Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Уральский федеральный университет**

**Имени первого Президента России Б.Н. Ельцина»**

**Институт математики и компьютерных наук**

**Кафедра алгебры и дискретной математики**

**Автоматизация процесса тестирования коммуникационного оборудования на основе генетических алгоритмов**

|  |  |
| --- | --- |
| Допустить к защите: | Дипломная работа  студента 6 курса  Перевощикова И.В.  Научный руководитель**:**  Доцент кафедры алгебры и дискретной математики ИМКН УрФУ,  кандидат технических наук  Синадский Н.И. |

Екатеринбург

2015

# РЕФЕРАТ

Перевощиков И.В. Автоматизация процесса тестирования коммуникационного оборудования на основе генетических алгоритмов, дипломная работа: **TODO**

Ключевые слова: СИНТЕЗ ТРАФИКА, ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, ТЕСТИРОВАНИЕ СЕТЕВОГО ОБОРУДОВАНИЯ

Объект исследования – эволюционные подходы к процессу тестирования коммуникационного оборудования на предмет выявления пороговых значений характеристик оборудования.

Цель работы – разработка и реализация программного комплекса на базе генетического алгоритма для автоматического тестирования сетевого оборудования.

В процессе работы проводилось исследование работы генетических алгоритмов и их применения в тестировании коммуникационного оборудования.

В результате работы был разработан программный код, реализующий комплексное тестирование сетевого оборудования с использованием автоматически сгенерированного трафика, близкого по своим характеристикам к реальному.

Результаты работы могут применяться разработчиками сетевого оборудования для автоматического тестирования на предмет выявления условий, являющихся для данного оборудования экстремальными.

# Содержание

[РЕФЕРАТ 2](#_Toc409960150)

[Содержание 3](#_Toc409960151)

[Введение 4](#_Toc409960152)

[Глава 1. Используемое программное обеспечение и библиотеки 7](#_Toc409960153)

[Заключение 8](#_Toc409960154)

[Список использованной литературы 9](#_Toc409960155)

[Приложение 10](#_Toc409960156)

# Введение

Целью данной работы является автоматизация исследования коммуникационного оборудования на выявление таких конфигураций сети, которые будут являться для тестируемого сетевого оборудования «экстремальными».

Другими словами, требуется разработать программный комплекс, автоматически оптимизирующий конфигурацию сети, в которой установлено тестируемое оборудование, таким образом, чтобы привести некоторые значимые для исследования характеристики данного оборудования к «крайним» (пороговым) значениям. Так как поле возможных вариантов сетей охватить методами грубой силы не представляется возможным из-за временных ограничений, было решено воспользоваться средствами генетического алгоритма.

Тема генетических алгоритмов исследована довольно широко, и генетические алгоритмы нашли свое применение в таких областях науки как

* Экстремальные задачи (нахождение точек минимума и минимума),
* Задачи о кратчайшем пути,
* Задачи компоновки,
* Составление расписаний,
* Аппроксимация функций,
* Отбор (фильтрация) входных данных,
* Настройка искусственной нейронной сети,
* Моделирование искусственной жизни (Artificial life systems),
* Биоинформатика (свертывание белков и РНК),
* Игровые стратегии,
* Нелинейная фильтрация,
* Развивающиеся агенты/машины (Evolvable agents/machines).

Данная работа является попыткой применить генетический алгоритм в автоматизации тестирования сетевого оборудования. С этой целью был разработан программный комплекс, состоящий из трех частей:

1. Реализация математической модель конфигурации сети, в которую входят подсети, узлы в этих подсетях и «потоки» между узлами модели, а также принимающие участие в эволюции функции распределения вероятности принятия какой либо характеристикой потока какого-либо значения из заданного отрезка.
2. Непосредственная реализация генетического алгоритма, который производит формирование популяции, скрещивание и отбор. «Особью» в данном алгоритме является некоторая конфигурация сети, представленная при помощи математической модели.
3. Модуль, отвечающий за тестирование некоторой конфигурации сети, полученной из генетического алгоритма, возвращающий для каждой конфигурации некоторую оценку, необходимую для произведения дальнейшего отбора генетическим алгоритмом.

Общая схема взаимодействия модулей представлена на рисунке 1.



Рис.

В качестве языка программирования в данной работе был выбран Python, как язык, наиболее эффективный при реализации сетевых взаимодействий с достаточной степенью гибкости настройки последних.

В качестве экспериментального объекта исследований – типичный представитель семейства «домашних» маршрутизаторов – маршрутизатор производства компании D-Link, модель DIR-300.

# Глава 1. Используемое программное обеспечение и библиотеки

Разработанный программный комплекс может быть развернут на компьютере под управлением операционной системы семейства Linux, с установленным Python 2.7, библиотеками Pyevolve и Scapy.

Разработка велась на компьютере с операционной системой Ubuntu 14.04 LTS, с установленными Python 2.7.6, Pyevolve 0.6rc и Scapy 2.1.1.

1.1 Библиотека Pyevolve

На данный момент существует множество способов реализации генетических алгоритмов. Самым быстрым и удобным способом для языка Python является библиотека Pyevolve [4].

Данная библиотека был разработан специально для создания полноценных генетических алгоритмов. Сама библиотека – лишь каркас, реализующий общую логику работы генетического алгоритма, но все же сильно упрощающий жизнь разработчика.

Для быстрого создания работающего генетического алгоритма авторы библиотеки предлагают реализовать только функциональную составляющую генетического алгоритма, как, например, программное представление особи или алгоритм скрещивания.

Всю общую и неизменную логику работы генетического алгоритма выполняет сам библиотека: генерирует популяции, производит отбор, скрещивания, мутации, проверяет критерий остановки эволюции и т.п.

Pyevolve также предоставляет возможности логирования, выгрузки данных в различные типы хранилищ (база данных, электронная таблица), взаимодействия с текущей популяцией в интерактивном режиме.

Простота реализации генетического алгоритма хорошо видна из нижеследующего примера, в котором реализован алгоритм поиска двоичного вектора длины 20 с наибольшим количеством нулей.

1. **def** eval\_func(chromosome):
2. score = 0.0
3. **for** value **in** chromosome:
4. **if** value==0:
5. score += 1
6. **return** score
8. genome = G1DList.G1DList(20)
9. genome.evaluator.set(eval\_func)
10. ga = GSimpleGA.GSimpleGA(genome)
11. ga.evolve(freq\_stats=10)
12. **print** ga.bestIndividual()

Листинг 1. (Пример простейшего ГА)

В строках 1-6 описывается функция, ставящая в соответствие каждой особи некое число или оценку «приспособленности». В строках 8-12 производится несложная настройка параметров работы генетического алгоритма.

Таким образом, библиотека Pyevolve предоставляет разработчику возможность сконцентрироваться на своей конкретной задаче, предоставляя удобный интерфейс для быстрой реализации требуемого генетического алгоритма и возлагая на себя всю «сервисную» работу.

1.2 Утилита Scapy

Scapy – утилита, написанная на языке Python, позволяющая пользователю посылать, просматривать, анализировать и генерировать сетевые пакеты.  В отличие от аналогичных утилит, утилита Scapy не ограничена только теми протоколами, пакеты которых она может генерировать. Фактически, она позволяет создавать любые сетевые пакеты, что обеспечивает требуемую для решения поставленной задачи гибкость.

Типичные примеры использования утилиты:

* Сканирование,
* Трассировка маршрута,
* Юнит-тестирование всевозможных сетевых функций,
* Различные виды сетевых атак.

К примеру, классический ICMP-ping с выводом всех ответивших на запрос узлов, с помощью Scapy реализуется всего в две строки:

1. ans,unans=sr(IP(dst="192.168.1.1-254")/ICMP())
2. ans.summary(**lambda** (s,r): r.sprintf("%IP.src% is alive") )

Листинг 2. (Пример реализации сетевого взаимодействия)

В данной работе утилита Scapy использовалась для следующих целей:

1. Создание TCP-пакетов,
2. Создание UDP-пакетов,
3. Создание ICMP-пакетов,
4. Создание файла-выгрузки в формате pcap, содержащего сгенерированные моделью сети пакеты.

Ниже приведены примеры использования утилиты Scapy в программном коде данной работы. Так, например, создается TCP – пакет:

1. l3 = IP(src=ip1, dst=ip2)  # создание IP-пакета
2. l4 = TCP(sport=self.port1, dport=self.port2)  # создание TCP-
3. пакета
4. ...
5. l34 = l3 / l4 # Инкапсуляция TCP-пакета в IP-пакет
6. ...
7. l5 = self.generate\_l5(params['flp'].random())  # генерирование
8. полезной нагрузки
9. ...
10. l34[TCP].flags |= flags\_on  # установка флагов TCP
11. l34[TCP].flags &= ~flags\_off
12. l34[TCP].seq = seq
13. l34[TCP].ack = ack
14. l34[IP].ttl = params['fttl'].random()  # установка времени жизни
15. пакета
16. p = l34 / l5  # помещение полезной нагрузки в готовый пакет

Листинг 3. (Пример создания TCP - пакета)

Аналогичным образом создается UDP – пакет:

1. l3 = IP(src=ip1, dst=ip2)
2. l4 = UDP(sport=self.port1, dport=self.port2)  # создание UDP –
3. пакета
4. l34 = l3 / l4
5. l5 = self.generate\_l5(params['flp'].random())
6. l34[IP].ttl = params['fttl'].random()
7. p = l34 / l5
8. p.time = t   # установка времени отправки пакета

Листинг 4. (Пример создания UDP - пакета)

Создание ICMP – пакета происходит аналогично вышеприведенным примерам.

А вот так происходит сохранение набора сетевых пакетов:

1. wrpcap("temp.cap",pkts)  # pkts – предварительно сгенерированный
2. массив пакетов

Таким образом, утилита Scapy, является наиболее подходящим инструментом, для создания сетевых пакетов, параметры которых генерируются в процессе работы генетического алгоритма, и полностью избавляет от необходимости самостоятельной реализации модуля создания сетевого пакета с многочисленными варьируемыми параметрами.

# Заключение

# Список использованной литературы

1. Панченко, Т. В. Генетические алгоритмы [Текст]: учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. — Астрахань: Издательский дом «Астраханский университет», 2007. — 87 [3] с.
2. Лутц М. Изучаем Python, 4-е издание. / Пер. с англ. – Санкт-Петербург: Символ-Плюс, 2011. – 1280 с., ил.
3. Сегаран Т. Программируем коллективный разум. / Пер. с англ. – Санкт-Петербург: Символ-Плюс, 2008. – 368 с., ил.
4. Christian S. Perone. Pyevolve Documentation Release 0.5 [Электронный ресурс] – Режим доступа: http://pyevolve.sourceforge.net/latex\_0\_5.pdf, свободный.
5. Philippe Biondi and the Scapy community. Scapy v2.1.1-dev documentation [Электронный ресурс] – Режим доступа: <http://www.secdev.org/projects/scapy/doc/>, свободный

# Приложение