УДК

ШИФР ЦЕЗАРЯ И ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Протько М.А., студент гр.050502

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Борисенко О.Ф. – канд. физ.-мат. наук, доцент

Аннотация: Используя наличие современных технологий и нововведений возможно внедрить новые подходы к решению таких задач, как поиск динамически развивающейся системы шифрования. Данная работа содержит теоретический материал необходимый для дальнейшего развития данной идеи.

Ключевые слова: генетический алгоритм, подстановочные шифры, моноалфавитные и полиалфавитные шифры, целевая функция, кроссинговер, турнирный отбор

Введение

С распространением глобальной передачи информации необходимо было придумать способы ее сокрытия от нежелательных лиц. Начиная от сдвиговых и подстановочных шифров и заканчивая хэш-функциями и эллиптическими кривыми. Методы сокрытия информации и ее преобразования, а если конкретнее, стеганография и криптография, развивались постепенно с ростом вычислительных возможностей и наших знаний в теории чисел. Казалось бы, давно уже должна была исчезнуть возможность нахождения задачи, которую невозможно вычислить. Но такие задачи все еще существуют. Они относятся к задачам с отсутствием четкого решения или с необходимостью огромнейшего перебора (задача коммивояжера, задача поиска больших простых чисел, задача поиска натуральных делителей и т.д.).

Но все же, для некоторых из этих задач был найден свой особый подход. В последнее время стали распространятся нейронные сети, но в моей работе более интересен их прародитель, а именно, генетические алгоритмы.

Генетические алгоритмы берут многое из области биологи. Используя стратегию, вдохновленную развитием и эволюцией живых существ, решаются самые разнообразные проблемы вычислительной техники: используя этот алгоритм, возможно находить оптимальные пути передвижения из точки А в точку Б, причем первоначальным участникам передвижения не будет ничего известно о местоположении цели («муравьиный» алгоритм, применение – выбор путей передачи пакетов между маршрутизаторами), также возможно использовать генетический алгоритм для нахождения решения функций, где посредством сравнений с желаемых результатом, находятся желаемые решения, ну, и разумеется, генетические алгоритмы используются для моделирования эволюционных процессов и создания искусственного интеллекта в играх, пусть даже и не столь впечатляющего.

Поскольку на современном этапе развития криптографии все очевидные решения уже были использованы, к тому же, теперь недостаточно просто шифровать что-либо каким-то конкретным хорошо показавшим себя способом, теперь необходимо делать это динамически (т.е ключи и порядок их использования непостоянны) , желательно используя несколько тактик (в протоколах безопасности используются комбинации нескольких шифров, отдельные для ключа, и совершенно другие для сообщения), поэтому необходимо использовать больше изобретательности.

Со всем вышеперечисленным на ум приходит идея: а что может нам дать в этой ситуации генетический алгоритм? Какие результаты, желательно с возможностью практического применения он может нам дать?

Возможно ли создать систему, имитирующую основные факторы, приведшие к созданию шифров таким образом, чтобы она, используя генетический алгоритм, повторила стратегию, используемую человеком?

В данной работе я попытаюсь ответить на эти вопросы.

Генетический алгоритм. Теория.

Генетические алгоритмы — это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде и основные понятия линейной алгебры.

Основной идеей генетических алгоритмов является организация «борьбы за существование» и «естественного отбора» среди пробных решений. Поскольку генетические алгоритмы используют биологические аналогии, то и применяющаяся терминология напоминает биологическую. Так, одно пробное решение, записанное в двоичной (или десятичной) форме, называться особью или хромосомой, а набор всех пробных решений – популяцией.

Генетический алгоритм чаще всего состоит из следующих этапов:

- Отбор
- Скрещивание /мутации
- Выбор индивидуума с максимальной приспособленностью
- Поддержание популяции

Логические связи между каждым этапом схематически изображены на рис.1.1

Рисунок 1.1 – Схема генетического алгоритма

Рассмотрим каждый этап подробнее:

Вычисление приспособленности для каждого индивидуума:

Чаще всего вычисляется с помощью целевой функции (fitness function). Чем меньше значение целевой функции, тем более приспособленной является особь, т.е. пробное решение, использовавшееся в качестве аргумента целевой функции.

К примеру, если необходимо найти минимум некой функции, целевая функция будет выбирать ту особь, значение вывода которой, подставленное в аргумент даст минимальное значение.

Есть случаи, когда можно обойтись без целевой функции, если составить условия существования таким образом, чтобы соревнование между особями происходило само собой. Можно использовать ограничение по времени жизни и количеству особей в системе, использовать принцип накопления энергий (кто имеет больше всего запасов, тот способен образовывать лучшее поколение) и т.д.

Скрещивание (кроссинговер):

Процесс размножения (рекомбинация), заключается в обмене участками хромосом между родителями. Например, пусть скрещиваются две хромосомы 111111 и 000000. Определяем случайным образом точку разрыва хромосомы, пусть, это будет 3: 111|111 000|000. Теперь хромосомы обмениваются частями, стоящими после точки разрыва, и образуют двух новых потомков: 111000 и 000111.

Иными словами, геномы двух и более особей меняются какими-либо частями между собой.

Мутации:

Случайные изменения генома особи в одном и более местах. Вероятность их возникновения регулируется самостоятельно или распределяется случайным образом.

Мутации способны улучшить или ухудшить приспособленность особи-потомка. Задача отсеивания последних приходится на вышеописанный первый пункт.

Процесс размножения:

На основе исходной популяции создается новая, с применением вышеописанных стратегий. В случае использования кроссинговера, необходимо определять родителей различными способами (по понятию схожести, наибольшей приспособленности и т. д.).

Существует несколько способов отбора в популяцию, это турнирный отбор и метод рулетки.

При турнирном отборе (tournament selection) из популяции, содержащей N особей, выбираются случайным образом t особей, и лучшая из них особь записывается в промежуточный массив. Эта операция повторяется N раз. Особи в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, часто равен 2. В этом случае говорят о двоичном (парном) турнире. Вообще же t называют численностью турнира. Преимуществом данного способа является то, что он не требует дополнительных вычислений.

В методе рулетки (roulette-wheel selection) особи отбираются с помощью N «запусков» рулетки, где N — размер популяции. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-го сектора пропорционален вероятности попадания в новую популяцию P(i), вычисляемой по формуле:

$$P(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)}, \qquad 1)$$

где f(i) — пригодность i-й особи. Ожидаемое число копий i-ой хромосомыпосле оператора рулетки определяются по формуле $N_i = P(i)N$.

При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью.

Другие способы отбора можно получить на основе модификации выше приведенных. Так, например, в рулеточном отборе можно изменить формулу для вероятности попадания особи в новую популяцию.

Простейшие шифры

Прежде чем приступить к рассмотрению принципов работы шифров, необходимо ознакомится с некоторыми общепринятыми терминами:

Шифром будем называть систему или алгоритм, трансформирующий произвольное сообщение в такую форму, которую не сможет прочитать никто кроме тех, кому это сообщение предназначено.

Ключом будем называть такую символьную последовательность, знания которой позволяют привести шифрованный текст в его изначальный вид. Причем, чаще всего считается, что именно ключ – гарантия невозможности расшифровки текста, даже если алгоритм шифрования известен.

Под *шифрованием* будем понимать процесс преобразования открытого текста в шифрованный текст. Соответственно, *дешифрование* – это процесс, обратный шифрованию, преобразующий зашифрованный текст в открытый.

Для дальнейшей работы необходимо будет рассмотреть следующие виды шифрования:

Шифр Цезаря:

Самый старейший из шифров, в нынешних реалиях, увы, не используемый из-за своей простоты, а, следовательно, и с весьма наглядной расшифровкой методом грубой силы (перебором всех значений ключа).

В шифре Цезаря каждая буква в открытом тексте смещается на три позиции так, что буква «A», например, замещается буквой «D». Буква «B» замещается буквой «E» и так далее. Конец алфавита замыкается на его начало так, что буква «X» замещается буквой «A», а буква «Y» – буквой «B», «Z» – «C». В шифре Цезаря каждая буква смещается на 3 позиции, однако в более широком смысле этот шифр можно рассматривать, как смещение на некое целое число позиций (k), причем число k будет играть роль ключа.

На рисунке 2.1 схематически изображено преобразование, выполняемое шифром Цезаря.

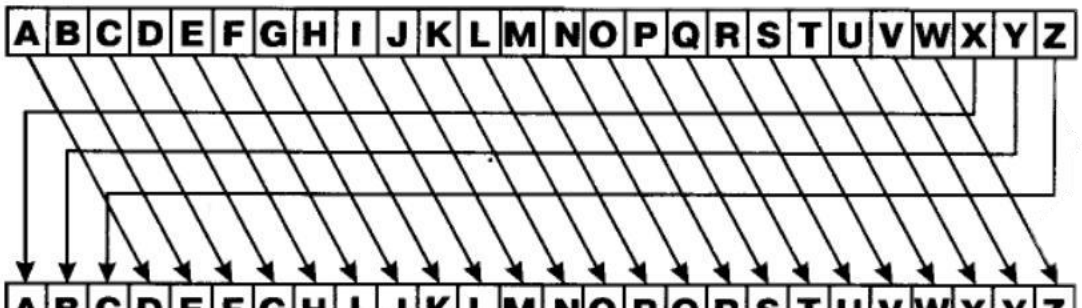


Рисунок 2.1 – Шифр Цезаря, схематическое преобразование

В итоге, получим:

Ключ:	3
Открытый текст:	Hello
Зашифрованный текст:	Khoor

Сразу становится заметно, что метод частотного анализа также будет весьма полезен в случае необходимости расшифровки. Чем длиннее будет сообщение, тем больше вероятность потери секретности.

Математическое определение шифра Цезаря (сдвигового шифра):

При произвольном ключе k, где K ϵ Z_{26}, где Z_{26} -множество целых чисел,0 ≤ k ≤ 25, и произвольном открытом тексте p в виде кортежа, где p = (p_1, p_2, p_3, ..., p_m) и p_i ϵ Z_{26} для 0 ≤ i ≤ m, результирующий шифрованный текст c будет представлен кортежем c = (c_1, c_2, c_3, ..., c_m) и c_i ϵ Z_{26} для 0 ≤ i ≤ m.

При этом функция шифрования для $E_k(p)$ для сдвигового шифра определяется следующим образом:

$c_i = E_k(p_i) = p_i + k(\text{mod } 26)$ для 0 ≤ i ≤ m . А функция дешифрования $D_k(c)$ определяется, как $p_i = D_k(c_i) = c_i - k(\text{mod } 26)$ для 0 ≤ i ≤ m.

Еще одна особенность шифра Цезаря, это бессмысленность многократной шифровки. При повторении процесса шифрования, два ключа будут представлять не что иное, как ключ, представляющий сумму смещений первых двух, что не дает никакого практического смысла.

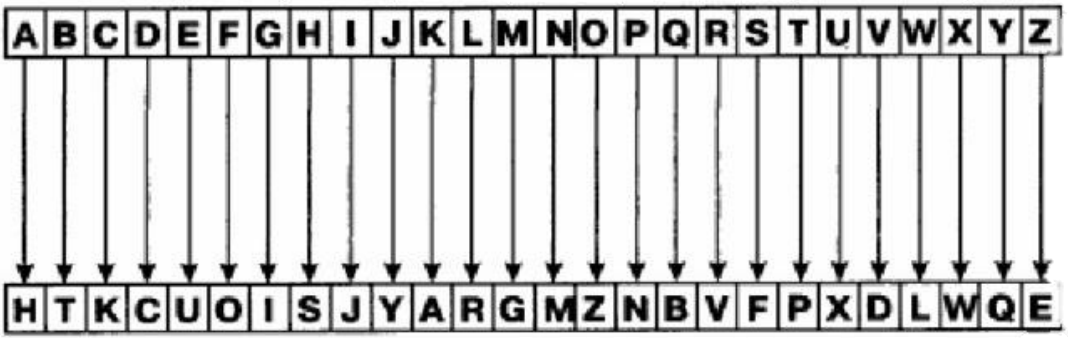
Далее следует ознакомится с немного более совершенными классическими шифрами, развивающими идею смещения относительно ключа.

Простой подстановочный шифр

В подстановочном шифре каждый символ заменяется заранее определенным символом подстановочного алфавита, что относит его, как и шифр Цезаря, к моноалфавитным подстановочным шифрам. Это означает, что существует однозначное соответствие между символами в открытом тексте и символами в тексте зашифрованном. Такое свойство шифра делает его также уязвимым для атаки, основанной на частотном анализе.

Схематический он изображен на рисунке 2.2.

Рисунок 2.2 – Подстановочный шифр, схематическое преобразование



Математическое определение подстановочного шифра (шифра простой замены)

Пусть K есть множество всех перестановок элементов в Z_{26} $K = \{(k_1, k_2, k_3, ..., k_{26}): k_i \in Z_{26}\}$. Выберем произвольный ключ k, где k ϵ K. Выберем произвольный текст p, где k ϵ K, p = (p_1, p_2, ..., p_m), где p_i ϵ Z_{26} для 1≤ i ≤ m. Пусть результирующий шифрованный текст c есть: c = (c_1, c_2, ..., c_m), где c_i ϵ Z_{26} для 1≤ i ≤ m. Далее определим функцию шифрования E_k (p), как $c_i = E_k(p_i) = k[p_i]$ для 1≤ i ≤ m. Функция дешифрования $D_k(c)$ определяется, как $p_i = D_k(c_i) = E_k - 1(c_i)$ для 1≤ i ≤ m.

В итоге, получим:

Ключ:	HTKCUOISJYARGMZNBVFPXDLWQE
Открытый текст:	Hello
Зашифрованный текст:	SURRZ

Иными словами, в подстановочном шифре мы придумываем свой собственный алфавит. Чем больше алфавит языка, который мы используем, тем больше длина ключа. К примеру, в английском алфавите 26 букв, а значит у нас будет 26-значный ключ с 26! = 403 291 461 126 605 635 584 000 000 различными значениями. Но, увы, длина ключа ни насколько не повлияла на сложность расшифровки.

Этот шифр крайне уязвим к частотному анализу, поскольку частота букв остается той же.

Шифр Виженера

Ранее известный как «нераскрываемый шифр», он во многом превзошел своих предшественников.

Шифр Виженера представляет собой полиалфавитный подстановочный шифр. Это означает, что для подстановки используются многие алфавиты, благодаря чему частоты символов в зашифрованном тексте не соответствуют частотам символов в тексте открытом. Следовательно, в отличие от моноалфавитных подстановочных шифров наподобие шифра Цезаря, шифр Виженера не поддается простому частотному анализу.

В сущности, шифр Виженера меняет соответствие между открытыми и зашифрованными символами для каждого очередного символа. Он основывается на таблице, вид которой приведен на рисунке 2.3. Каждая строка этой таблицы – не что иное, как шифр Цезаря, сдвинутый на число позиций, соответствующее позиции в строке. Строка A сдвинута на 0 позиций, строка B – на 1 и так далее.

В шифре Виженера такая таблица используется в сочетании с ключевым словом, при помощи которого шифруется текст.

Для шифрования вы повторяете ключ столько раз, сколько необходимо для достижения длины открытого текста, просто записывая символы под символами открытого текста. Затем вы получаете поочередно каждый символ зашифрованного текста, беря столбец, определенный по символу открытого текста, и пересекая его со строкой, определенной по соответствующему символу ключа. Например, первый символ открытого текста G в сочетании с первым символом ключа P, как видно из рисунка 2.6, дадут столбец и строку, пересекающиеся на символе V, каковой и будет первым символом шифрованного текста. Все последующие символы определяются аналогичным образом.

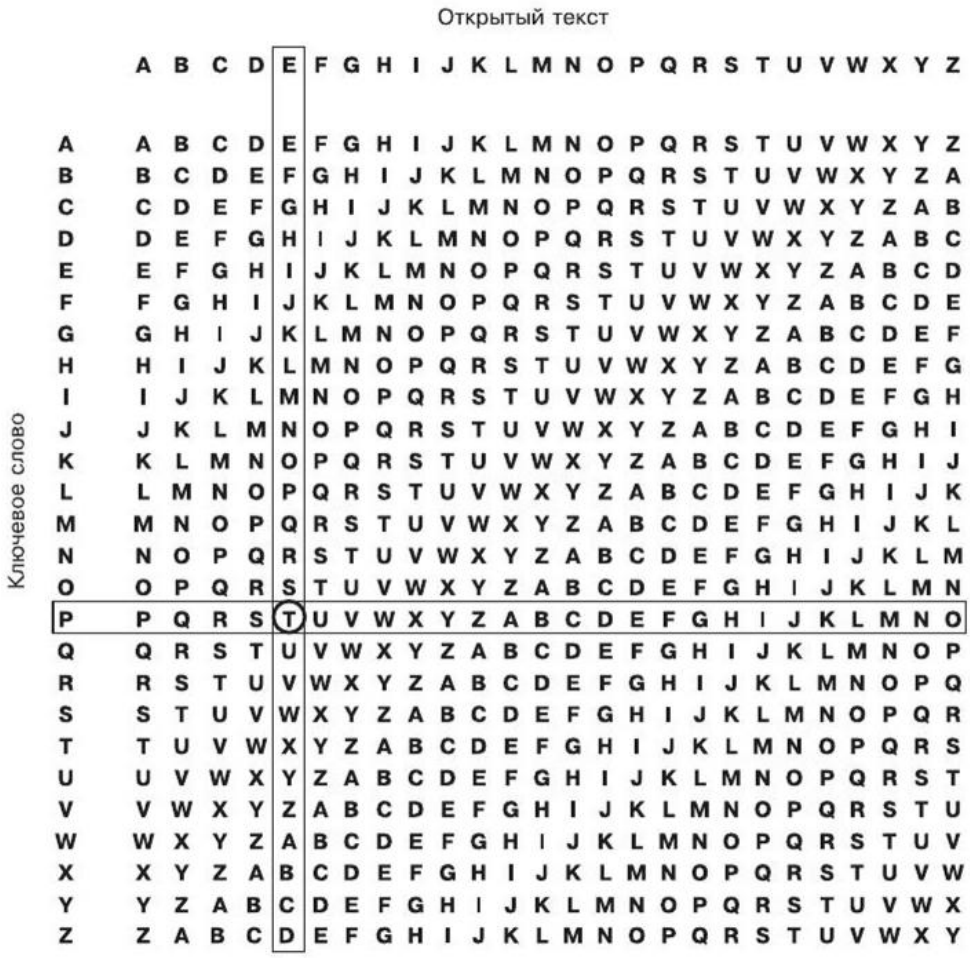


Рисунок 2.3 – Шифр Виженера, схематическое преобразование

Ключ:	PRO PA GA NDA PROP AGAN DAPR OPA GAND
Открытый текст:	GOD IS ON OUR SIDE LONG LIVE THE KING
Зашифрованный текст:	VFR XS UN BXR HZRT LUNT OIKV HWE QIAJ

Математическое определение подстановочного шифра (шифра простой замены):

Пусть даны произвольный ключ k, где k = (k_1, k_2, k_3, ..., k_n), где k_i ϵ Z_{26} для 1 ≤ i ≤ n, и произвольный открытый текст p, где p = (p_1, p_2, ..., p_m), где p_j ϵ Z_{26} для 1 ≤ j ≤ m. Пусть зашифрованный текст с представлен, как c = (c_1, c_2, ..., c_m), где c_j ϵ Z_{26} для 1 ≤ j ≤ m. Тогда мы определим функцию шифрования E_{ki} , как $c_j = E_{ki}$ (p_j), где E_{ki} (p) : $p_j \rightarrow$ = pj + ki (mod 26), а функция дешифрования D_k(c) определяется, как $p_j = D_{ki}$ (c_j), где D_{ki} (c) : $c_j \rightarrow c_j – k_i$ (mod 26).

Иными словами, шифр Виженера – это повторяющийся шифр Цезаря с разным смещением для каждого элемента в ключе.

Проектирование системы

Основная цель системы – создать динамический меняющийся алгоритм, способный подстраиваться под входные условия (скорость обработки данных, количество ключей и т.д.). Это достижимо, если придерживаться стратегии, используемой в генетических алгоритмах.

Для того, чтобы создать толчок к развитию, необходимо иметь то, что нужно развивать. Необходимо создать геном, способный охарактеризовать индивидуума таким образом, чтобы условия создания открытого и закрытого текстов выполнялись.

Для простоты будем использовать английский алфавит из 26 букв. Учитывая математическое определение шифров, нам необходимы ключи и сдвиг.

Иными словами, если разбить действия, совершаемые алгоритмом гена на простые шаги, получится приблизительно следующее:

1) Взять букву изначального текста
2) Определиться со сдвигом
3) Начать сдвиг

4) Передать полученную букву

Повторять, пока не будет достигнут конец слова.

Мне хотелось бы достичь масштабируемости в своем геноме, который возможно будет образовать, поэтому, прежде чем создавать систему, нужно определиться с ее условиями, при этом помня о необходимости дешифровки, поскольку нам нужен обратимый результат.

У всех шифров есть три ключевых элемента — ключ, алфавит, и операция, которая объединяет их вместе (позволяет им коррелировать), поэтому, включим их в геном, причем так, чтобы при необходимости они могли образовать следующие последовательности:

Шифр Цезаря:

1 ключ, 1 операция, 1 алфавит (порядок следования английских букв)

Шифр подстановок:

26 ключей, 1 операция, 2 алфавита (изначальный и смещенный)

Шифр Виженера:

1 <ключей <26, 1 операция, число алфавитов, совпадающее с числом ключей (каждая позиция ключа дает смещение изначального алфавита на одну позицию (математически: операции на кольцевом множестве Z_{26})

Поскольку в каждом из шифров используется 1 операция, стоит начать с ее непосредственной кодировки.

Во всех этих шифрах можно использовать любую обратимую математическую операцию (сложение, умножение, возведение в степень и т.д.), на данный момент рассмотрим шифр на основе сложения, помня, что при необходимости, операцию можно заменить.

Поскольку алгоритм предполагает возможность расшифровки, возможна кодировка не конкретной операции (прибавить или отнять), а самой необходимости произвести некую операцию, зависящую от внешне заданных условий.

Иными словами, в зависимости от характеристик значение в гене может представлять собой как сложение, так и вычитание в случаях шифрования и дешифрования соответственно.

Допустим, непосредственно операция будет кодироваться 1.

Следующий неизменный элемент — алфавит, допустим, английский (возможно с любым другим языком или последовательностью, организованной известным вам образом). Допустим, мы будем кодировать каждую букву в том же порядке, что и в английском алфавите. Получим последовательность кодов от 2-27.

С такими характеристиками операций гена уже возможно воссоздать шифр Цезаря, который будет кодироваться следующим образом, с учетом того, что считывание гена происходит слева направо и при считывании позиции смещение происходит на 1 цифру вправо:

5 1 5 1 5 1...5 1 5 1 — ген длиной в слово, которое мы можем зашифровать

Алгоритм дешифровки не меняется, стоит только использовать второе значение 1 - обратную операцию.

Полученный шифр подстановки (Виженера):

7 1 4 1 15 1 ...9 1 — основное условие для получение чистого шифра подстановки — цифры между единицами не должны повторяться.

Первый пришедший на ум вариант, как можно заметить, весьма избыточен. Да, мы получили малое количество элементов для кодировки, но учитывая факт развития популяции, вероятность того, что за приемлемое время (максимум 200-300 поколений) в каждой нечетной/четной позиции гена образуется повторяющаяся последовательность от начала до конца длиною в кодируемое слово весьма мала. Для слова длиной в 5 букв нам понадобится ген длиной в 10 цифр, причем различных значений в этих 10 цифрах - 2. Вероятность возникновений такой последовательности очень мала, из чего следует, что мутаций будет необходимо очень и очень много, да и индивидуумов тоже.

И еще один минус: как много попыток уйдет на получение способности индивида обнаруживать конец слова, учитывая то, что его ген имеет фиксированную длину и, скорее всего, индивид будет рождаться не согласованно с изменением длины слова (слово родителей может весьма отличаться от слова, доставшегося детям, что представляет собой проблему — в чем смысл динамический развивающейся системы, если она способна разеваться только в рамках слов какой-либо фиксированной длины?)

Стоит добавить еще одну возможную кодировку — код смещения чтения собственного гена. По умолчанию он — единица, но, если добавить возможность перемещаться по всему геному от начала и до конца при необходимости?

Для того, чтобы это сделать, в любом случае, нужно знать фиксированную длину гена, что уже не позволит попросту устанавливать количество кода в созависимости с словом. Но в данном случае, мы можем и не столкнуться с ситуацией, когда ген должен быть слишком большим, чтобы не быть способным зашифровать слово большой длины.

Допустим, длина нашего генома — 55 символов, тогда, по аналогии с кольцом для алфавита, нам необходимо будет кольцо Z_{55}. Нам понадобятся цифры от 28-55 для кодировки смещения.

Один из примеров для шифра Цезаря:

5 1 27 – выигрышная последовательность, ведь стоит ей появится, и она закрепится.

Для шифра Цезаря без возможности сдвига по собственному гену мы получили всего две выигрышные вариации, здесь же множество вариантов, и это лишь первые, что приходят на ум. Они самые рациональные, но возможны будут варианты просто с смещением на 4 рабочие позиции гена, а после назад. Разброс и сжатость кодировки позволяют увеличить работоспособность и приспосабливаемость индивида.

Следует заранее оговорить тот момент, что скорее всего при выполнении алгоритма чтения гена программа будет сама брать последовательно элементы слова и решать кода ей выйти из цикла, но можно пойти и более тяжелым путем, закодировав и эти две операции.

Тогда получим приблизительно следующий набор команд:

- 1 - +/ -
- 0 - взять слово
- 2 - выйти из цикла
- 3-28 – алфавит
- 29-56 смещение (для длины генома в 55 символов)

Способность взять элемент слова потенциально может дать индивидууму возможность кодировать какую-то часть информации, но не ее всю, что может быть проблемой.

Теперь, зная приблизительно геном, которого мы можем добиться и его самые желательные вариации следует заняться разработкой непосредственно самого генетического алгоритма.

Первое, с чем следует определиться, это с количеством мутаций. Допустим, для начала они будут образовываться случайным образом в 1-5 позициях.

Я не уверена в целесообразности работы кроссинговера, поскольку одна такая операция может полностью лишить смыла весь естественный отбор, разве что дав возможность выжить только тем индивидуумам, код которых будет буквально продублирован дважды. Не думаю, что это целесообразно.

Следующий элемент - окружающая среда. Допустим, у нас имеется 10 изначальных индивидуумов в начальной популяции. Они используют общее пространство, но двигаться не способны.

Следующий вопрос — использовать целевую функцию, турнирный отбор, или соревнование?

Если реализовывать соревнование по принципам естественного отбора, получим приблизительно следующее:

Приоритет будет отдаваться индивидуумам способным преобразовывать слово, посредством прибавления количества итераций их жизни (если ген индивида способен зашифровать слово полностью - +20 возможных циклов, если способен дешифровать его - +40 и возможность создать свою абсолютную копию).

Также можно контролировать преобладающий ген в системе давая приоритет по размножению тем индивидуумам, которые пережили всех остальных. То есть, новый индивид способен будет появляться только при наличии свободной позиции.

Также возможно будет влиять на процесс с помощью введения проверки с неким подобием турнира, если индивидуум гарантированно покажет свою негодность (к примеру, если добавить с систему отслеживание поколений, при наличии индивидуума, дающего популяцию, все еще не продвинувшуюся в основной задаче за 50-100 циклов, возможно, стоит от него избавиться).

В итоге, мы должны получить динамическую систему, в которой будет выживать от 10 индивидуумов. Идея для соревнования может быть такой: для того, чтобы выжить, индивидуум должен зашифровать слово таким образом, чтобы через какое-то время только этот индивидуум был способен правильно дешифровать его, следовательно, выжить.

Использовать данную стратегию отбора стоит при наличии системы, стабильно дающей необходимый результат, для того чтобы определить, что созданная система удовлетворяет поставленным требованиям, стоит пройтись по нескольким тестам.

Тесты:

- Тест на длинные слова
- тест на простейшие последовательности (только с двумя возможными типами кодировок)
- тест на рабочую последовательность кода (корреляция между его длиной и количеством поколений перед возникновением результата)
- тест при различных начальных условиях (разное время жизни, размер популяции, условия роста, наличия fittness функции)
- тест на мутации (какое их количество оптимально и в зависимости от какой длины кода и количества кодируемых операций)
- тест на добавление возможности выхода
- тест на добавление возможности выбирать элементы слова (возможно, даже непоследовательно)
- тест на кроссинговер

Заключение:

Основная цель этой работы - проектирование алгоритма для создания аналогии с естественным отбором и живыми существами для того, чтобы проверить, какие внешние условия будут влиять на развитие конкретных стратегий и посмотреть, к чему это в дальнейшем приведёт.

В работе делался упор на ту модель, которую я лично считаю наиболее рабочей. Только наличие нескольких практических результатов и данных позволит сказать, является ли она таковой. Для этого стоит пройтись по всем тестам, что я описала выше, и посмотреть, что произойдет.

Пока у меня имеется лишь теоретический материал, подтверждаемый формулами цифрами и догадками, но я уверена, что, начав практическое исполнение этой системы я столкнусь с многими занимательными вещами. Поэтому, дальнейшее развитие моего проекта - написание такого генетического алгоритма с помощью программных средств и современного аппаратного обеспечения, причем явно будет недостаточно создать просто рабочий алгоритм, необходима будет возможность его анализа и сбора информации о индивидуумах и их развитии, желательно удобным образом.

Возможно, если этот проект удастся, я смогу увеличить его масштаб для возможности кодирования более современных и криптостойких алгоритмов. Но это в более дальнем будущем.

Список использованных источников:

1. Торстейнсон П., Ганеш Г.А. Криптография и безопасность в технологии .NET: учеб. пособие / П. Торстейнсон, Г. А. Ганеш; пер. с англ. — М.: БИНОМ.: Изд-во Лаборатория знаний, 2015. — 479 с.

2. Панченко Т.В., Генетические алгоритмы: учеб. пособие / под ред. Ю. Ю. Тарасевича.; Изд-во Астраханского ун-та, 2007. — 87 с.

3. Протько М.А., Принципы построения алгоритмов шифрования, V Международная научно-практической конференция студентов, аспирантов, преподавателей, Армавир, 202. — 3 с.

UDK

CAESAR'S CIPHER AND GENETIC ALGORITHM

Protko M.A.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Borisenko O.F. – PhD in Physics and Mathematics

Annotation. Using the availability of modern technologies and innovations, it is possible to introduce new approaches to solving problems such as the search for a dynamically developing encryption system. This work contains the theoretical material necessary for the further development of this idea.

Keywords. Genetic algorithm, substitution ciphers, monoalphabetic and polyalphabetic ciphers, objective function, crossover, tournament selection