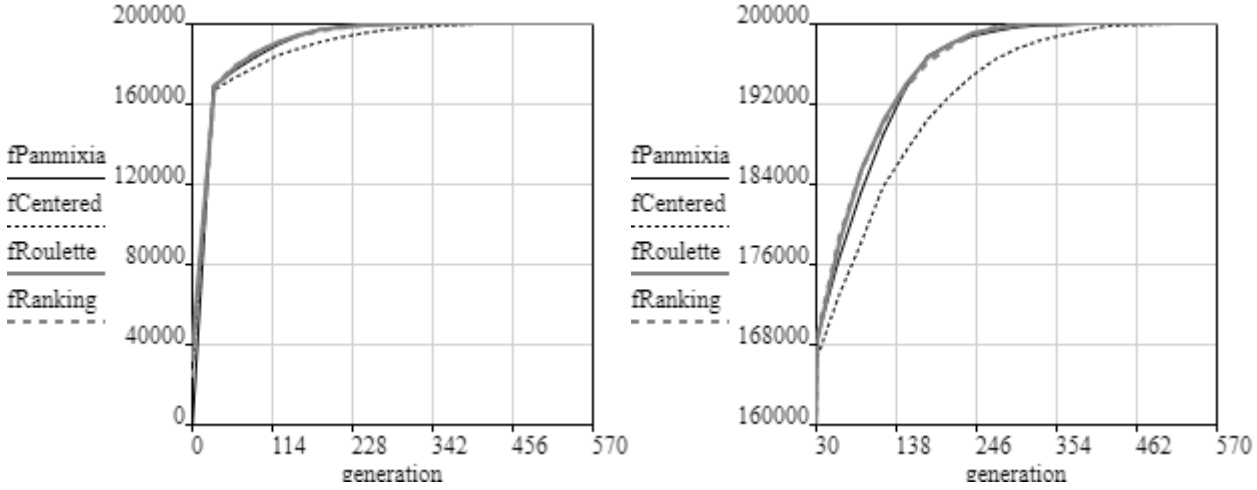

Рисунок 5.13 – Эволюционные траектории у операторов выбора родителей (решение простой задачи упаковки рюкзака)

Сравнив эволюционные траектории на рисунках 5.10 и 5.13, мы увидим, что жесткие операторы выбора родителей лучше справляются с поставленной задачей, если она является простой.

Таким образом, предлагается стратегия выбора оператора выбора родителей на основе сложности решаемой задачи. В случае сложной задачи необходимо использовать мягкие операторы выбора родителей, которые не отдают предпочтение не только лучшим индивидам популяции. Несмотря на то, что они временно проигрывают в скорости поиска, в перспективе они

приводят алгоритм к более точному решению. В случае простой задачи, где любое улучшение ведет к глобально оптимальному значению, выгоднее использовать жесткий оператор выбора родителей, который отдает предпочтение лучшим индивидам популяции.

5.5 Приближение эволюционной траектории математической моделью

При детальном рассмотрении полученных ранее эволюционных траекторий, можно сделать вывод о том, что форма эволюционной траектории похожа и на сигмоиду, и на гиперболу. Действительно, эволюционные траектории на рисунках из раздела 5.2 напоминают сигмоиду, в то время как остальные эволюционные траектории похожи на гиперболу. За основу разрабатываемой модели была взята логистическая функция (5.3). В результате совмещения сигмоиды и гиперболы была разработана модельная функция (5.4), где x – номер поколения, а $model\ \boldsymbol{x}$ – значение степени приспособленности (далее имеется в виду степень приспособленности наилучшего индивида) в данном поколении.

$$logistic\ \boldsymbol{x} = \frac{1}{1+e^{-x}} \tag{5.3}$$

$$model\ \boldsymbol{x} = \frac{a}{1+x^{c}b} + d \tag{5.4}$$

В данной работе предлагается гипотеза, что эволюция степени приспособленности происходит по закону (5.4). В таком случае возникает возможность быстрого прогнозирования хода работы генетического алгоритма по его первым результатам. Это позволит существенно ускорить процесс поиска подходящей конфигурации, так как время моделирования генетического алгоритма можно будет сильно сократить. Получение конкретной модели заключается в определении констант a, b, c и d, для этого необходимо выбрать 4 точки на эволюционной траектории. Очевидно, что нахождение данных констант в общем случае связано с решением трансцендентного уравнения, которое возникает из-за наличия $\boldsymbol{x}^{c}$. Предполагается, что численное решение такого уравнения не уступает по сложности моделированию настоящего

генетического алгоритма, а потому ценность данной модели будет заключаться в отсутствии необходимости использования численных методов.

Рассмотрим систему уравнений (5.5)-(5.8), возникающую при попытке нахождении констант в общем случае. Разработаем «удобный» частный случай и обойдем ограничения, связанные с трансцендентностью.

$$y_1 = \frac{a}{1 + x_1{}^c b} + d \tag{5.5}$$

$$y_2 = \frac{a}{1 + x_2{}^c b} + d \tag{5.6}$$

$$y_3 = \frac{a}{1 + x_3{}^c b} + d \tag{5.7}$$

$$y_4 = \frac{a}{1 + x_4{}^c b} + d \tag{5.8}$$

Логика применения данной модели предполагает $x_1 = 0$ и позволяет существенно сократить сложность (5.5). Игнорируем случай $c = 0$ и предполагаем справедливость замены (5.5) уравнением (5.9) при $x_1 = 0$.

$$y_1 = a + d \tag{5.9}$$

Похожим способом упростим также (5.6), приняв $x_2 = 1$. Можно видеть, что такой вариант замены не позволяет пользователю выбирать номер поколения, что может негативно сказаться на качестве модели в случае неудачно собранных данных о работе алгоритма. Решим возникшую проблему заменой представления $y_n = f(x_n)$ на $y_n = f(h * x_n)$, где h является некоторой целочисленной константой. В таком случае система уравнений примет вид (5.10).

$$y_n = \frac{a}{1 + (h * x_n)^c b} + d \tag{5.10}$$

Можно видеть, что $(h * x_n)^c b = h^c x_n{}^c b$, где h^c константа. В дальнейшем при использовании полученной модели учтем значение h как масштаб на Ox в виде $model(x_h)$ и примем замену $(h * x_n)^c b$ исходным $x_1{}^c b$. Таким образом, мы сохранили некоторую свободу действий пользователя без существенных усложнений разрабатываемого способа решения.

Аналогично проведенным ранее действиям заменим $x_3 = 2$ и в итоге получим систему уравнений (5.11)-(5.14).

$$y_1 = a + d \tag{5.11}$$

$$y_2 = \frac{a}{1+b} + d \tag{5.12}$$

$$y_3 = \frac{a}{1+2^c b} + d \tag{5.13}$$

$$y_4 = \frac{a}{1+x_4{}^c b} + d \tag{5.14}$$

Представим искомое значение в виде уравнений (5.15)-(5.18).

$$a = f_a(y_1, d) \tag{5.15}$$

$$b = f_b(y_2, a, d) \tag{5.16}$$

$$c = f_c(y_3, a, b, d) \tag{5.17}$$

$$d = f_d(y_1, y_2, y_3, y_4) \tag{5.18}$$

Заменим (5.17) уравнением (5.19), тем самым сохранив алгебраическую форму имеющихся значений.

$$2^c = f_{2c}(y_3, a, b, d) \tag{5.19}$$

В условиях замены (5.19) логичным является принятие $x_4 = 4$, так как $4^c = (2^c)^2$, что позволит представить все решение в алгебраическом виде. Таким образом, мы получили такую систему уравнений (5.20)-(5.23), которая позволяет без существенных затрат вычислительной мощности вычислить модель генетического алгоритма.

$$y_1 = a + d \tag{5.20}$$

$$y_2 = \frac{a}{1+b} + d \tag{5.21}$$

$$y_3 = \frac{a}{1+2^c b} + d \tag{5.22}$$

$$y_4 = \frac{a}{1+4^c b} + d \tag{5.23}$$

Можно видеть, что данная система проще исходной (5.5)-(5.8) и лучше поддается обработке. Выполнив ряд преобразований над полученной системой

уравнений, мы сможем найти вид функций f_a, f_b, f_c и f_d, которые представлены как (5.24)-(5.27).

$$f_a\ y_1, d\ = y_1 - d \tag{5.24}$$

$$f_b\ y_2, a, d\ = \frac{a}{y_2 - d} - 1 \tag{5.25}$$

$$f_c\ y_3, a, b, d\ = \log_2 \frac{a + d - y_3}{y_3 - d\ * b} \tag{5.26}$$

$$f_d\ y_1, y_2, y_3, y_4\ = \begin{matrix} \min(d_1, d_2), & y_1 > y_4 \\ \max(d_1, d_2), & y_1 < y_4 \end{matrix} \tag{5.27}$$

$$d_{1,2} = \frac{\pm\ \overline{\beta^2 - 4\alpha\gamma} - \beta}{2\alpha}$$

$$\alpha = {y_3}^2 + y_1 y_2 - 2y_1 y_3 + y_1 y_4 - y_2 y_4$$

$$\beta = 2y_2 y_3 y_4 - {y_1}^2 y_2 + 2{y_1}^2 y_3 - y_2 {y_3}^2 - {y_1}^2 y_4 - {y_3}^2 y_4$$

$$\gamma = y_1 y_2 {y_3}^2 - {y_1}^2 {y_3}^2 + {y_1}^2 y_2 y_4 + y_1 {y_3}^2 y_4 - 2y_1 y_2 y_3 y_4$$

Рассмотрим работу модели над ранее полученными эволюционными траекториями из рисунка 5.2, изображенную ниже на рисунке 5.14.

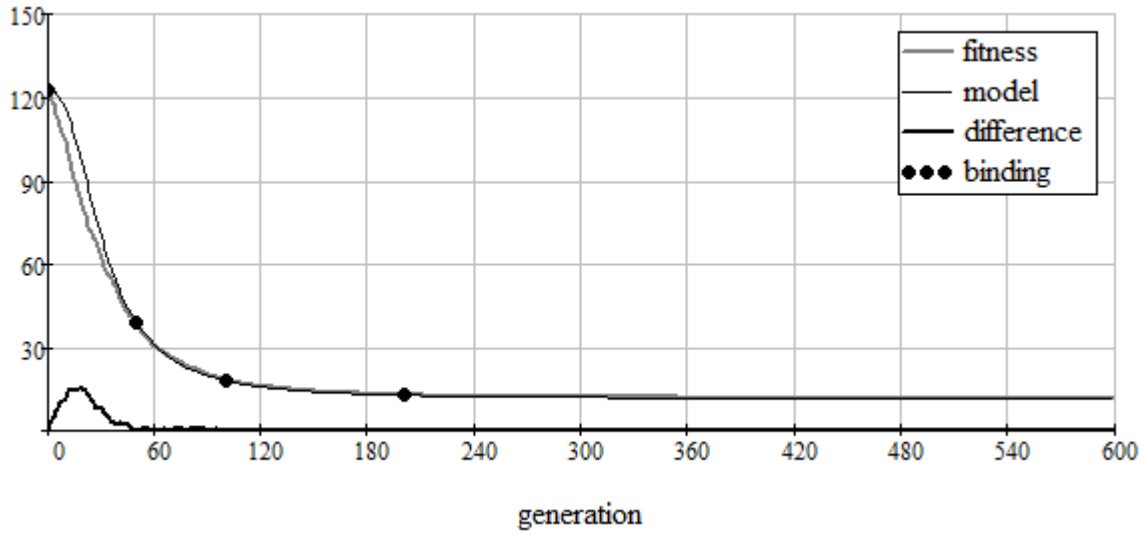

Рисунок 5.14 – Аппроксимация эволюционной траектории (конфигурация с малой популяцией)

Смоделированное значение показано тонкой черной линией, реальная выборка – жирной серой линией, разница между моделью и реальным

измерением – жирной черной линией (около Ox), кругами показаны точки для расчета коэффициентов по формулам (5.24)-(5.27). Можно видеть, что высказанная ранее гипотеза о том, что формирование эволюционной траектории можно смоделировать при помощи (5.4) подтверждается на практике. Далее на рисунках 5.15 и 5.16 изображена аппроксимация других вариантов эволюционных траекторий.

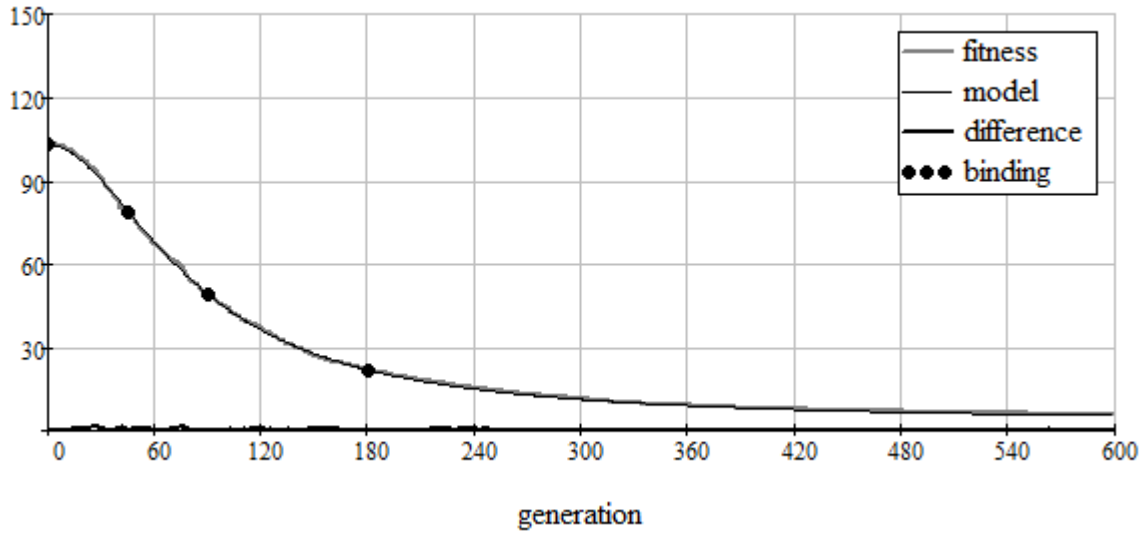

Рисунок 5.15 – Аппроксимация эволюционной траектории (конфигурация со средней популяцией)

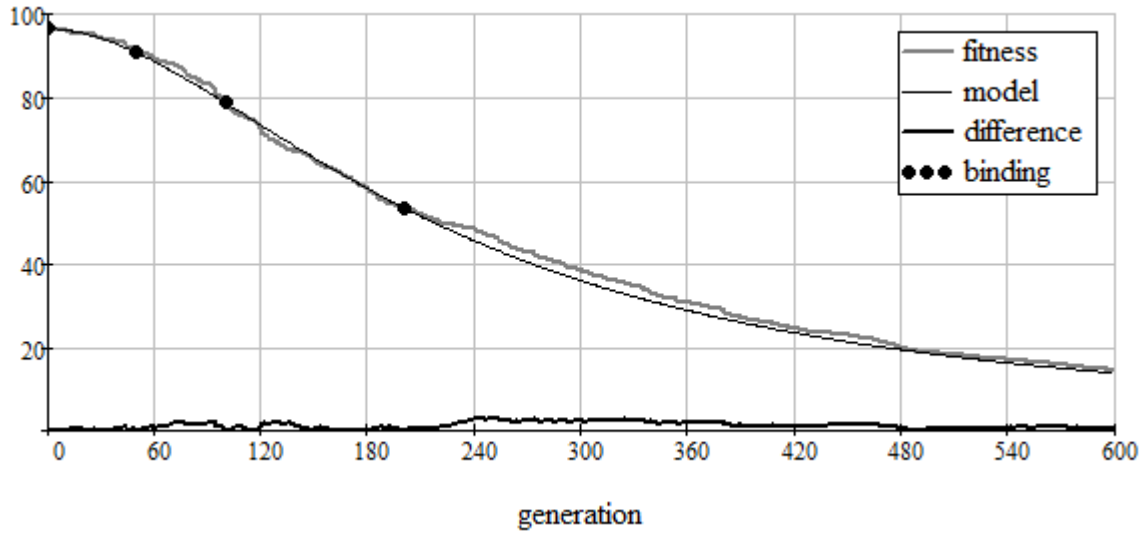

Рисунок 5.16 – Аппроксимация эволюционной траектории (конфигурация с большой популяцией)

Можно видеть, насколько сильно форма моделируемой величины совпадает с измеренной эволюционной траекторией при условии равенства всего лишь 4 точек из нескольких сотен имеющихся вокруг (особенно хорошо это видно на рисунке 5.15). Более того, можно видеть, как полученная модель способна прогнозировать (с некоторыми погрешностями) дальнейшую работу генетического алгоритма всего лишь по первым значениям эволюционной траектории.

На рисунке 5.16 можно видеть некоторые различия между аппроксимированным и реальным значением. Данная разница возникает при попытке получить аппроксимацию на основе данных малой выборки. Несмотря на то, что число пусков у алгоритмов на рисунках 5.15 и 5.16 одинаковое, больший размер популяции приводит к большему разбросу относительно гладкой траектории даже у осредненных данных. В таких случаях предлагается увеличить число тестовых запусков, которое позволит уменьшить влияние выбросов на математическое ожидание эволюционной траектории. Похожая ситуация также возникает в начале работы алгоритма, изображенного на рисунке 5.14. Данная конфигурация использует популяцию малого размера, скорость развития которой существенно выше. Из-за большой вероятности возникновения выбросов на данном участке возникает различие между реальным измерением и прогнозируемым значением.

Свойства, которыми обладает данная модель, также позволяют определить качество искомого решения, сгенерированного алгоритмом спустя много времени. Горизонтальная асимптота, к которой стремится правая ветвь моделируемой кривой, находится на уровне d. Практический смысл данного показателя заключается в том, что это наилучшее значение, которого способен достичь данный алгоритм. В случае, когда степень приспособленности необходимо уменьшать, необходимо подбирать такие конфигурации генетического алгоритма, у которых асимптота модели d имеет как можно меньшее значение. Также данная модель соблюдает полученное ранее свойство

эволюционных траекторий: медленный прогресс приводит к более точному ответу.

На рисунке 5.17 изображен неудачный (относительно ранее показанного на рисунке 5.15) вариант прогноза эволюционной траектории. Это связано с тем, что участок эволюционной траектории, на котором основывается прогноз, имеет множество выбросов. Начало работы генетического алгоритма имеет множество вариантов развития, а потому эмпирически полученное математическое ожидание левой части эволюционной траектории может не содержать адекватных показателей, пригодных для построения качественного прогноза.

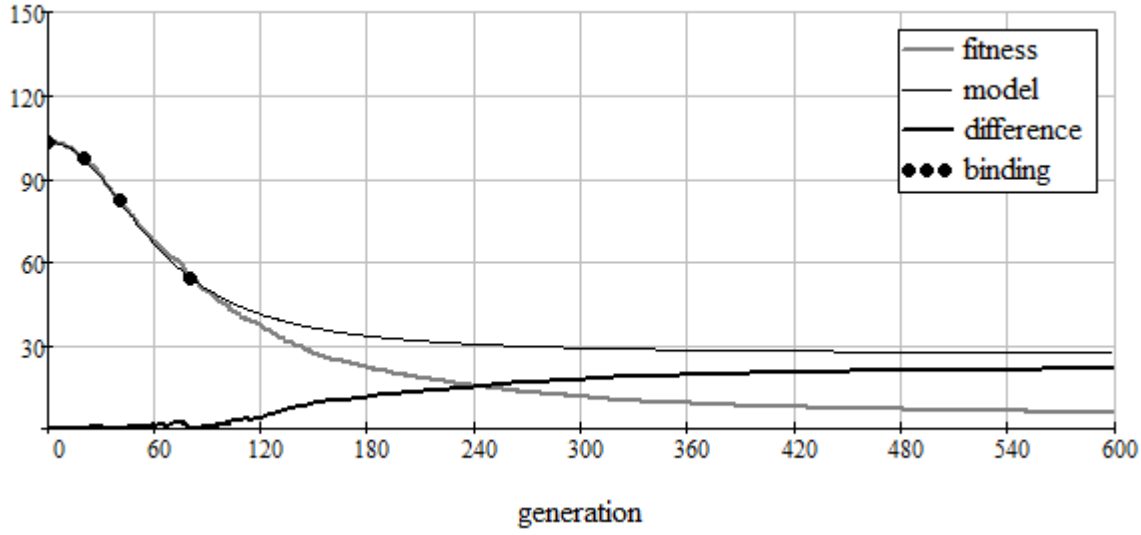

Рисунок 5.17 – Аппроксимация эволюционной траектории
(неудачный вариант прогноза конфигурации со средней популяцией)

Таким образом, предлагается вариант ускоренного поиска качественных эволюционных траекторий, основанный на замене большего участка эволюционной траектории прогнозируемой величиной. Несмотря на то, что данный метод может оказаться успешным, гарантия построения качественного прогноза отсутствует. Это связано одновременно с характером работы генетического алгоритма, у которого начало работы содержит существенные изменения состояния популяции, и подходом к построению конкретной

модели, основанным на использовании как раз начального (неустойчивого, нестабильного) участка эволюционной траектории.

5.6 Адаптивные генетические операторы

Особенности работы генетического алгоритма, рассмотренные в разделах 5.3 и 5.4, показывают существование обратной зависимости между скоростью нахождения решения и его точностью. Закономерным является желание совместить положительные характеристики различных генетических операторов и получить такой оператор, который найдет точное решение быстро. Одним из вариантов реализации такого подхода предлагается применение адаптивных генетических алгоритмов.

В состав адаптивного генетического алгоритма входят адаптивные генетические операторы. Основное отличие классического генетического алгоритма от адаптивного заключается в том, что конфигурация классического алгоритма остается постоянной все время работы, когда как конфигурация адаптивного алгоритма может меняться. В данной работе предлагается представление адаптивного генетического алгоритма в виде совокупности начальной конфигурации и множества событий, изменяющих конфигурацию при возникновении некоторых условий. Так классический генетический алгоритм становится частным случаем адаптивного алгоритма, у которого множество событий оставлено пустым.

Событие генетического алгоритма состоит из условия вызова и ответной реакции на данное условие. В рамках данной работы предлагается реакция на слишком слабый прогресс поиска, исполненная следующим образом: если степень приспособленности f лучшего индивида популяции не изменилась на d за n поколений, то событие возникнет. В качестве ответной реакции происходит замена имеющегося генетического оператора новым, обозначенном в событии [29,30].

В разделе 5.3 при решении задачи поиска минимума на функции сферы возникла ситуация, когда оператор мутации большой мощности быстро

приводил алгоритм к окрестности оптимального решения, после чего резко терял свою эффективность (рисунок 5.5). При этом оператор мутации малой мощности существенно проигрывал в скорости нахождения более точного решения (рисунки 5.4 и 5.5). Рассмотрим попытку борьбы с данным явлением при помощи адаптивного генетического алгоритма, который определит неприемлемо низкую эффективность текущей конфигурации и заменит ее более подходящей для полученной популяции. На рисунке 5.18 изображена работа классических и адаптивных генетических алгоритмов над задачей поиска минимума функции сферы.

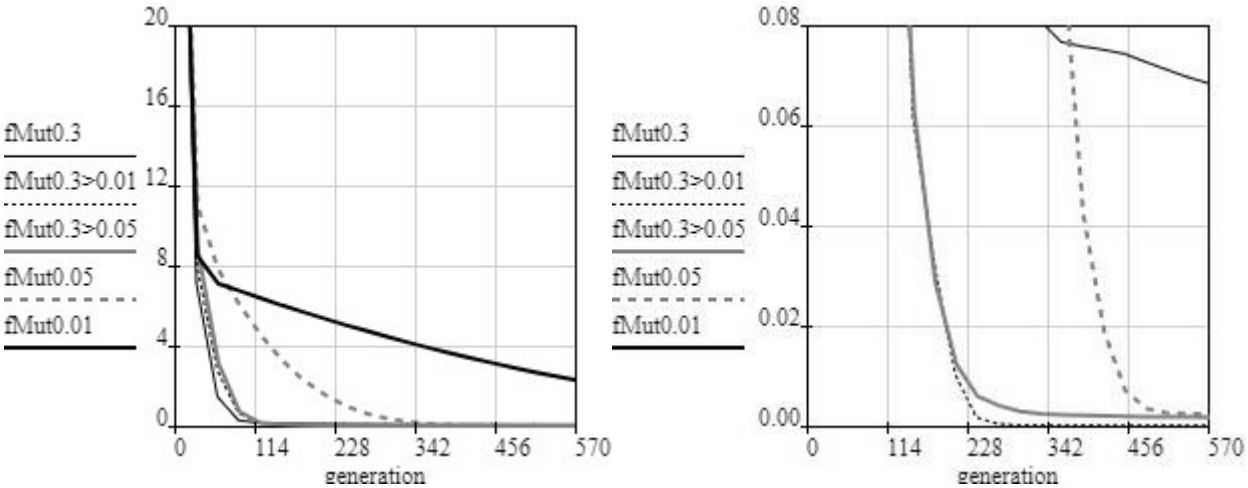

Рисунок 5.18 – Эволюционные траектории у мутаций разной мощности (решение задачи поиска минимума на функции сферы)

Эволюционные траектории fMut0.3, fMut0.05 и fMut0.01 описывают работу классического генетического алгоритма. Эволюционные траектории fMut0.3>0.01 и fMut0.05 принадлежат адаптивным генетическим алгоритмам, которые ослабили мощность мутации при обнаруженном падении эффективности. Можно видеть, что своевременная замена оператора мутации на более подходящий вариант существенно улучшает характеристики генетического алгоритма. Так нам удалось получить одновременно быстрый и точный поиск, недоступный при использовании классического генетического алгоритма.

С другой стороны, данный подход оказался менее успешным при попытке заменить оператор выбора родителей. Рассмотрим работу адаптивного генетического алгоритма над задачей поиска минимума функции Растригина. Ранее на рисунке 5.9 мы заметили, как жесткие операторы выбора родителей имели временное преимущество, проигрывая в перспективе мягким операторам выбора родителей. На рисунке 5.19 изображена попытка заменить жесткий оператор выбора родителей на мягкий оператор.

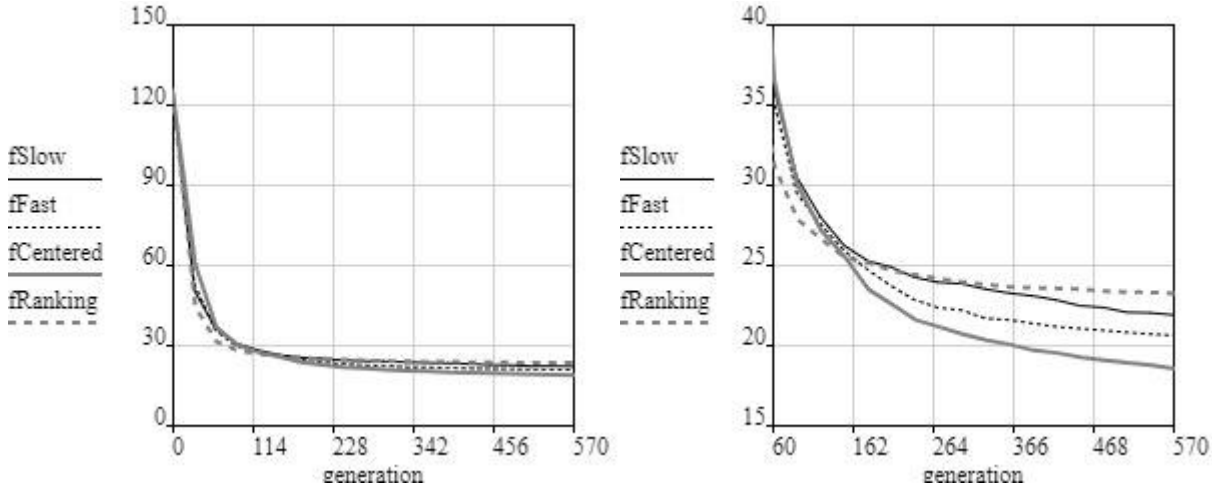

Рисунок 5.19 – Эволюционные траектории у операторов выбора родителей (решение задачи поиска минимума на функции Растригина)

Эволюционные траектории fCentered и fRanking соответствуют классическим генетическим алгоритмам, тогда как fSlow и fFast – адаптивным версиям. Конфигурация fFast заменяет метод ранжирования центральным выбором раньше, чем fSlow. При этом можно видеть, что чем дольше использовался центральный выбор, тем точнее решение задачи. Так адаптивный генетический алгоритм не смог объединить положительных качеств двух разных операторов выбора родителей.

Таким образом, подход по улучшению конфигурации генетического алгоритма при помощи заменяемых элементов имеет переменный успех и является перспективным направлением для дальнейшего изучения. Полученные результаты показывают перспективность данного направления, однако необходимы дополнительные исследования, которые позволят разработать

подходящие для замены генетические операторы, а также помогут выявить условия, при возникновении которые следует использовать замену.

5.7 Рекомендации по разработке генетических алгоритмов

Исследование, построенное на сборе и анализе эволюционных траекторий, позволило обнаружить опорные явления, на которые можно ориентироваться при конфигурировании генетического алгоритма. Так выявленная разница в работе над простыми и сложными задачами позволила сформировать две противоположные стратегии построения генетических алгоритмов для простых и сложных задач.

Для решения простой задачи подходит генетический алгоритм с малой популяцией. Это позволяет повысить скорость поиска без существенной потери точности. Также для решения простой задачи необходим жесткий оператор выбора родителей (например, метод рулетки или метод ранжирования). Данная конфигурация работает быстрее по сравнению с мягкими операторами выбора родителей без существенных потерь в точности. Кроме того, решение простой задачи может быть получено существенно быстрее путем использования мощной мутации, но следует обратить внимание на точность получаемого решения: она обратно пропорциональна мощности мутации.

Для решения сложной задачи подходит генетический алгоритм с увеличенной популяцией. Точность решения изменяется прямо пропорционально размеру популяции, однако, скорость поиска изменятся обратно пропорционально размеру популяции, из-за чего следует искать баланс точности решения и скорости его нахождения. Также для решения сложной задачи необходим мягкий оператор выбора родителей (например, панмиксия или центральный выбор). Данная конфигурация позволяет получать более точные решения по сравнению с жесткими операторами выбора родителей, хотя на начальном этапе уступает жестким операторам. Кроме того, решение сложной задачи может будет получено при использовании оператора мутации

малой мощности. При этом необходимо помнить, что слишком слабая мутация ослабит поисковую мощность алгоритма и неприемлемо замедлит поиск.

Обнаруженное явление обратной зависимости между скоростью нахождения решения и его точностью в некоторых случаях может быть подавлено адаптивными генетическими операторами. Так в случае решения простой задачи удалось добиться быстрой генерации точного ответа при помощи замены мощного оператора мутации на слабый оператор. Генетический алгоритм самостоятельно определил момент, когда популяция стала готовой к обработке новой конфигурацией, после чего применил новый оператор мутации и довел промежуточное неточное решение до необходимого качества. Стоит заметить, что применение адаптивного генетического алгоритма не всегда позволяет объединить положительные свойства нескольких генетических операторов: так в случае с оператором выбора родителей не удалось достичь существенных улучшений.

В ходе анализа эволюционных траекторий различной формы был выявлен эвристический закон, по которому предположительно происходит развитие эволюционной траектории во времени. Аппроксимация готовых эволюционных траекторий при помощи предлагаемой функции показывает схожесть смоделированных и реальных результатов. Предлагается прогнозировать дальнейшее поведение генетического алгоритма по первоначальному этапу его работы путем построения эвристической аппроксимации. В таком случае долговременная перспектива работы генетического алгоритма может быть определена без долгой работы генетического алгоритма. С другой стороны, начальный этап работы генетического алгоритма имеет большую дисперсию, а потому для получения математического ожидания необходимо множество коротких запусков алгоритма. Требуются дополнительные исследования относительно эффективности применения полученной модели по сравнению с реальным запуском генетического алгоритма.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было проведено исследование способов и особенностей получения производительных конфигураций генетического алгоритма, основанное на анализе эволюционных траекторий. Изучение классического генетического алгоритма и составление математических моделей генетических операторов позволило выявить трудности, связанные с поисковым характером составления качественной конфигурации. С целью автоматизации процесса подбора генетических операторов был реализован генетический алгоритм с каркасной архитектурой, способный также собирать данные о динамике состояния популяции.

Процесс испытания конфигурации генетического алгоритма включает в себя реализацию генетического алгоритма при данной конфигурации, его запуск и одновременный сбор данных о ходе работы алгоритма. Стохастическая природа генетического алгоритма усложняет подобные испытания тем, что пусков необходимо делать много, а собранные данные усреднять. Для упрощения такого способа исследования разработана и описана система испытания генетических алгоритмов, принимающая в качестве входных данных конфигурации генетического алгоритма и форматы сводных таблиц, а выдающая в качестве результата сводные таблицы с данными о ходе работы алгоритма при некоторой конфигурации. Реализована возможность импорта в виде таблиц и в виде графиков.

Решение основной проблемы, связанной с высокой сложностью разработки генетических алгоритмов из-за поискового характера данного процесса, представлено в виде правил по настройке генетического алгоритма, основанных на сложности решаемой задачи: разработаны две стратегии по выбору размера популяции, оператора выбора родителей и мощности оператора мутации для простых и сложных задач. Обозначена обратная зависимость между точностью найденного решения и скоростью его получения. Описан способ борьбы с данной зависимостью при помощи адаптивного оператора

мутации, применение которого позволило быстро получить точное решение простой задачи. Разработана функция, способная аппроксимировать эволюционную траекторию показателя наилучшей степени приспособленности в поколении. Также предоставлены результаты попыток прогнозирования формы эволюционной траектории при помощи данной функции. Полученные результаты позволят сократить время поиска качественной конфигурации за счет уменьшения как числа пробуемых конфигураций (применение правил выбора генетических операторов), так и за счет сокращения времени моделирования генетического алгоритма (применение формы аппроксимирующей функции вместо реальной работы алгоритма).

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Архипов, И.В. Применение генетического алгоритма для многокритериальной задачи календарного планирования // Научно-технический вестник информационных технологий, механики и оптики, Т. 15, № 3, 2015. – С. 525–531.

2. Гончаров, Е.Н. Генетический алгоритм для задачи календарного планирования с ограниченными ресурсами // Автомат. и телемех., № 6, 2017. – С. 173–189.

3. Гончаров, Е.Н. Стохастический жадный алгоритм для задачи календарного планирования с ограниченными ресурсами // Дискретный анализ и исследование операций, Т. 21, № 3, 2014. – С. 11–24.

4. Губайдуллин, И. М. Применение разрывного метода Галеркина для решения обратной задачи диффузии лекарственных веществ из хитозановых пленок // Журнал Средневолжского математического общества., Т. 18, № 2, 2016. — С. 94–105.

5. Дивеев, А.И. Вариационный генетический алгоритм для решения задачи оптимального управления // Современные проблемы науки и образования, № 1, 2014. http://www.science-education.ru/ru/article/view?id=11474

6. Дивеев, А.И. Метод бинарного генетического программирования для поиска математического выражения // Вестник Российского университета дружбы народов. Серия: Инженерные исследования, Т. 18, № 1, 2017. – С. 125—134.

7. Дунаев, М.П. Параметрическая оптимизация системы управления насосной станцией с помощью генетического алгоритма // Наука и Образование: Научное издание, № 8, 2014. – С. 194–205.

8. Евсютин, О.О. Алгоритм встраивания информации в сжатые цифровые изображения на основе операции замены с применением оптимизации // Компьютерная оптика, Т. 41, № 3, 2017. – С. 412–421.

9. Ершов, Н.М. Неоднородные клеточные генетические алгоритмы // Компьютерные исследования и моделирование, Т. 7, № 3, 2015. – С. 775-780.
10. Жукович, Я.П. Генетические алгоритмы // Образование и наука в современных условиях: материалы IX Междунар. науч.–практ. конф., № 4, 2016. – С. 145-149.
11. Караев, А.Д. Применение генетического алгоритма для имитации искусственного интеллекта в игре // Студенческая наука: современные реалии : материалы II Междунар. студенч. науч.-практ. конф., 2017. – С. 53–60.
12. Коваленко, Ю.В. О сложности оптимальной рекомбинации для задач составления расписаний в многостадийной системе поточного типа // Дискретн. анализ и исслед. опер., № 2, 2016. – С. 41–62
13. Королёва, А.С. Генетический алгоритм решения задачи о ранце // Студенческая наука XXI века : материалы VI Междунар. студенч. науч.–практ. конф., № 3, 2015. – С. 83–84.
14. Кошев, А.Н. Разработка генетического алгоритма с адаптивными мутациями для определения глобального экстремума функции n-переменных // Интернет-журнал «НАУКОВЕДЕНИЕ», Т. 8, № 6, 2016. http://naukovedenie.ru/PDF/32TVN616.pdf
15. Матвиенко, Г.Г. Моделирование переноса излучения методом Монте-Карло и решение обратной задачи на основе генетического алгоритма по результатам эксперимента зондирования аэрозолей на коротких трассах с использованием фемтосекундного лазерного источника // Квантовая электроника, № 2, 2015. – С. 145–152.
16. Урбан, А.Р. Решение задачи поиска оптимального столбца в условиях оптимального раскроя бумажного полотна // Вестник СПбГУ. Серия 10. Прикладная математика. Информатика. Процессы управления, № 1, 2015. – С. 100–106.
17. Шрейдер, М.Ю. Проектирование систем физической защиты с помощью генетического алгоритма // Интернет-журнал «НАУКОВЕДЕНИЕ», Т. 9, № 4, 2017. http://naukovedenie.ru/PDF/94TVN417.pdf

18. Шумков, Е.А. Использование генетических алгоритмов для обучения нейронных сетей // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета, № 91, 2013. – С. 455–464.

19. Ahmed F.A., Mohamed A.T., A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems // Ain Shams Engineering Journal. 2016. http://www.sciencedirect.com/science/article/pii/S2090447916301101

20. Aibinua A.M., Bello Salaub H., Najeeb Arthur R., Nwohud M.N., Akachukwue C.M., A novel Clustering based Genetic Algorithm for route optimization // Engineering Science and Technology, an International Journal. 2016. http://www.sciencedirect.com/science/article/pii/S2215098616300738

21. Divo D.S., Consorcia E.R., Felino P.L., Rolando G.P., Nathaniel C.B., Using Genetic Algorithm Neural Network on Near Infrared Spectral Data for Ripeness Grading of Oil Palm (Elaeis guineensis Jacq.) Fresh Fruit // Information Processing in Agriculture. 2016. http://www.sciencedirect.com/science/article/pii/S2214317316300427

22. Esraa A. Abdelhalim, Ghada A. El Khayat, A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA) // Alexandria Engineering Journal. Vol. 55(2). 2016. P. 1395–1409.

23. Marcos V.G., Juan P.C., M. Angeles L.C., Analyzing the determinants of the voting behavior using a genetic algorithm // European Research on Management and Business Economics. Vol. 22(3). 2016. P. 162–166.

24. Maryam H., Fatemeh S., Exergy analysis and optimization of a high temperature proton exchange membrane fuel cell using genetic algorithm // Case Studies in Thermal Engineering. 2016. Vol. 8. P. 207–217.

25. Mohamed A.Y., Mohamed T.L., The path planning of cleaner robot for coverage region using Genetic Algorithms // Journal of Innovation in Digital Ecosystems. Vol. 3(1). 2016. P. 37–43.

26. Mohammad G., Ghasem Z., Hooshang J.R., Prediction of asphaltene precipitation using support vector regression tuned with genetic algorithms // Petroleum. Vol. 2(3). 2016. P. 301–306.

27. Nikolaidis A., Low overhead reversible data hiding for color JPEG images // Multimedia Tools and Applications. 2016. Vol. 75(4). P. 1869–1881.

28. Salari N., Shohaimi S., Najafi F., A Novel Hybrid Classification Model of Genetic Algorithms, Modified k-Nearest Neighbor and Developed Backpropagation Neural Network // National Center of Biotechnology Information. 2014. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4242540

29. Ying L., Haibo D., Niels L., Sanja P., A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance // International Journal of Production Economics. Vol. 179. P. 259–272.

30. Younes A., Barzegar G., Pishvaiec M.R., Moradid R., Monfaredc M.S., Optimal control of batch cooling crystallizers by using genetic algorithm // Case Studies in Thermal Engineering. Vol. 8. 2016. P. 300–310.