Министерство образования и науки РФ
ФГБОУ ВО «Томский государственный университет
систем управления и радиоэлектроники»
Кафедра комплексной информационной безопасности
электронно-вычислительных систем (КИБЭВС)

Т.Т. Газизов, А.О. Мелкозеров

ОПТИМИЗАЦИЯ ГЕНЕТИЧЕСКИМИ АЛГОРИТМАМИ

Учебное пособие

для студентов специальностей и направлений
10.03.01 – «Информационная безопасность»,
10.05.02 – «Информационная безопасность
телекоммуникационных систем»,
10.05.03 – «Информационная безопасность
автоматизированных систем»,
10.05.04 – «Информационно-аналитические системы безопасности»
09.03.02 – «Информационные системы и технологии»

В-Спектр
Томск, 2017

УДК 004.021
ББК 30.2-5-05
Г 13

Г 13 **Газизов Т.Т., Мелкозеров А.О.** Оптимизация генетическими алгоритмами: учебное пособие. – Томск: В-Спектр, 2017. – 30 с.
ISBN 978-5-91191-376-2

В пособии рассмотрены общие вопросы теории генетических алгоритмов, реализация их возможностей в библиотеке GALib и описание практического использования генетических алгоритмов. Пособие предназначена для студентов специальностей 10.05.03 – «Информационная безопасность автоматизированных систем», 09.03.02 – «Информационные системы и технологии», 10.05.04 – «Информационно-аналитические системы безопасности» и направления 10.03.01 – «Информационная безопасность», а также дисциплины «Администрирование сетей ЭВМ» для специальности 10.05.02 – «Информационная безопасность телекоммуникационных систем».

УДК 004.056
ББК 32.973.26-018.2

Работа выполнена при финансовой поддержке Министерства образования и науки РФ в рамках базовой части государственного задания ТУСУРа на 2017–2019 годы (проект № 2.8172.2017/БЧ)

ISBN 978-5-91191-376-2

Оглавление

1. Введение

Перед тем как приступить к изучению генетических алгоритмов (ГА), необходимо понять: какую задачу Вы собираетесь решать с их помощью. Далее следует цитата К. Де Йонга (крупного специалиста в области ГА), одного из студентов Д. Холланда, а ныне профессора Университета Дж. Мейсона, о целесообразности использования ГА:

The key point in deciding whether or not to use genetic algorithms for a particular problem centers around the question: what is the space to be searched? If that space is well understood and contains structure that can be exploited by special-purpose search techniques, the use of genetic algorithms is generally computationally less efficient. If the space to be searched is not well understood and relatively unstructured, and if an effective GA representation of that space can be developed, then GAs provide a surprisingly powerfull search heuristic for large, complex spaces [De Jong K.A. Introduction to the second special issue on genetic algorithms / Machine Learning. 1990. № 5(4). P. 351–353].

Решающий аргумент использования генетических алгоритмов тесно связан с вопросом о том, какое пространство поиска будет исследовано. Если это пространство легко анализировать и его топология позволяет использовать специализированные технологии поиска, то использование генетических алгоритмов не эффективно с точки зрения затрат вычислительных ресурсов. Если же пространство поиска не поддается анализу и мало структурировано, и если существует эффективный способ генетического отображения этого пространства, то ГА представляют удивительно впечатляющий поисковый метод в больших и сложных областях [Де Йонг К.А. Введение ко второму специальному выпуску по генетическим алгоритмам / Машинное обучение. 1990. № 5(4). С. 351–353].

Большой плюс эволюционных вычислений заключается в предоставляемом ими унифицированном подходе к решению самых различных проблем. Так же, для сложных переборных задач (большинство из которых NP-полные, т.е. не решаются полным перебором за полиномиальное время), таких как задача коммивояжера и поиск булевых термов, ГА показывают блестящие результаты, о чем и говорит Де Йонг [1].

2. Общие сведения о ГА

Сегодня наибольшее распространение из всех эволюционных алгоритмов получил ГА. На его основе осуществляются: оптимизация профилей балок в строительстве, обработка рентгеновских изображений в медицине, оптимизация работы нефтяных трубопроводов [2] и т.д. Основанный на теории Ч. Дарвина, генетический алгоритм был использован учёными в компьютерных исследованиях, так чтобы он функционировал по аналогии с законами эволюции, в основе которой лежат принципы естественного отбора. Эволю-

ция состоит из последовательности поколений. При этом в каждом поколении отбираются те особи, которые имеют большее значение приспособленности (естественный отбор). Далее следует рекомбинация и малая мутация хромосом, отобранных особей (особь представляет собой набор хромосом, а каждая хромосома состоит из генов, поэтому под рекомбинацией и мутацией хромосом понимаются операции над генами), в итоге, мы получаем новое, более приспособленное поколение.

Опишем ГА с технической точки зрения. Рассмотрим случай функции одной переменной. Пусть имеются пространство решений, пространство независимых переменных и взаимно однозначное отображение, переводящее пространство переменных в пространство решений (целевая функция). Необходимо найти такую переменную, при которой решение будет удовлетворять заданному условию. Изначально указываем допустимые пределы изменения истинного решения (можно не делать). Далее, хаотично выбираем n численных значений переменной (формируем первое поколение). Представляем каждое из значений переменной в бинарном виде (традиционно, ГА базируется на системе состоящей из бинарных символов, для простоты следующих операций рекомбинации и мутации). Далее, выполняем рекомбинации и мутации (будут подробно рассмотрены ниже). Значения переменных, полученные в результате рекомбинаций и мутаций, представляем в десятичном виде. Далее, для этих значений переменных вычисляем функцию пригодности и выбираем k значений переменных удовлетворяющих исходному условию. Далее формируем новое поколение, состоящее из k только что полученных значений и m новых (хаотично выбранных) значений переменных ($n = k + m$). Все указанные выше шаги повторяются многократно. В итоге, получаем, что каждое следующее поколение лучше предыдущего.

Остановимся подробней на операциях рекомбинации и мутации. Существует несколько видов рекомбинации. Мы будем использовать наиболее простой из них, называемый одноклеточным кроссовером или кроссинговером [1]. Его суть заключается в следующем: пусть имеются два вектора $\mathbf{a_1}(a_{11}, a_{12}, \ldots, a_{1N})$ и $\mathbf{a_2}(a_{21}, a_{22}, \ldots, a_{2N})$, тогда в результате кроссинговера мы получим два новых вектора $(a_{11}, a_{12}, \ldots a_{1m}\, a_{2,m+1}, a_{2N})$ и $(a_{21}, a_{22}, \ldots, a_{2m}\, a_{1,m+1}, a_{1N})$. При этом точка m выбирается случайным образом. Мутация представляет собой случайный выбор какого-либо значения переменной, выполняемый следующим образом. Значение переменной представляют в двоичном виде. Часть разрядов числа, в соответствии с коэффициентом мутации, изменяют на противоположное значение (с 0 на 1 и наоборот). Важными параметрами являются соответствующие коэффициенты. Так, коэффициент рекомбинации обычно лежит в пределах от 0,6 до 1, когда коэффициент мутации чаще всего берут равным 0,1. Мутация не даёт прогресса в поиске решения, но страхует от локальных максимумов либо минимумов.

3. Терминология ГА

Определимся с используемой терминологией относительно ГА:

– генетический алгоритм – (ГА) метод или путь решения поставленной задачи;

– геном – класс возможных решений, дает представление о том, каким решение может быть вообще;

– целевая функция – взаимно однозначное отображение, переводящее пространство переменных в пространство решений, возвращает значение пригодности переменной;

– ген – один из параметров задачи;

– особь (хромосома, индивид) – набор генов, одно из решений;

– популяция – набор особей, совокупность решений;

– поколение – цикл жизни популяции: от создания до формирования нового;

– эволюция – последовательность поколений до достижения условия останова ГА.

4. Общие сведения о GAlib

Библиотека GALib – это полнофункциональная библиотека объектов и методов для разработки ГА. Она включает в себя типы данных, различные классы генотипов, ГА, популяций, схем селекций, классов сбора и накопления статистики по работе алгоритма и классы генерации случайных чисел. Все права на распространение принадлежат Массачусетскому Технологическому Институту и автору программы – Мэтью Волу (Matthew Wall) [3]. Библиотеку можно использовать и распространять сторонним лицам в целях разработки некоммерческих программ.

Структура GAlib представлена на рис. 4.1.

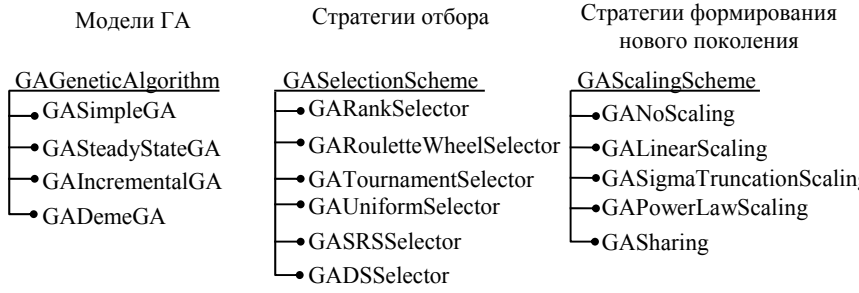

Рис. 4.1. Структура ГА

Классы ГА:

– GAGeneticAlgorithm (основной класс ГА),

– GASimpleGA (не перекрывающиеся популяции),
– GASteadyStateGA (перекрывающиеся популяции),
– GAIncrementalGA (возрастающие популяции),
– GADemeGA (параллельные популяции с миграцией),
– GAStatistics (класс статистики ГА),
– GAParameterList (класс параметров ГА),
– GAPopulation (класс популяции ГА),
– GAScalingScheme (класс стратегий отбора для скрещивания),
– GANoScaling (без оценки),
– GALinearScaling (линейная оценка),
– GASigmaTruncationScaling (оценка с уменьшением ошибки),
– GAPowerLawScaling (оценка по закону власти),
– GASharing (оценка по разделению),
– GASelectionScheme (класс стратегий отбора для формирования будущих поколений),
– GARankSelector,
– GARouletteWheelSelector,
– GATournamentSelector,
– GAUniformSelector,
– GASRSSelector,
– GADSSelector.

4.1. Основные возможности GALib

Библиотека GALib может быть скомпилирована под операционными системами DOS/Windows, Windows NT/95, MacOS, и UNIX системами.

• Параметры ГА могут быть заданы из файла, с командной строки и/либо исходного кода программы.

• Возможно использование ГА с перекрывающимися и не перекрывающимися поколениями, также возможен выбор процента перекрытия.

• Возможна разработка своего ГА благодаря возможности создания своих классов на базе основного класса ГА.

• Возможно применение элитизма для не перекрывающихся видов ГА.

• Реализованы основные виды отбора: ранговый, рулетка, турнир, стохастический, универсальный, вероятностный.

• Доступна полная статистика по всем параметрам эволюции.

• Хромосомы могут быть созданы из любого типа данных, доступных в C++.

• Доступны операторы инициализации: uniform random, order-based random, and initialize-to-zero.

• Доступны операторы мутации: random flip, random swap, Gaussian, destructive, swap subtree, swap node, swap node.

• Доступны операторы кроссовера: partial match, ordered, cycle, single point, two point, even, odd, uniform, node- and subtree-single point.

5. Реализация ГА в GAlib

При работе с библиотекой GAlib вы работаете с двумя классами: Genome и Genetic Algorithm. Genome – геном, представляет собой совокупность возможных решений Вашей задачи. Genetic Algorithm – класс ГА, определяющий, как это решение должно достигаться. Как говорилось выше, ГА использует целевую функцию (которую определяет пользователь), чтобы вычислить степень «приспособленности» каждого генома к выживанию. Чтобы решить поставленную задачу, используя ГА, необходимо:

- определить представление чисел;
- выбрать операторы ГА;
- задать целевую функцию.

Структурная схема работы ГА представлена на рис. 5.1.

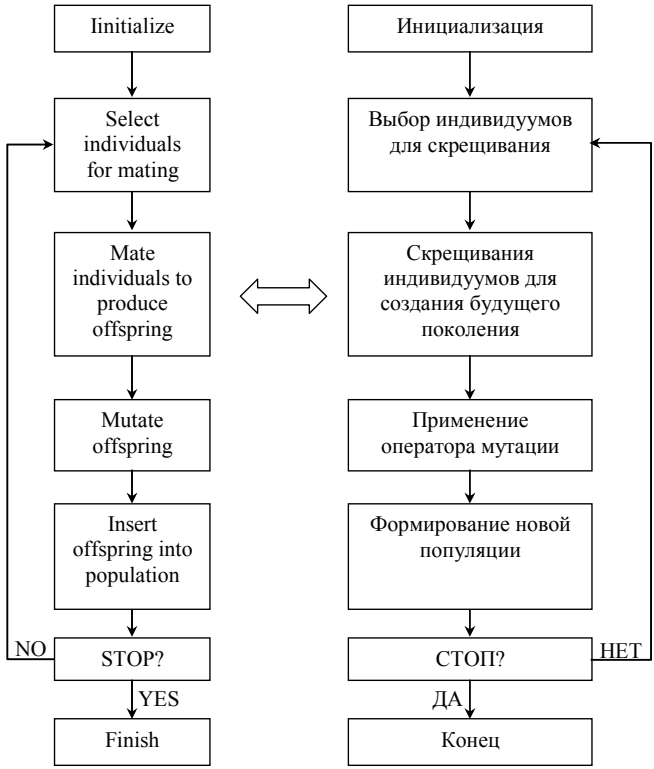

Рис. 5.1. Схема работы ГА

Библиотека GAlib предоставляет полный набор инструментов, чтобы быстро и просто реализовать представление чисел и использовать нужные операторы ГА. Но целевую функцию должен задать именно пользователь. Существует много разных видов ГА. GAlib включает четыре основных типа: «простой» (simple), «устойчивый» (steady-state), «возрастающий» (incremental), «параллельный» (deme). Все они отличается только тем КАК новые индивидуумы будут заменять старых в ходе эволюции. Стоит отметить, выбор типа ГА полностью зависит от сложности поставленной задачи. Правильное использование ГА позволяет использовать их как при глобальном, так и при локальном поиске. При этом временные затраты зависят от корректного выбора операторов и параметров ГА, а так же от сложности «формы» пространства решений.

6. Работа ГА в GAlib

ГА определяет, какие индивидуумы должны выжить, какие участвовать в скрещивании, а какие умереть. Так же ГА определяет, как долго будет длиться процесс эволюции. Обычно ГА не имеет четкого условия останова. Вы должны указать алгоритму, когда остановится. Часто таким критерием является количество поколений или вырождение популяции, т.е. если практически нет разнообразия в генах особей популяции, либо просто вышел лимит времени.

Библиотека содержит 4 типа ГА. Первый – простой ГА («simple»). Этот алгоритм является одним из первых и самых простых. Он использует не перекрывающиеся популяции и элитизм. С каждым поколением алгоритм создает полностью новую популяцию индивидуумов. Второй тип – устойчивый ГА («steady-state») использует перекрывающиеся популяции. В этом алгоритме вы можете определить, какая часть популяции будет заменена в каждом поколении. Третий вариант есть возрастающий ГА («incremental»), в котором каждое поколение состоит только из одного или из двух детей. Возрастающий ГА позволяет обычными методами замены определить, как новые особи попадут в популяцию. Так, например, полученный ребенок может занять место родителя, может заменить любого члена популяции случайным образом, либо заменить индивидуума, который больше всего похож на него. Четвертый тип – групповой ГА («deme»). Этот алгоритм создает параллельно множество популяций, используя устойчивый ГА. В этом алгоритме с каждым поколением несколько индивидуумов одной популяции мигрируют в другую популяцию.

6.1. Представление чисел

Используйте структуру данных, которая соответствует вашей задаче. Если вы оптимизируете функцию реальных чисел, используйте реальные чис-

ла в вашем геноме. Если решение вашей задачи может быть представлено мнимыми числами или целыми числами, используйте соответствующий геном.

Задание соответствующего представления чисел – часть искусства использования ГА. Ваше представление чисел должно отражать любое решение вашей задачи, но вы должны выбрать его так, чтобы оно не отражало недопустимых решений вашей задачи. Помните, что если геном может представлять недопустимые решения, то целевая функция должна их учитывать.

Представление чисел не должно содержать информацию больше той, которая необходима для представления решения задачи. Какое бы представление чисел вы не выбрали, используйте те операторы, которые соответствуют вашему представлению.

6.2. Операторы ГА

Каждый геном имеет три основных оператора: инициализация, мутация, кроссовер. Используя эти операторы, вы можете влиять на начальную популяцию, определить особенности мутации и кроссовера для вашей задачи, либо изменять части ГА по мере развития вашей популяции. В библиотеке GAlib все эти операторы имеют установки по умолчанию для каждого типа генома, но вы можете изменить любой из них.

Оператор инициализации определяет, как будет инициализирован геном. Он вызывается, когда вы инициализируете популяцию или ГА. Этот оператор не создает геном, скорее он «наполняет» геном первоначальным генетическим материалом, из которого будут получены все решения. Класс популяции имеет свой оператор инициализации. По умолчанию, в популяции просто вызывается оператор инициализации генома, но вы можете вызывать тот оператор инициализации, который захотите.

Оператор мутации определяет процесс мутации каждого генома. В разных типах данных мутация действует по-разному. Например, типичный оператор мутации, примененный к бинарной строке, проинвертирует биты в строке с заданной вероятностью. Оператор мутации должен производить новый генетический материал так же хорошо, как изменять существующий.

Оператор кроссовера определяет процедуру создания ребенка от двух родителей генома. Как и у оператора мутации, действие оператора кроссовера зависит от типа данных. В отличие от мутации, кроссовер включает множество геномов. В GAlib, каждый геном «знает» свой предпочитаемый метод скрещивания (метод кроссовера по умолчанию).

Помимо трех основных операторов, каждый геном должен также содержать целевую функцию и так же может содержать компаратор. Целевая функция используется для развития генома. Компаратор (часто упоминается как функция расстояния) используется, чтобы определить насколько один

геном отличается от другого. Для каждого ГА необходима целевая функция – именно по ней ГА определяет, какие индивидуумы лучше, чем другие. Для некоторых ГА необходим компаратор.

6.3. Классы GAlib

6.3.1. GASimpleGA

Простой ГА (без перекрываюшихся поколений).

Этот ГА использует не перекрывающиеся поколения. Когда вы создаете простой ГА вы должны определить либо индивидуум, либо популяцию индивидуумов. Новый ГА будет клонировать индивидуум(ов), который(х) вы определили, чтобы создать свое собственное поколения. Вы можете изменять поведение ГА после запуска, непосредственно в ходе эволюции.

Простой ГА создает начальное поколение за счет клонирования индивидуума или поколения, введенное Вами, когда Вы его создали. С каждым поколением ГА создает полностью новое поколение индивидуумов за счет отбора из предшествующего поколения, затем скрещивания для создания нового поколения для нового поколения. Этот процесс продолжается пока критерий останова не будет достигнут.

Возможно использование элитизма. По умолчанию, элитизм включен, это означает, что лучший индивид из каждого поколения переносится в следующее поколение.

Частота счета для этого ГА по умолчанию равна 1 (запись лучшего индивидуума поколения ведется с каждым поколением). Вычисление пригодности, по умолчанию: линейное, метод выборки: рулетка.

6.3.2. GASteadyStateGA

Установившийся ГА (перекрывающиеся поколения).

Этот ГА схож с алгоритмом описанным ДеДжонгом. Он использует перекрывающиеся поколения с коэффициентом перекрытия заданным пользователем. Алгоритм создает поколение индивидуумов за счет клонирования генома или поколения, которое вы ввели, когда создали его. С каждым поколением алгоритм создает временные поколения индивидуумов, прибавляет их к предшествующим поколениям, затем удаляет наихудшие индивидуумы, чтобы вернуть поколение к первоначальному размеру.

Вы можете выбрать число перекрытий между поколениями. Это процент той части популяции, которая будет заменяться с каждым поколением. Новое созданное поколение добавляется к популяции, затем наихудшие индивидуумы уничтожаются (поэтому новое поколение может попасть, а может не попасть в популяцию, зависит от того лучше они, чем наихудшие индивидуумы в популяции или нет).

Если вы определите процент замены, тогда этот процент популяции будет заменяться с каждым поколением. Либо можно задать количество индивидуумов (меньшее числа в популяции) заменяемое в каждом поколении.

Частота счета по умолчанию – 100 (алгоритм записывает лучшего индивидуума поколения каждые 100 поколений). Вычисление пригодности, по умолчанию: линейное, метод выборки: рулетка.

6.3.3. GAIncrementalGA

Возрастающий ГА (перекрывающиеся поколения с 1 или 2 детьми за поколение).

ГА схож с теми, что основаны на модели GENITOR (коэффициент мутации устанавливается адаптивно в зависимости от разнообразия популяции). Он использует перекрывающиеся популяции, но с очень маленьким перекрытием (только 1 или 2 индивидуума заменяются в каждом поколении). По умолчанию, используется схема замещения, при которой заменяется наихудшая особь в поколении (стратегия замены наихудшего).

Так как данный ГА основан на двухклеточном кроссовере, количество потомства должно быть равно 1 или 2. По умолчанию 2.

Необходимо определить стратегию замены, которую будет использовать ГА. Стратегия замены определяет, как новые дети будут внедрены в популяцию. Если вы хотите, чтобы ребенок заменял случайного члена популяции, используйте «случайную стратегию». Если вы хотите, чтобы ребенок заменял наихудшего члена популяции, используйте «наихудшую стратегию».

Частота счета по умолчанию – 100 (алгоритм записывает лучшего индивидуума поколения каждые 100 поколений). Вычисление пригодности, по умолчанию: линейное, метод выборки: рулетка.

6.3.4. GADemeGA

Групповой ГА (параллельные популяции с миграцией).

ГА имеет множество независимых популяций. Он создает популяции за счет клонирования генома или популяции, которую Вы ввели, создавая его.

Каждое поколение развивается, используя установившийся ГА, но в каждом поколении некоторые индивидуумы мигрируют из одной популяции в другую. Алгоритм мигрирования заключается в детерминированном методе ступенек; каждое поколение перемещает определенное количество своих лучших индивидуумов к своему соседу. Главная популяция обновляется с каждым поколением за счет лучших индивидуумов из каждой популяции.

6.4. Схемы отбора ГА

Схемы отбора используются, чтобы отбирать геномы из популяции для скрещивания. Схемы отбора определяют основное поведение селектора. Каж-

дый объект отбора может быть привязан к популяции, из которой он будет производить отбор. Член селектора возвращает ссылку на единичный геном. Селектор может оперировать над нормированными целевыми значениями обычных целевых значений. Поведение по умолчанию: оперировать с нормированными значениями. Ниже представленный конструкторы схем отбора.

6.4.1. GARankSelector

Селектор по рангу.

Отбирает лучшего члена популяции каждый раз.

6.4.2. GARouletteWheelSelector

Селектор по рулетке.

Этот метод селекции отбирает индивидуумов, основываясь на величине значения приспособленности относительно всей популяции. Чем выше значение приспособленности, тем вероятнее индивид будет отобран. Каждый индивидуум может быть выбран с вероятностью p, где p равно отношению приспособленности индивидуума к сумме приспособленностей каждого индивидуума в популяции.

6.4.3. GATournamentSelector

Селектор по турниру.

Турнирный отбор использует рулеточный метод отбора, чтобы отобрать два индивидуума, затем отбирает тот, у которого выше значение приспособленности. Обычно, турнирный отбор выбирает более приспособленных индивидуумов чаще, чем рулеточный отбор.

6.4.4. GASRSSelector

Селектор по детерминированному осуществлению выборки.

Этот селектор использует двух уровневую процедуру. На первом уровне высчитывается ожидаемое представление каждого индивидуума. Временная популяция заполняется индивидуумами с наилучшими ожидаемыми результатами. Все оставшиеся места заполняются первоначальными индивидуумами в соответствии с десятичной частью их ожидаемого представления, затем отбираются наилучшие. Второй уровень селекции есть обычный случайный отбор из временного поколения.

6.4.5. GADSSelector

Селектор по осуществлению стохастической выборки остатка.

Стохастическая выборка остатка использует двух уровневую процедуру отбора. На первом уровне высчитывается ожидаемое представление каждого индивидуума. Временная популяция заполняется индивидуумами с наи-

большими ожидаемыми значениями. Любые частично ожидаемые представления используются, чтобы индивидуумы наименее вероятно заполнили пространство. Например, индивид со значением е 1.4 будет на первой позиции, чем а с возможностью 40% занять вторую позицию. Второй уровень селекции есть обычная случайная выборка из временной популяции.

6.4.6. GAUniformSelector

Селектор: стандартный

Стохастическое стандартное осуществление выборки производится случайным образом из популяции. Любой индивид в популяции имеет вероятность p быть выбранным, где p есть отношение 1 к размеру популяции.

6.4.7. Схемы оценки ГА

Объект оценки вложен в объект популяции. Этот объект следит за значением приспособленности каждого индивидуума в популяции. Именно на основе этой оценки происходит формирование нового поколения. Ниже представлены конструкторы схем оценивания.

6.4.8. GANoScaling

Без оценки.

Значения приспособленности равны целевым значениям. Оценки нет.

6.4.9. GALinearScaling

Линейная оценка.

Значения пригодности получают из целевых значений, используя линейный метод оценки, описанный в книге Голдберга. Вы можете задать оценочный коэффициент. Отрицательные целевые значения не допускаются в этом методе. Целевые значения преобразуются в значения пригодности через выражение:

$$f = a \cdot obj + b \,,$$

где a и b вычисляются на основе целевых значений индивидуумов в популяции, как описано в книге Голберга.

6.4.10. GASigmaTruncationScaling

Оценка с уменьшением ошибки.

Используйте этот метод оценки, если ваши целевые значения отрицательны. Эта оценка основана на вариации среднего популяции и произвольном усечении до 0. Соответствие целевого значения пригодности для каждого индивидуума равно:

$$f = obj - (obj\_ave - c \cdot obj\_dev) \,.$$

6.4.11. GAPowerLawScaling

Оценка по закону власти.

Отображение целевого значения значению пригодности по закону власти определяется через экспоненциальное выражение:

$$f = obj^k .$$

6.4.12. GASharing

Оценка по разделению.

Этот метод оценки используется при видообразовании в процессе эволюции. Значение пригодности получается из целевого значения за счет сравнения индивидуума с другими индивидуумами в популяции. Если есть другие такие же индивидуумы, то значение пригодности уменьшается. Функция расстояния используется, чтобы определить, насколько схожи индивидуумы между собой. Функция расстояния должна возвращать значение 0 и выше, где 0 означает, что два индивидуума одинаковы. Для данного индивидуума:

$$f = \frac{obj}{\sum_{j=0}^{n} s(d_j)},$$

$$s(d_j) = \begin{cases} d_j < \sigma \to 1 - \left(\frac{d_j}{\sigma}\right)^{\alpha}, \\ d_j \geq \sigma \to 0. \end{cases}$$

d_j – значение расстояния между текущим индивидуумом и индивидуумом j;

n – количество индивидуумов в популяции.

По умолчанию разделяющий объект использует треугольную функцию разделения, описанную в книге Голдберга. Вы можете определить значение сокращения (σ в книге Голдберга), используя сигма член функции. Искривление функции разделения контролируется значением α. Когда α равна 1.0 функция разделения – прямая линия (треугольное разделение). Если вы определяете компаратор, эта функция будет использована как функция расстояния для всех сравнений. Если вы не определяете компаратор, разделяющий объект будет использовать компаратор, установленный по умолчанию, для каждого генома.

Заметим, что оценка по разделению различна, в зависимости от того будет ли целевая функция минимизироваться или максимизироваться. Если задача максимизировать целевое значение, повторные значения будут разде-

лены на разделяющий фактор. Если задача минимизировать целевое значение, повторные значения будут умножены на разделяющий фактор.

7. Практическое использование ГА в системе TALGAT

Считается, что Вы внимательно ознакомились с методическими указаниями к курсовой работе «Основы электромагнитной совместимости радиоэлектронной аппаратуры» и работали с системой TALGAT (далее система), а так же имеете навык использования TALGAT_Script – встроенного в систему интерпретируемого скриптового языка, с помощью которого пользователь «общается» с системой.

7.1. Модуль генетических алгоритмов GA

7.1.1. Общие сведения

Модуль GA выполнять оптимизацию анализируемых структур посредством ГА. Предполагается, что пользователь знаком с ГА, принципами их работы и терминологией. Сначала необходимо загрузить модуль GA (листинг 7.1).

```
INCLUDE "UTIL"
CHECK_CORE_VERSION 6000
INCLUDE "GA"
```

Листинг 7.1. Подготовка к использованию модуля GA

7.1.2. Простой пример оптимизации

Команда GA_MIN находит минимум функции. Параметры:

1) размер популяции;
2) число поколений;
3) коэффициент мутации;
4) коэффициент кроссовера;
5) число аргументов функции качества num_vars;
6) от 6 до 5+2·num_vars — пары значений нижнего и верхнего пределов изменения аргументов функции качества;
7) последний параметр – функция качества.

При этом внутри функции качества существуют команды GA_PARAM_1, GA_PARAM_2 и так далее до GA_PARAM_ с числом аргументов функции качества, возвращающие параметры функции качества, для которых необходимо рассчитать ее значение.

Команда GET_BEST_GA_RESULT возвращает найденное ГА оптимальное значение функции качества.

Команда GET_BEST_GA_PARAMETER возвращает значения аргументов функции качества, при которых ее значение равно оптимальному. Параметр – индекс аргумента функции качества (индексация начинается с нуля и соответствует порядку, в котором аргументы идут при вызове команды GA_MIN).

В листинге 7.2 и на рис. 7.1 показан пример поиска минимума функции $функция\_качества\ (x,y)=x^2+y^2$. Сама функция качества создаётся динамически командой CREATE_KEYWORD.

```
CREATE_KEYWORD "quality_function"
   SET "x" GA_PARAM_1
   SET "y" GA_PARAM_2
   SET "qf_result" PLUS MUL x x MUL y y
END_CREATE_KEYWORD qf_result

GA_MIN 30 5 0.1 0.5 2 -10. 10. -10. 10. "quality_function"

ECHO FORMAT_STRING std GA_optim_result GET_BEST_GA_RESULT
ECHO FORMAT_STRING std optimized_x GET_BEST_GA_PARAMETER 0
ECHO FORMAT_STRING std optimized_y GET_BEST_GA_PARAMETER 1
```

Листинг 7.2. Пример поиска минимума функции

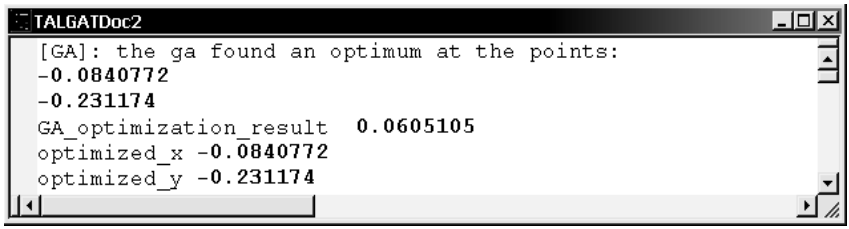

Рис. 7.1. Результаты выполнения скрипта из листинга 7.2

Команда GA_MAX выполняет поиск максимума функции качества. Параметры аналогичны команде GA_MIN. При поиске максимума той же функции, ГА выдает в качестве результата значения, близкие к границам диапазона оптимизации параметров функции – (–10, –10) или (10, 10), что говорит о том, что функция не имеет максимума (рис. 7.2).

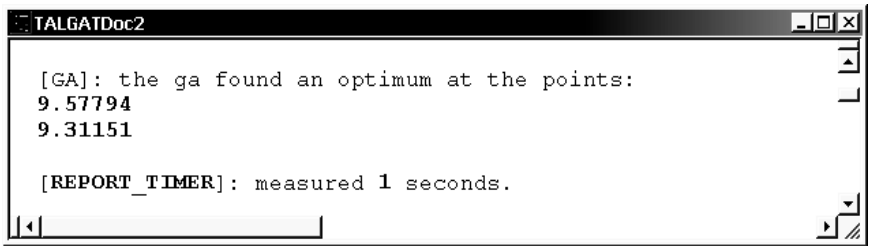

Рис. 7.2. Результаты выполнения скрипта из листинга 7.2

7.1.3. Измерение времени оптимизации

Пример измерения времени оптимизации приведён в листинге 7.3. Пример оптимизации по максимуму функции. Команда REPORT_TIMER в данном случае выведет время оптимизации в секундах. Обратите внимание, что из-за ограниченной ширины страницы данного пособия некоторые строки скрипта были перенесены. Однако в системе TALGAT эти строки не должны содержать переносов.

```
REPORT_TIMER GA_MAX 30 5 0.1 0.5 2 -10. 10. -10. 10.
                "quality_function"
```

Листинг 7.3. Пример оптимизации по максимуму функции

7.1.4. Дополнительные команды

Команда SET_NBITS_PER_NUMBER устанавливает для ГА количество битов, которым кодируется один параметр (по умолчанию 16). Соответственно, GET_NBITS_PER_NUMBER возвращает текущее количество битов.

7.1.5. Оптимизация двумерной конфигурации

Задача: минимизировать разницу (Kc-Kl) для обращенной микрополосковой линии (рис. 7.3) путем изменения параметра hd2 (толщины второго слоя диэлектрика). Исходные данные: число линий – 2, $d = 1$; $w = d$; $t = w$ 0,1; $s = d$; $hd1 = w$ 0,5; $hd2 = w$ 1; $er1 = 2$; $er2 = 5$; $er3 = 1$.

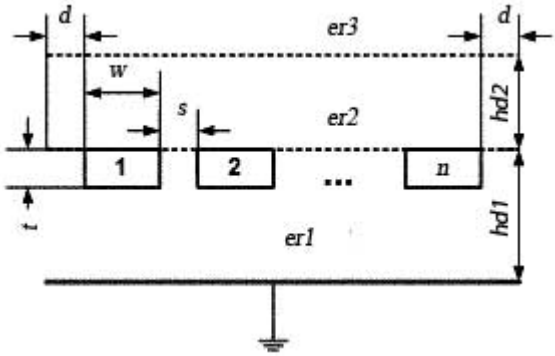

Рис. 7.3. Обращенная микрополосковая линия

В листинге 7.4 приведён скрипт, инициализирующий динамическую команду add_cnd, которая создаёт проводник и часть диэлектрика под ним (рис. 7.4, *а*).

а *б*

Рис. 7.4. Пошаговое создание обращённой микрополосковой линии:
а – результат выполнения двух вызовов add_cnd;
б – результат однократного выполнения qf

```
INCLUDE "MOM2D"
INCLUDE "MATRIX"

SET "num_of_lines" 2

SET "d" 1.
SET "w" d
SET "t" MUL w 0.1
SET "s" d
SET "hd1" MUL w 0.5
SET "hd2" MUL w 1.

SET "er1" 2.
SET "er2" 5.
SET "er3" 1.

SET "subint" 8

SET "si_diel" subint
SET "si_diel_btw" DIV subint 4
SET "si_cndw" DIV subint 4
SET "si_cndt" DIV subint 8

SET_INFINITE_GROUND 1

CREATE_KEYWORD "add_cnd"

DIELECTRIC
   SET_SUBINTERVALS si_diel_btw
   SET_ER_PLUS er1
   SET_ER_MINUS er2
   LINE MINUS x_begin d hd1 x_begin hd1

CONDUCTOR
   SET_ER_PLUS er2
   SET_SUBINTERVALS si_cndw
   LINE x_begin hd1 PLUS x_begin w hd1
   SET_ER_PLUS er1
   SET_SUBINTERVALS si_cndt
   LINETO PLUS x_begin w MINUS hd1 t
   SET_SUBINTERVALS si_cndw
   LINETO x_begin MINUS hd1 t
```

```
   SET_SUBINTERVALS si_cndt
   LINETO x_begin hd1
SET_VARIABLE "x_begin" PLUS x_begin PLUS w s
END_CREATE_KEYWORD
```

Листинг 7.4. Задание исходных параметров и динамической команды для оптимизации двумерной конфигурации

Скрипт из листинга 7.5 инициализирует динамическую функцию качества qf, которая, циклически вызывая add_cnd (число вызовов равно числу линий), полностью достраивает конфигурацию, анализирует структуру и вычисляет разницу Kc-Kl – эта величина и возвращается функцией качества. Затем проводится минимизация с помощью ГА. Для того чтобы нарисовать оптимальную конфигурацию, в функции качества постоянно сохраняются новые конфигурации с (Kc-Kl) меньшим, чем начальное значение. Обратите внимание, что из-за ограниченной ширины страницы данного пособия некоторые строки скрипта были перенесены. Однако в системе TALGAT эти строки не должны содержать переносов. Результат работы приведен на рис. 7.5.

```
CREATE_KEYWORD "qf"

SET "hd2" MUL w GA_PARAM_1
SET "hd2" PLUS hd1 hd2

SET "x_begin" d

CYCLE num_of_lines add_cnd

DIELECTRIC
   SET_SUBINTERVALS si_diel_btw
   SET_ER_PLUS er1
   SET_ER_MINUS er2
   LINE MINUS x_begin s hd1 PLUS MINUS x_begin s d hd1
   SET_ER_PLUS er2
   SET_ER_MINUS er3
   SET_SUBINTERVALS si_diel
   LINE 0. hd2 PLUS MINUS x_begin s d hd2

SET_VARIABLE "conf" GET_CONFIGURATION_2D
SET_VARIABLE "smn" SMN_C conf
SET_VARIABLE "cm" CALCULATE_C smn conf
SET_VARIABLE "smn_l0" SMN_L0 conf
SET_VARIABLE "cm_l0" CALCULATE_L0 smn_l0 conf

SET "kc" DIV MINUS 0. GET_MATRIX_VALUE cm 0 1
          GET_MATRIX_VALUE cm 0 0
```

```
SET "kl" DIV GET_MATRIX_VALUE cm_l0 0 1
        GET_MATRIX_VALUE cm_l0 0 0

IF LESS MINUS kc kl save_kc_kl
THEN SET "save_conf" conf
THEN SET "save_kc_kl" MINUS kc kl

END_CREATE_KEYWORD MINUS kc kl

SET "save_kc_kl" 1.e+6

SET "time" REPORT_TIMER GA_MIN 10 10 0.1 0.6
                    1 1.e-6 1. "qf"

ECHO FORMAT_STRING sdtsd hd2/w=
             GET_BEST_GA_PARAMETER 0
             (kc-kl)= GET_BEST_GA_RESULT

ECHO FORMAT_STRING dtdtd time GET_BEST_GA_PARAMETER
             0 GET_BEST_GA_RESULT

DRAW_CONFIGURATION save_conf
```

Листинг 7.5. Задание функции качества и оптимизация двумерной конфигурации с помощью ГА

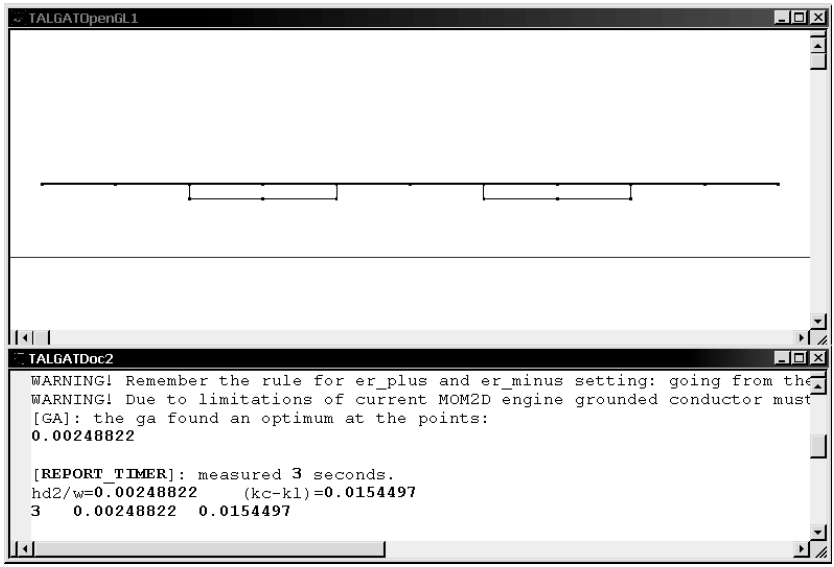

Рис. 7.5. Результат выполнения скрипта из листингов 0.6 и 7.5

В результате оптимизации получили, что оптимальное соотношение hd2/w равно 0,0024882, при этом разница Kc-Kl минимальна и составляет 0,0154497.

7.1.6. Структурная оптимизация

Задача: минимизировать *z*-составляющую напряженности электрического поля в точке (центр источника_*x*, 100, центр источника_*z*) путем изменения количества пассивных диполей в структуре. Исходные данные: начальное число диполей по оси *X* – 5, по оси *Z* – 3, номер диполя с источником питания по оси *X* – 2, по оси *Z* – 1, высота диполей – 5, частота 30 МГц, радиус дальней зоны – 100.

В листинге 7.7 создаются две динамических команды. Первая — do_source, создает на основе заданных параметров диполь с источником питания. Вторая – do_dipole, создает на основе заданных параметров пассивный диполь при условии, что параметр функции качества ГА с соответствующим номером больше 0. Обратите внимание, что из-за ограниченной ширины страницы данного пособия некоторые строки скрипта были перенесены. Однако в системе TALGAT эти строки не должны содержать переносов.

```
INCLUDE "MOMW"

SET "num_dipoles_by_x" 5
SET "num_dipoles_by_z" 3
SET "source_dipole_x" 2
SET "source_dipole_z" 1

SET "wire_height" 5.
SET "btw_wire_x" 5.
SET "btw_wire_z" 5.
SET "num_subsections" 10
SET "btw_wire_z" PLUS btw_wire_z wire_height

SET "PI_method" PIA126_127
SET "f" 30.e+6
SET "far_zone_radius" 100.

SET_SUBSECTIONS_ON_WAVE 1
SET_SUBSECTIONS_MINIMAL num_subsections
SET "source_dipole_subsections" PLUS MUL
          num_subsections 2 1

CLEAR_STRUCTURE

RADIUS 5.e-3

CREATE_KEYWORD "do_source"

SET_SUBSECTIONS num_subsections
SET "x" MUL TO_DOUBLE source_dipole_x btw_wire_x
SET "z" MUL TO_DOUBLE source_dipole_z btw_wire_z
BEGIN x 0. z
END x 0. PLUS z MINUS DIV wire_height 2. DIV
             wire_height 100.
CREATE_WIRE
```

```
BEGIN x 0. PLUS z MINUS DIV wire_height 2. DIV
            wire_height 100.
END x 0. PLUS z PLUS DIV wire_height 2. DIV
          wire_height 100.
EXCITATION (1.,0)
SET "where_calc_E_x" x
SET "where_calc_E_z" PLUS z MINUS DIV wire_height 2.
               DIV wire_height 100.
CREATE_WIRE
EXCITATION (0.,0.)

SET_SUBSECTIONS num_subsections
BEGIN x 0. PLUS z PLUS DIV wire_height 2. DIV
            wire_height 100.
END x 0. PLUS z wire_height
CREATE_WIRE

END_CREATE_KEYWORD

CREATE_KEYWORD "do_dipole"

SET "call_create" 1
BEGIN x 0. z
END x 0. PLUS z wire_height
IF EQU source_dipole_z num_z
THEN IF EQU source_dipole_x num_x
THEN SET "call_create" 0
IF LESS GET_VARIABLE PLUS GA_PARAM_ TO_STRING w_num 0.
THEN SET "call_create" 0
SET "w_num" PLUS w_num 1

IF EQU call_create 1
THEN CREATE_WIRE

SET "x" PLUS x btw_wire_x
SET "num_x" PLUS num_x 1
IF EQU num_x num_dipoles_by_x
THEN SET "num_x" 0
THEN SET "num_z" PLUS num_z 1
THEN SET "x" 0.
THEN SET "z" PLUS z btw_wire_z

END_CREATE_KEYWORD
```

Листинг 7.7. Создание динамических команд do_source и do_dipole

В листинге 7.8 и на рис. 7.6 создается функция качества qf. В ней происходит вызов в цикле динамической команды do_dipole, которая на основе значений параметров функции качества GA_PARAM_n создает или не создает пассивные диполи с номерами *n*. Команда GA_MIN_STRUCTURAL оптимизирует структуру. Параметры:

1) размер популяции;
2) число поколений;
3) коэффициент мутаций;
4) коэффициент кроссовера;
5) число аргументов num_vars (число элементов структуры, которые могут создаваться или не создаваться);
6) функция качества.

Обратите внимание, что из-за ограниченной ширины страницы данного пособия некоторые строки скрипта были перенесены. Однако в системе TALGAT эти строки не должны содержать переносов.

```
CREATE_KEYWORD "qf"

CLEAR_STRUCTURE

SET "num_x" 0
SET "num_z" 0
SET "x" 0.
SET "z" 0.
SET "w_num" 1
CYCLE MUL num_dipoles_by_x num_dipoles_by_z do_dipole
do_source

SET "structure" GET_STRUCTURE
SET "subsec" CREATE_SUBSECTIONS f structure
SET "exc_vec" GET_EXCITATION_VECTOR
SET "imp_m" CALCULATE_IMPEDANCE_MATRIX f PI_method subsec

SET "currents" LU_SOLVE LU_FACT imp_m exc_vec
SET "e" ABS GET_MATRIX_VALUE CALCULATE_E_FAR_ZONE f
     currents subsec where_calc_E_x
          far_zone_radius where_calc_E_z 0 2

IF LESS e save_e
THEN SET "save_e" e
THEN ECHO FORMAT_STRING sbd minimized save_e
THEN SET "save_subsec" subsec
THEN SET "save_currents" currents

END_CREATE_KEYWORD e

SET "pop_size" 5
SET "gen_num" 3

REPORT_TIMER GA_MIN_STRUCTURAL pop_size gen_num
               0.6 MUL num_dipoles_by_x
               0.1 num_dipoles_by_z "qf"
ECHO GET_BEST_GA_RESULT

DRAW_CURRENTS save_subsec save_currents 0
```

Листинг 7.8. Задание функции качества и структурная оптимизация

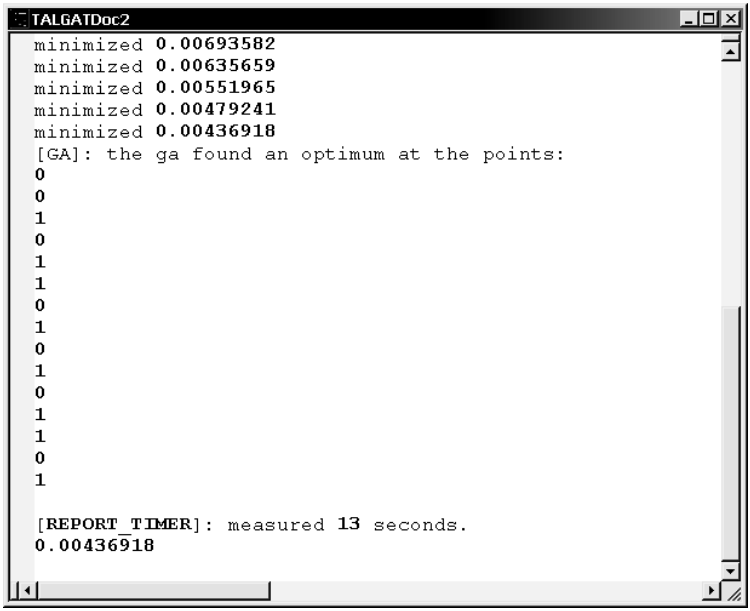

Рис. 7.6. Результат выполнения скрипта из листингов 7.8 и 7.8

В результате оптимизации получены следующие результаты: минимальное значение *z*-составляющей напряженности электрического поля в заданной точке составляет 0,00436918 при расположении диполей, приведённом на рис. 7.7.

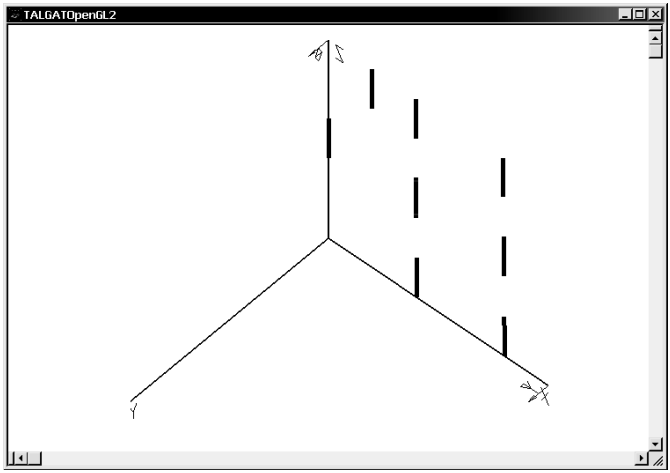

Рис. 7.7. Полученная в результате оптимизации структура

7.2. Пример 1

Рассмотрим пример использования ГА для оптимизации функции двух переменных. За основу возьмем функцию вида:

$$y(x_1,x_2)=\frac{\sin(x_1-x_{01})}{x_1-x_{01}}\cdot\frac{\sin(x_2-x_{02})}{x_2-x_{02}},$$

где x_1, x_2 – аргументы оптимизируемой функции; x_{10}, x_{20} – значения, при которых функция будет иметь максимум.

Таким образом:

$$y(x_1,x_2)=\left|\frac{\sin(\pi\cdot(x_1-3))}{\pi\cdot(x_1-3)}\right|\cdot\left|\frac{\sin(\pi\cdot(x_2-3))}{\pi\cdot(x_2-3)}\right|,$$

функция $y(x_1,x_2)$ будет иметь максимум при $x_1=3$, $x_2=3$. График функции представлен на рис. 7.8.

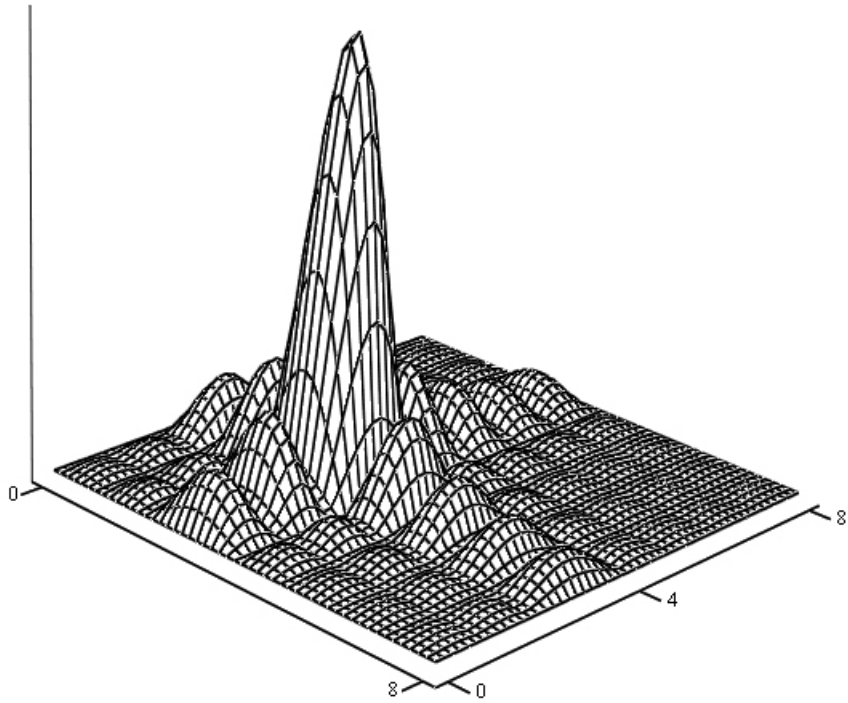

Рис. 7.8. График функции $y(x_1,x_2)=\left|\frac{\sin(\pi\cdot(x_1-3))}{\pi\cdot(x_1-3)}\right|\cdot\left|\frac{\sin(\pi\cdot(x_2-3))}{\pi\cdot(x_2-3)}\right|$

Используем систему TALGAT для поиска максимума данной функции.

```
    INCLUDE "UTIL"
  INCLUDE "GA"
  INCLUDE "INFIX"
  SET_VARIABLE "my_x1" 3.
  SET_VARIABLE "my_x2" 3.
  CREATE_KEYWORD "quality_function"
  SET_INFIX_VARIABLE "x1" GA_PARAM_1
    SET_INFIX_VARIABLE "x2" GA_PARAM_2
    SET_INFIX_VARIABLE my_x01 my_x1
    SET_INFIX_VARIABLE my_x02 my_x2
  SET_VARIABLE "my_y1" INFIX abs(sin(pi*(x1-
my_x01))/(pi*(x1-my_x01)))
    SET_VARIABLE "my_y2" INFIX abs(sin(pi*(x2-
my_x02))/(pi*(x2-my_x02)))
    SET_VARIABLE "qf_result" MUL my_y1 my_y2
     END_CREATE_KEYWORD qf_result
  GASTART_MAX 0 0 0 30 100 0.01 0.05  2 0. 30. 0. 30.
"quality_function"
  ECHO GET_BEST_GA_PARAMETER 0
  ECHO GET_BEST_GA_PARAMETER 1
```

Листинг 7.9. Пример поиска максимума функции двух переменых

Как видно из листинга 7.9 значения x_{10}, x_{20}, равные 3, хранятся в переменных `my_x1`, `my_x2`. Сама функция $y(x_1,x_2)$ представлена в виде произведения двух функций `my_y1`, `my_y2`. Команда `GASTART_MAX` схожа с командой `GA_MAX`, но имеет следующие параметры:

1. Модель ГА.
2. Стратегия отбора.
3. Стратегия формирования нового поколения.
4. Размер популяции.
5. Число поколений.
6. Коэффициент мутации.
7. Коэффициент кроссовера.
8. Число аргументов функции качества num_vars.
9. От 9 до 8+2·num_vars — пары значений нижнего и верхнего пределов изменения аргументов функции качества.
10. Последний параметр – функция качества.

Стоит отметить, что использование просто команды `GA_MAX` (а также `GA_MIN`, `GA_MAX_STRUCTURAL`, `GA_MIN_STRUCTURAL`) означает, что по умолчанию были выбраны следующие параметры:

1. GASimpleGA

2. GASigmaTruncationScaling
3. GARouletteWheelSelector

Команда GASTART_MAX является расширенным вариантом GA_MAX и рассчитана на более опытных пользователей. Результат выполнения листинга 7.9 представлен на рис. 7.9.

Рис. 7.9. Результаты выполнения скрипта из листинга 1

Использование первых трех параметров команды GASTART_MAX представлено в табл. 7.1.

Таблица 7.1

Параметры команды GASTART_MAX

Модель ГА	Стратегия отбора	Стратегия формирования нового поколения
0 – GASimpleGA	0 – GARankSelector	0 – GANoScaling
1 – GASteadyStateGA	1 – GARouletteWheelSelector	1 – GALinearScaling
2 – GAIncrementalGA	2 – GATournamentSelector	2 – GASigmaTruncationScaling
3 – GADemeGA	3 – GAUniformSelector	3 – GAPowerLawScaling
	4 – GASRSSelector	4 – GASharing
	5 – GADSSelector	

7.3. Пример 2

Рассмотрим более сложный пример использования ГА. Исследуем функцию вида:

$$y(x_1,x_2)=\sin(0.1\cdot x_1)\cdot\sin(x_1)\cdot\sin(0.1\cdot x_2)\cdot\sin(x_2),$$

где x_1, x_2 – аргументы оптимизируемой функции; функция $y(x_1,x_2)$ будет иметь максимум при $x_1=14{,}15$,; $x_2=14{,}15$. График функции представлен на рис. 7.10.

Используем систему TALGAT для поиска максимума данной функции.

```
INCLUDE "UTIL"
INCLUDE "GA"
INCLUDE "INFIX"
CREATE_KEYWORD "quality_function"
SET_INFIX_VARIABLE "x1" GA_PARAM_1
SET_INFIX_VARIABLE "x2" GA_PARAM_2
SET_INFIX_VARIABLE my_x01 my_x1
SET_INFIX_VARIABLE my_x02 my_x2
```

```
   SET_VARIABLE "my_y1" INFIX sin(0.1*x1)*sin(x1)
   SET_VARIABLE "my_y2" INFIX sin(0.1*x2)*sin(x2)
   SET_VARIABLE "qf_result" MUL my_y1 my_y2
      END_CREATE_KEYWORD qf_result
   GASTART_MAX 0 0 3 30 1000 0.01 0.05  2 0. 30. 0. 30.
"quality_function"
   ECHO GET_BEST_GA_PARAMETER 0
   ECHO GET_BEST_GA_PARAMETER 1
```

Листинг 7.10. Пример поиска максимума функции двух переменных

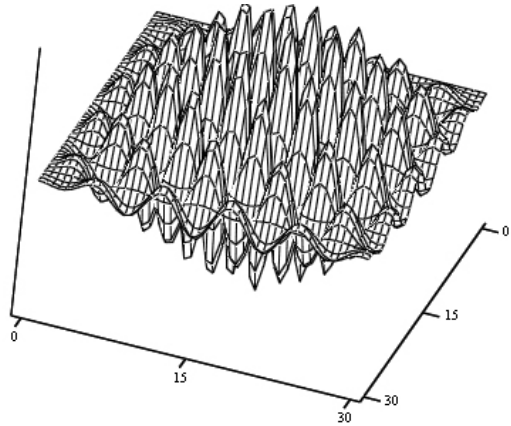

Рис. 7.10. График функции $y(x_1,x_2)=\sin(0{,}1\cdot x_1)\cdot\sin(x_1)\cdot\sin(0{,}1\cdot x_2)\cdot\sin(x_2)$

Так как функция является довольно сложной, увеличили количество поколений до 1000. Результат выполнения листинга 7.10 представлен на рис. 7.11

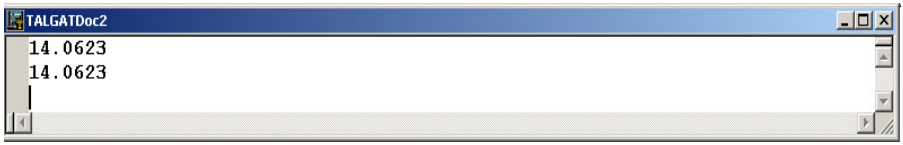

Рис. 7.11. Результаты выполнения скрипта из листинга 7.11

8. Литература

1. http://qai.narod.ru

2. Технология прямого поиска при решении задач прикладной математики / В.А. Архипов, С.С. Бондарчук, И.Г. Боровской, А.А. Шелупанов // Вычислительные технологии. - 1995. - Т. 4, № 10. - С. 19.

3. Matthew Wall GAlib: A C++ Library of Genetic Algorithm Components.

Учебное издание

Тимур Тальгатович Газизов
Александр Олегович Мелкозеров

ОПТИМИЗАЦИЯ ГЕНЕТИЧЕСКИМИ АЛГОРИТМАМИ

Учебное пособие
для студентов специальностей и направлений
10.03.01 – «Информационная безопасность»,
10.05.02 – «Информационная безопасность
телекоммуникационных систем»,
10.05.03 – «Информационная безопасность
автоматизированных систем»,
10.05.04 – «Информационно-аналитические системы безопасности»
09.03.02 – «Информационные системы и технологии»

Верстка – В.М. Бочкаревой
Текст дан в авторской редакции, без корректуры

Издательство «В-Спектр»
Подписано к печати 15.11.2017.
Формат $60\times84^{1}/_{16}$. Печать трафаретная.
Печ. л. 2. Тираж 150 экз. Заказ 39.

Тираж отпечатан ИП Бочкаревой В.М.
ИНН 701701817754
634055, г. Томск, пр. Академический, 13-24, тел. 49-09-91.
E-mail: bvm@sibmail.com

Для заметок