Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Утверждаю: Зав.каф. ПМиИ, профессор

_____ (Тимченко С. В.)

Тимченко С. В., Мещеряков П. С.

ПРИКЛАДНАЯ ИНФОРМАТИКА

Методические указания по курсовому проекту
для бакалавров по направлению 210100 «Электроника и наноэлектроника»
и студентов по специальности 210106 «Промышленная электроника»

Томск 2012

Оглавление

ВВЕДЕНИЕ

В последние годы при решении задач функциональной оптимизации стали широко применяться адаптивные методы поиска, среди которых особую популярность завоевали эволюционные и, в том числе, генетические алгоритмы (ГА). Они основаны на генетических процессах биологических организмов: биологические популяции развиваются в течение нескольких поколений, подчиняясь законам естественного отбора и принципу «выживает сильнейший» (точнее, наиболее приспособленный). Предложенные в конце 60-ых годов и получившие теоретическое обоснование в 1975 году (Holland, «Adaptation in Natural and Artifical Systems») генетические алгоритмы (в силу своей универсальности и высокой эффективности) в дальнейшем получили чрезвычайно широкое распространение для решения задач поиска и оптимизации в самых разнообразных областях науки и техники.

Природа часто поражает как своей сложностью, так и богатством всех своих проявлений. Среди примеров можно назвать всевозможные, как правило, очень сложные социальные и биологические системы. Многое из того, что мы наблюдаем, можно объяснить единой теорией эволюции через наследственность, изменчивость и отбор. Начиная с работы Чарльза Дарвина «Происхождение видов», опубликованной в 1859 году, теория эволюции оказала существенное влияние на мировоззрение людей. При этом, безусловно, ее влияние существенно сказалось не только в биологии, но и во многих других областях научного знания. Поэтому неудивительно, что ученые, занимающиеся компьютерным моделированием, воспользовались идеями, заложенными в теории эволюции.

Таким образом, генетические алгоритмы – это вдохновленные эволюцией вычислительные модели, которые представляют собой семейство алгоритмов поиска, основанных на механизмах природной селекции и генетики, и имеющих дело с набором (популяцией) возможных решений задачи (испытаний), которые подвергаются циклическому воздействию трех основных «генетических» операторов - селекции, скрещивания и мутации. Суще-

ственная особенность этих алгоритмов заключается в кодировании потенциального решения некоторой проблемы в виде простой хромосомо-подобной структуры данных. К набору таких хромосом собственно и применяют операторы рекомбинации с тем, чтобы сохранить критическую информацию.

Работа генетического алгоритма начинается с генерирования некоторой (как правило, случайной) популяции хромосом, каждой из которых соответствует точка в пространстве поиска. Затем происходит оценка этих структур (расчет целевой функции для каждой хромосомы) и распределение между ними репродуктивных возможностей таким образом, чтобы хромосомы, представляющие лучшее решение исходной проблемы, получили больше шансов для «воспроизводства» чем хромосомы, соответствующие худшим решениям. Качество решения обычно определяется по отношению к текущей популяции.

Будучи вероятностными, генетические алгоритмы тем не менее не являются просто еще одним вариантом случайного поиска, поскольку при отборе новых точек с ожидаемыми более хорошими характеристиками, они эффективно используют предыдущую информацию.

Важной особенностью генетических алгоритмов является то, что они обеспечивают сходимость к глобальному оптимуму (что очень важно для задач, у которых целевая функция имеет несколько локальных экстремумов). В отличие от классических градиентных методов оптимизации, при реализации генетических алгоритмов не требуется ставить ограничения на гладкость целевой функции. Более того, они позволяют находить оптимум даже в случае разрывной целевой функции.

Хотя генетические алгоритмы обычно рассматриваются как функциональные оптимизаторы, область проблем, в которой они могут быть применены, весьма широка.

ВВЕДЕНИЕ В ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Понятие оптимизации

Пусть у нас есть некоторый определенный тип или класс объектов (т.е. множество объектов, удовлетворяющих некоторому набору условий). И пусть нам необходимо найти в этом классе некоторый объект, удовлетворяющий другому некоторому условию. Такой процесс можно обобщенно назвать поиском или решением. Класс, среди объектов которого производится поиск, назовем областью поиска или пространством поиска. Искомый объект можно назвать целью или целевым объектом поиска, а условие, которому он должен удовлетворять – целевым условием. Для определения условия обычно задается некоторая функция на пространстве поиска. Достижение функцией определенного значения и является целевым условием. Такая функция называется целевой функцией. Таким образом, поиск заключается в просмотре по определенным правилам пространства поиска всех объектов, пока не будет обнаружен целевой. Функции выбора нового кандидата для проверки называются операторами поиска. Оператор поиска определяет своего рода прыжок или шаг по пространству поиска.

Примером задачи оптимизации может служить поиск минимума функции $y = x^2$. Областью поиска является все пространство вещественных чисел, целевым условием – минимум функции.

Естественная эволюция

Как известно, у Природы есть свой метод создания лучших организмов. Дарвин назвал его Эволюцией вследствие Естественного отбора. Эволюция подразумевает под собой последовательное развитие организмов – непрерывную последовательность родителей и их детей, когда дети многое наследуют от своих родителей, но кое в чем от них отличаются. Естественный отбор – это непрерывное сражение за жизнь между всеми. "Выживает сильнейший" – вот жизненное кредо Природы, если награж-

дать титулом "сильный" самого подходящего, самого приспособленного для жизни.

Если подходить к описанию эволюции более формально, то вначале необходимо отметить, что объектом развития (т.е. эволюции) являются не сами организмы, а виды в целом. Вид – это совокупность организмов, сходных по строению и другим признакам. Пользуясь терминологией объектно-ориентированного программирования, вид – это класс, а принадлежащие виду индивиды – объекты этого класса. Совокупность индивидов одного вида назовем популяцией. Чтобы эволюция вообще была возможна, организмы должны отвечать 4 важнейшим свойствам:

1. Каждый индивид в популяции способен к размножению.
2. Отличия индивидов друг от друга влияют на вероятность их выживания.
3. Каждый потомок наследует черты своего родителя (подобное происходит от подобного).
4. Ресурсы для поддержания жизнедеятельности и размножения ограничены, что порождает конкуренцию и борьбу за них.

Все процессы в живых организмах работают за счет сложных молекул – белков. Каждый белок представляет собой маленький биологический автомат. Молекула белка состоит из последовательности аминокислот. Совокупность информации и строение всех белков в организме определяет его изначальную структуру (развитие организма происходит также и под действием внешней среды). Вся эта информация называется генетической информацией, или генотипом. Процесс построения, развития организма по информации из генотипа называется онтогенезом. А строение, качества и свойства организма – фенотип. Т.к. внешняя среда воздействует на организм в целом, то можно сказать, что вероятность выживания организма определяется фенотипом.

Генетическая информация в клетке хранится в специальных молекулах – нуклеиновых кислотах. Нуклеиновая кислота представляет собой полимер, т.е. молекулу, представляющую собой последовательность из соединенных между собой небольших молекул – мономеров. Мономерами нуклеиновых кислот являются

нуклеотиды. Для кодирования информации используется 4 вида нуклеотидов, обозначаемых по названиям входящих в них азотистых оснований – A,T,G,C. Таким образом, алфавит кодировки состоит из 4 букв.

При сексуальном размножении потомку передается информация о строении родителей путем передачи ДНК. При этом, для построения ДНК потомка, родительские ДНК меняются своими участками. Это процесс называется скрещиванием (кросовер – crossover). При этом новый ген представляет собой комбинацию информации из родительских ДНК (рекомбинация наследственной информации). При размножении может произойти мутация ДНК, т.е. случайное изменение небольшой ее части.

Из этого небольшого обзора хорошо видно, что естественный процесс оптимизации является некоторым компромиссом между вариацией потомства (получением новых индивидов), обеспечением достаточного процента жизнеспособности и стремлением получить хорошее потомство, т.е. не хуже, или в большинстве случаев лишь чуть хуже предков.

Генетические алгоритмы

Для компьютерных алгоритмов решения (поиска), использующих вычислительные модели механизмов естественной эволюции в качестве ключевых структурных элементов, используется обобщенное название – эволюционные алгоритмы. Существует множество разновидностей подобного рода алгоритмов, отличающихся использованием или неиспользованием конкретных механизмов, а также различиями трактовки этих механизмов и представлением индивидов.

В генетическом алгоритме (ГА) каждый индивид кодируется сходным с ДНК методом – в виде строки из символов одного типа. Длина строки (ДНК) постоянна. Популяция из индивидов подвергается процессу эволюции с интенсивным использованием скрещивания и мутаций.

Назовем представление каждого индивида геномом. Для каждого вида и каждого представления для данного целевого условия задается целевая функция. Значение целевой функции

назовем целевым значением. Вектор, состоящий из целевых значений всех индивидов в популяции, назовем вектором целевых значений. Тогда если вычислен вектор целевых значений, то можно определить приспособленность (fitness) индивида в популяции, для чего задается специальная функция приспособленности от данного целевого значения и от вектора целевых значений. Аналогично вектору целевых значений введем вектор приспособленности. Мы отделяем приспособленность от целевого значения специально, т.к. приспособленность индивида зависит и от остальных индивидов, и важна для выживаемости индивида, а целевое значение важно в первую очередь для нас. Часто целевое значение называют приспособленностью, а значение приспособленности, в смысле вероятность участия в размножении, неявно вычисляется во время отбора. Процесс эволюции останавливается, когда популяция отвечает определенному критерию – критерию завершения.

Принципиальная схема работы ГА состоит из следующих основных фаз:

1. Создание начальной популяции. Задание генома каждому из индивидов. Расчет вектора целевых значений.
2. Шаг эволюции – построение нового поколения.
3. Проверка критерия завершения, если не выполнено – переход на 2.

Шаг эволюции можно разделить на следующие этапы:

- Вычисление вектора приспособленности.
- Отбор кандидатов на скрещивание (Отбор – Selection) .
- Скрещивание, т.е. порождение каждой парой отобранных кандидатов новых индивидов, путем геномов.
- Мутация геномов.
- Вычисление вектора целевых значений и построение новой популяции (нового поколения).

Целевая функция и кодирование

Среди компонентов, образующих генетический алгоритм, в большинстве случаев только два непосредственно определяются конкретной задачей – это кодировка задачи (отображение про-

странства поиска на пространство битовых строк) и целевая функция. Рассмотрим задачу параметрической оптимизации, заключающуюся в определении набора переменных, миниимизирующих некоторую величину (цель). В более традиционных терминах, задача состоит в поиске минимума некоторой функции $F(X_1, X_2, \ldots, X_M)$.

На первом этапе обычно делается предположение, что переменные, представляющие параметры, могут быть представлены битовыми строками. Это означает, что переменные предварительно дискретизируются некоторым образом и что область дискретных значений соответствует некоторой степени 2. Например, с 10 битами на параметр мы получаем область из $2^{10} = 1024$ дискретных значений. Если параметры непрерывны, то проблема дискретизации не заслуживает особого внимания. Разумеется, предполагается, что дискретизация обеспечивает достаточное разрешение, чтобы сделать возможным регулирование получения результата с желаемым уровнем точности. Предполагается также, что дискретизация в некотором смысле представляет основную функцию.

Битовую строку длины N можно рассматривать как целое двоичное число I, которому соответствует некоторое вещественное значение r из заданного диапазона $[Min, Max]$. Это соответствие устанавливается формулой

$$r = Min + (Max - Min)\frac{I}{2^N - 1}.$$

За исключением проблемы кодирования, целевая функция обычно дается как часть постановки задачи. С другой стороны, разработка целевой функции иногда может непосредственно входить в разработку алгоритма оптимизации или поиска решения. В других случаях значение целевой функции может зависеть от реализации и давать только приближенный или частный результат.

Общая структура генетического алгоритма

В природе особи в популяции конкурируют друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют за при-

влечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, будут иметь относительно больше шансов на воспроизводство потомков. Слабо приспособленные особи либо совсем не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространяться в увеличивающемся количестве потомков на каждом последующем поколении. Таким образом, вид развивается, все лучше приспосабливаясь к среде обитания.

Генетические алгоритмы используют прямую аналогию с таким механизмом. Они работают с совокупностью "особей" – популяцией, каждая из которых представляет возможное решение данной проблемы. Каждая особь оценивается мерой ее "приспособленности" согласно тому, насколько "хорошо" соответствующее ей решение задачи. Например, мерой приспособленности могло бы быть отношение силы/веса для данного проекта моста. (В природе это эквивалентно оценке того, насколько эффективен организм при конкуренции за ресурсы.) Наиболее приспособленные особи получают возможность "воспроизводить" потомство с помощью "перекрестного скрещивания" с другими особями популяции. Это приводит к появлению новых особей, которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Наименее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции.

Таким образом, воспроизводится вся новая популяция допустимых решений, выбирая лучших представителей предыдущего поколения, скрещивая их и получая множество новых особей. Это новое поколение содержит более высокое соотношение характеристик, которыми обладают хорошие члены предыдущего поколения. Таким образом, из поколения в поколение хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге популяция будет сходиться к оптимальному решению задачи.

Имеется много способов реализации идеи биологической эволюции в рамках генетических алгоритмов. Каноническим считается генетический алгоритм, представленный в виде следующего псевдокода.

begin {генетический алгоритм}

t := 0;

done := **false**;

init_population(*P* (0) **)**;

{генерация начальной популяции *P* (0) (случайным образом или случайным в комбинации с некоторым набором начальных решений) и оценка пригодности каждой особи}

while not *done* **do begin**

P' := select_parents**(** *P* (*t*));

{выбор родителей для скрещивания при помощи оператора селекции согласно пригодности и помещение их в промежуточную популяцию *P*'}

recombine *P*'(*t*);

{промежуточная популяция попарно подвергается оператору рекомбинации или скрещивания}

mutate *P*' (*t*);

{каждая особь в промежуточной популяции подвергается оператору мутации}

evaluate *P*' (*t*);

{расчет целевой функции для всех новых особей}

P(*t*+1) := *P*'(*t*);

{промежуточная популяция копируется в новое поколение *P*(*t*+1)}

Done := || *P*(*t*+1) - *P*(*t*+1)||<ε;

{проверка выполнения критерия сходимости}

t := *t* + 1;

end {while}

end

Хотя модель эволюционного развития, применяемая в генетических алгоритмах, сильно упрощена по сравнению со своим природным аналогом, тем не менее генетические алгоритмы являются достаточно мощным средством и могут с успехом применяться для широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно решить другими методами. Однако, генетические алгоритмы не гарантируют обнаружения глобального решения за конечное время. Генетические алгоритмы не гарантируют и того, что глобальное решение будет найдено (впрочем, для произвольной целевой функции за конечное время этого невозможно сделать ни одним алгоритмом), но они хороши для поиска "достаточно хорошего" решения задачи "достаточно быстро". Главным же преимуществом генетических алгоритмов является то, что они могут применяться даже на сложных задачах, там, где не существует никаких специальных методов. Даже там, где хорошо работают существующие методики, можно достигнуть улучшения сочетанием их с генетическими алгоритмами.

ОПИСАНИЕ ПРОСТОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Генетические алгоритмы носят итерационный характер и имеют дело с обработкой популяций индивидуумов $P(t)=\{x_1^t, x_2^t, \ldots, x_n^t\}$ для итерации t (поколение t). Каждый индивидуум представляет собой потенциальное решение задачи (испытание) и представлен в некоторой, возможно достаточно сложной, структуре данных S. В качестве S будем рассматривать бинарные строки

Каждое решение x_i^t оценивается и определяется мера его "пригодности". Затем формируется новая популяция (итерация или поколение $t + 1$). На первом шаге этого формирования – этапе селекции – происходит отбор индивидуумов, обладающих лучшей пригодностью. На следующем шаге некоторые из отобранных таким образом индивидуумов подвергаются преобразованиям с помощью "генетических операторов": мутации и скрещивания. Оператор мутации создает нового индивидуума путем относительно малого изменения в одном индивидууме, а оператор скрещивания осуществляет более сильные трансформации и создает нового индивидуума путем комбинирования частей из нескольких (двух или больше) индивидуумов. После ряда итерационных шагов алгоритм сходится к лучшему из возможных решений. Остановимся теперь более подробно на трех "генетических операторах" – селекции, скрещивании и мутации.

Селекция. Целью селекции является осуществление выборки индивидуумов в текущей популяции (т.е. из некоторого набора) пропорционально их пригодности. Обычно используют четыре различных механизма селекции – “колесо рулетки”, остаточная стохастическая выборка, стохастическая равномерная выборка и турнирная селекция. Первые три алгоритма являются вариантами пропорциональной селекции, а последний – непропорциональной.

Вам предлагается использовать так называемую турнирную селекцию, не требующую предварительного ранжирования функции пригодности. При этом последовательно берутся два сосед-

них элемента текущей популяции (первый и второй, третий и четвертый и т.д.) и лучший из них (т.е. элемент, обладающий меньшим значением целевой функции или функции пригодности) помещается в промежуточную популяцию *P'*. После первого прохода (пока сформирована только половина промежуточной популяции) исходная популяция случайным образом перемешивается и описанный процесс повторяется еще один раз. Здесь лучшие или худшие индивидуумы рассматриваются в смысле их упорядочивания согласно соответствующим значениям целевой функции.

Скрещивание. Наиболее простым является *одноточечное скрещивание* – каждая выбранная таким образом пара строк скрещивается следующим образом: случайным образом выбирается положение точки сечения (целое число *k* в промежутке от 1 и *l*-1, где *l* – длина строки). Затем, путем обмена всеми элементами между позициями *k*+1 и *l* включительно, рождаются две новых строки. Например, пусть первая особь – $A=(x_1,x_2,x_3,x_4,x_5)$ а вторая соответственно $B=(y_1,y_2,y_3,y_4,y_5)$ и пусть случайно выбранная точка сечения будет после третьего гена (бита). Тогда в результате скрещивания получим две особи-потомки – $A'=(x_1,x_2,x_3,y_4,y_5)$ и $B'=(y_1,y_2,y_3,x_4,x_5)$. После этого потомки замещают родительские особи в промежуточной популяции P'. Схематично этот вариант показан на рисунке 1.1.

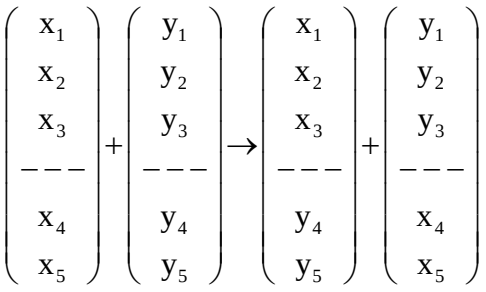

Рисунок – 1.1

Одноточечное скрещивание легко обобщается на ***n-точечное*** с *n* точками сечения. Предельным случаем является

равномерное скрещивание, при котором каждый ген первого из родителей случайным образом передается любому из потомков, при этом другой потомок, соответственно, получает ген от другого родителя.

После рекомбинации, применяется также оператор ***мутации***. Каждый бит каждой особи в популяции мутирует (точнее, пытается мутировать) с некоторой низкой вероятностью p_m. Обычно мутация применяется с вероятностью меньше чем 1 %. Обычно мутация интерпретируется как “зеркальное отражение“ бита (инверсия его значения, т.е. изменение его с 1 на 0 или с 0 на 1).

Кроме описанной ***инверсионной*** мутации можно применить оператор ***двухточечной*** мутации. В этом случае если мутация происходит, то случайным образом выбираются два гена, которые обмениваются своими значениями.

После завершения процессов выбора, рекомбинации и мутации, следующая популяция может быть оценена. Процесс оценки, выбора, рекомбинации и мутации формирует одно поколение в выполнении генетического алгоритма.

Полезно рассмотреть выполнение генетического алгоритма как двухстадийный процесс (рис. 1.2). Начинается он с *текущей популяции,* к которой применяется оператор выбора, чтобы создать промежуточную популяцию. При этом, например, особь s_2 копируется в промежуточную популяцию дважды, а s_3 – ни разу. После этого к промежуточной популяции применяются операторы рекомбинации и мутации для того, чтобы создать следующую популяцию. Процесс продвижения от текущей популяции до следующей популяции составляет одно поколение в выполнении генетического алгоритма.

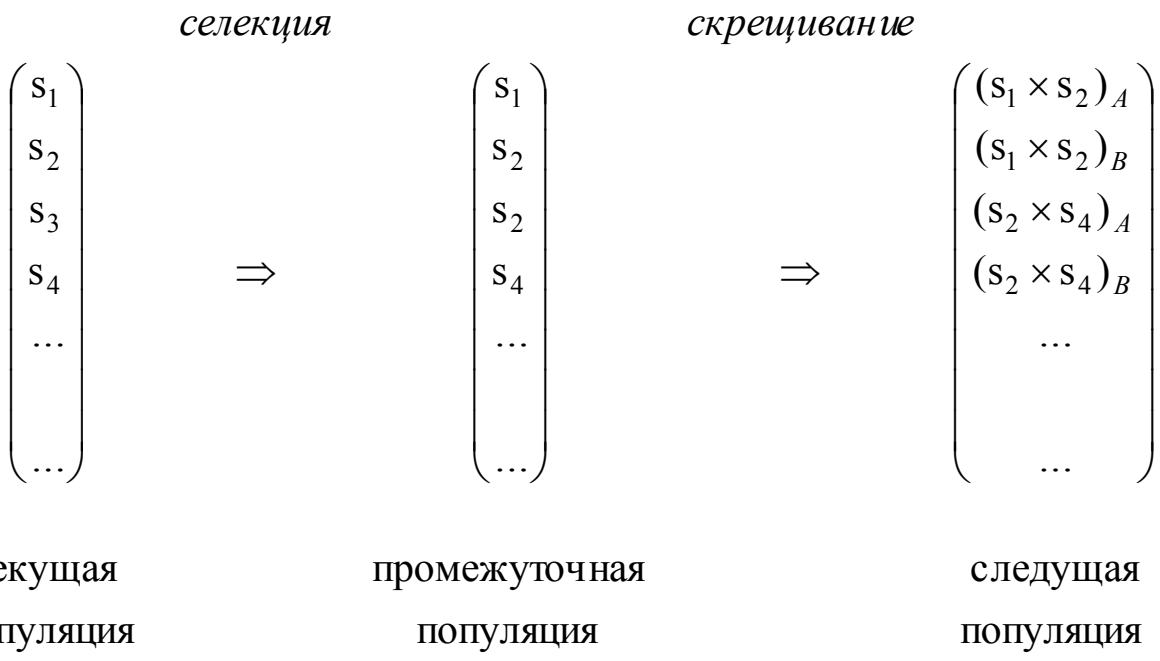

Рисунок – 1.2

В приложении приведен пример программы, реализующей простой генетический алгоритм (турнирная селекция, одноточечное скрещивание, одноточечная мутация) для нахождения минимума функции одного переменного $y = x^2, -5.12 \le x \le 5.12$.

ВАРИАНТЫ ЗАДАНИЙ

Во всех вариантах необходимо найти минимум функции $z(x, y)$ в заданной области.

При выполнении данного проекта необходимо учитывать, что решение задачи является подверженным влиянию случайных величин. Поэтому каждый запуск программы необходимо повторять, по крайней мере, 20–30 раз. После этого из набора полученных решений надо отобрать лучшее. Разумеется, это надо сделать, поместив содержание главной программы в соответствующий цикл, в котором будет одновременно выбираться наилучшее решение. Одновременно надо вычислить и среднее значение минимума за эти 20-30 прогонов.

1. $z(x, y) = x^2 + y^2, \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$

Рассмотреть одноточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 20 битами.
Провести расчеты для 40 и 80 поколений.
Сравнить получающиеся решения при размерах популяции 8, 12, 20 особей.

2. $z(x, y) = 100(x^2 - y)^2 + (1 - x)^2,$
$-2.048 \le x \le 2.048, \quad -2.048 \le y \le 2.048$

Рассмотреть одноточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

3. $z(x, y) = \text{int}(x) + \text{int}(y), \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$

Рассмотреть одноточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 50 и 100 поколений.

Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

4. $z(x, y) = x^2 + y^2, \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$

Рассмотреть двухточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

5. $z(x, y) = 100(x^2 - y)^2 + (1 - x)^2,$
$-2.048 \le x \le 2.048, \quad -2.048 \le y \le 2.048$

Рассмотреть двухточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 20 битами.
Провести расчеты для 40 и 80 поколений.
Сравнить получающиеся решения при размерах популяции 8, 12, 20 особей.

6. $z(x, y) = \text{int}(x) + \text{int}(y), \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$

Рассмотреть двухточечное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

7. $z(x, y) = x^2 + y^2, \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$

Рассмотреть равномерное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 50 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

8. $z(x,y)=100(x^2-y)^2+(1-x)^2$,
$-2.048 \le x \le 2.048,\ -2.048 \le y \le 2.048$

Рассмотреть равномерное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

9. $z(x,y)=\text{int}(x)+\text{int}(y),\quad -5.12 \le x \le 5.12,\quad -5.12 \le y \le 5.12$

Рассмотреть равномерное скрещивание и инверсионную мутацию.
Каждая переменная кодируется 20 битами.
Провести расчеты для 40 и 80 поколений.
Сравнить получающиеся решения при размерах популяции 8, 12, 20 особей.

10. $z(x,y)=x^2+y^2,\quad -5.12 \le x \le 5.12,\quad -5.12 \le y \le 5.12$

Рассмотреть одноточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

11. $z(x,y)=100(x^2-y)^2+(1-x)^2$,
$-2.048 \le x \le 2.048,\ -2.048 \le y \le 2.048$

Рассмотреть одноточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 50 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

12. $z(x,y)=\text{int}(x)+\text{int}(y),\quad -5.12 \le x \le 5.12,\quad -5.12 \le y \le 5.12$

Рассмотреть одноточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

13. $z(x, y) = x^2 + y^2, \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$
Рассмотреть двухточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 20 битами.
Провести расчеты для 40 и 80 поколений.
Сравнить получающиеся решения при размерах популяции 8, 12, 20 особей.

14. $z(x, y) = 100(x^2 - y)^2 + (1 - x)^2,$
$-2.048 \le x \le 2.048, \quad -2.048 \le y \le 2.048$
Рассмотреть двухточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 50 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

15. $z(x, y) = \text{int}(x) + \text{int}(y), \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$
Рассмотреть двухточечное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

16. $z(x, y) = x^2 + y^2, \quad -5.12 \le x \le 5.12, \quad -5.12 \le y \le 5.12$
Рассмотреть равномерное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.

Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

17. $z(x, y) = 100(x^2 - y)^2 + (1 - x)^2,$
$-2.048 \le x \le 2.048, \ -2.048 \le y \le 2.048$

Рассмотреть равномерное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 20 битами.
Провести расчеты для 40 и 80 поколений.
Сравнить получающиеся решения при размерах популяции 8, 12, 20 особей.

18. $z(x, y) = \text{int}(x) + \text{int}(y), \quad -5.12 \le x \le 5.12, \ -5.12 \le y \le 5.12$

Рассмотреть равномерное скрещивание и двухточечную мутацию.
Каждая переменная кодируется 30 битами.
Провести расчеты для 30 и 100 поколений.
Сравнить получающиеся решения при размерах популяции 10, 20, 30 особей.

ЛИТЕРАТУРА

1. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.:ФИЗМАТЛИТ, 2003. – 432 с.
2. Батищев Д.А. Генетические алгоритмы решения экстремальных задач. – Воронеж: Изд.-во ВГТУ, 1995.

ПРИЛОЖЕНИЕ

```
program sga;
uses crt;

const
        maxpop = 100;
        maxstring = 30;
        dim = 1; {размерность пространства поиска}

type
        allele = boolean;
                {Алель – позиция в битовой строке}
        chromosome = array[1..maxstring*dim] of allele;
                {битовая строка}
        fenotype = array[1..dim] of real;
        individual = record
                chrom:chromosome;
                        {генотип = битовая строка}
                x:fenotype; {фенотип = массив вещественных
                                координат точки в пространстве поиска}
                fitness:real; {значение целевой функции}
                end;
        population = array[1..maxpop] of individual;

const
        xmax:fenotype=(5.12);
                {массив максимальных значений для координат точки в пространстве поиска}
        xmin:fenotype=(-5.12);
                {массив минимальных значений для координат точки в пространстве поиска}

var
        oldpop, newpop, intpop :population;
                {Три непересекающихся популяции – старая, новая и промежуточная}
        popsize, lchrom, gen, maxgen: integer;
                {Глобальные целые переменные}
        pcross, pmutation, sumfitness:real;
                {глобальные вещественные переменные}
        nmutation, ncross:integer;
```

```
            {Статистические целые}
        avg, max, min:real; {Статистические вещественные}

{Вероятностные процедуры}

function random_:real;
begin
        random_ := random(65535)/(65535-1);
end;

function flip(probability:real):boolean;
        {подбрасывание монетки – true если орел}
begin
        if probability = 1.0 then
                flip := true
        else
                flip := (random_ <= probability);
end;

function rnd(low,high:integer):integer;
        {Случайный выбор между low и high}
var
        i:integer;
begin
        if low >= high then
                i := low
        else begin
                i := trunc( random_ * (high-low+1) + low);
                if i > high then i := high;
        end;
        rnd := i;
end;

{интерфейсные процедуры: decode and objfunc}

function objfunc(x:fenotype):real;
begin
        objfunc:= sqr(x[1]);
end;

procedure decode(chrom:chromosome; lbits:integer; var x:fenotype);
        {Декодирование строки в массив вещественных координат точки в
        пространстве поиска - true=1, false=0}
```

```
var
        i,j:integer;
        f, accum, powerof2:real;
begin
        for i:=1 to dim do begin
                accum := 0.0;
                powerof2 := 1;
                f:=1;
                for j := 1+lbits*(i-1) to lbits+lbits*(i-1) do begin
                        if chrom[j] then accum := accum + powerof2;
                        powerof2 := powerof2 * 2;
                        f:=f*2;
                end;
                x[i] := xmin[i]+(xmax[i]-xmin[i])*accum/(f-1)
        end
end;

{Расчет статистических величин: statistics }
procedure statistics(popsize:integer; var max,avg,min,sumfitness:real;
                                var pop:population);
        {Расчет статистик популяции }
var
        j:integer;
begin
        {Инициализация }
        sumfitness := pop[1].fitness;
        min := pop[1].fitness;
        max := pop[1].fitness;
        {Цикл для max, min, sumfitness }
        for j := 2 to popsize do with pop[j] do begin
                sumfitness := sumfitness + fitness;
                {Накопление суммы значений функции пригодности}
                if fitness>max then max := fitness;
                {Новое значение max}
                if fitness<min then min := fitness;
                {Новое значение min}
                end;
        {Расчет среднего}
        avg := sumfitness/popsize;
end;

{Процедура инициализации initpop}
procedure initpop;
```

```
        {Инициализация начальной популяции случайным образом}
var
        j, j1:integer;
begin
        for j := 1 to popsize do with oldpop[j] do begin
                for j1 := 1 to lchrom*dim do chrom[j1] := flip(0.5);
                        {Бросок монетки}
                decode(chrom,lchrom,x);
                        {Декодирование строки}
                fitness := objfunc(x);
                        {Вычисление начальных значений функции пригодно-
                сти}
        end;
end;

{3 генетических оператора: отбора (select), скрещивания (crossover) и
мутации (mutation)}

procedure select;
        {процедура выбора}
var
        ipick:integer;
        procedure shuffle(var pop:population);
                {процедура перемешивания популяции в процессе отбора}
        var
                i,j:integer;
                ind0:individual;
        begin
                for i := popsize downto 2 do begin
                        j:= random(i-1)+1;
                        ind0:=pop[i];
                        pop[i]:=pop[j];
                        pop[j]:=ind0;
                end;
        end;

        function select_1:integer;
        var
                j1,j2,m:integer;
        begin
                if (ipick>popsize) then begin
                        shuffle(oldpop);
                        ipick:=1
```

```
                end;
                j1:=ipick;
                j2:=ipick+1;
                if (oldpop[j2].fitness<oldpop[j1].fitness) then
                        m:=j2
                else
                        m:=j1;
                ipick:=ipick+2;
                select_1:=m;
        end;

var
        j:integer;
begin
        ipick:=1;
        for j:=1 to popsize do begin
                intpop[j]:=oldpop[select_1];
        end;
        oldpop:=intpop;
end;

function mutation (alleleval:allele; pmutation:real;
                        var nmutation:integer):allele;
{мутация одного бита в строке (аллеля) с вероятностью pmutation, count number of mutations}
var
        mutate:boolean;
begin
        mutate := flip(pmutation);
                {Flip the biased coin}
        if mutate then begin
                nmutation := nmutation + 1;
                mutation := not alleleval;
                        {Change bit value}
        end else
                mutation := alleleval;
                        {No change}
end;

procedure crossover(var parent1, parent2, child1, child2:chromosome;
                flchrom:integer; var ncross, nmutation, jcross:integer;
                var pcross, pmutation:real);
```

```
{Скрещивание 2 родительских строк, результат помещается в 2 строках-потомках}
var
	j:integer;
begin
	if flip(pcross) then begin
		{Выполняется скрещивание с вероятностью pcross}
		jcross := rnd(1,flchrom-1);
			{Определение точки сечения в диапазоне между 1 и l-1}
		ncross := ncross + 1;
			{Инкрементирование счетчика скрещиваний}
	end else
		jcross := flchrom;
	{певая часть обмена , 1 to 1 and 2 to 2}
	for j := 1 to jcross do begin
		child1[j] := mutation(parent1[j], pmutation, nmutation);
		child2[j] := mutation(parent2[j], pmutation, nmutation);
	end;
	{вторая часть обмена, 1 to 2 and 2 to 1]
	if jcross<>flchrom then
		{пропуск, если точка скрещивания равна flchrom--скрещивание
		не происходит}
	for j := jcross+1 to flchrom do begin
		child1[j] := mutation(parent2[j], pmutation, nmutation);
		child2[j] := mutation(parent1[j], pmutation, nmutation);
	end;
end;

{Процедура создания нового поколения: generation}

procedure generation;
	{Генерирование нового поколения при помощи отбора, скрещивания
	и мутации}
	{Прим: предполагается, что популяция имеет четный размер}
var
	j, mate1, mate2, jcross:integer;
begin
	select;
	j := 1;
	repeat
		{выполняются отбор, скрещивание и мутация, пока полностью
		не сформируется новая популяция – newpop}
		mate 1:= j;
```

```
                {выбор родительской пары}
            mate 2:= j+1;
            {скрещивание и мутация – мутация вставлена в процедуру
            скрещивания}
            crossover(oldpop[mate1].chrom, oldpop[mate2].chrom,
                newpop[j ].chrom, newpop[j + 1].chrom,
                lchrom*dim, ncross, nmutation, jcross, pcross, pmutation);
            {Декодирование строки и вычисление пригодности}
            with newpop[j ] do begin
                decode(chrom, lchrom,x);
                fitness := objfunc(x);
            end;
            with newpop[j+1] do begin
                decode(chrom, lchrom,x);
                fitness := objfunc(x);
            end;
            j := j + 2;
        until j>popsize
end;

begin { Главная программа }
        popsize:=20;
        {размер популяции}
        lchrom:=20;
        {число битов на один кодируемый параметр}
        maxgen:=100;
        {максимальное число поколений}
        pmutation:=0.01;
        {вероятность скрещивания}
        pcross:=0.9;
        {вероятность мутации}
        {Инициализация генератора случайных чисел}
        randomize;
        {Инициализация счетчиков}
        nmutation := 0;
        ncross := 0;
        initpop;
        statistics (popsize, max, avg, min, sumfitness, oldpop);
        gen:= 0; {Установка счетчика поколений в 0}
        repeat {Главный итерационный цикл}
            gen:= gen + 1;
            generation;
            statistics(popsize, max, avg, min, sumfitness, newpop);
```

```
            oldpop:= newpop;
                {переход на новое поколение }
            writeln('min= ',min);
        until (gen >= maxgen)
end. {End главной программы}
```