**Инструкции для преподавания по теме «Генетические алгоритмы и информационная безопасность».**

1. **Титульный слайд (название модуля, автор)**

Приветствие. Занятие «Генетические алгоритмы и информационная безопасность» в модуле «Искусственный интеллект в информационной безопасности».

1. **Определение генетических алгоритмов**

Определение. Генетический алгоритм — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе.

Генетические алгоритмы реализуют упрощенный вариант дарвиновской эволюции. Имитируя процессы естественного отбора и воспроизводства, генетические алгоритмы могут находить высококачественные решения задач, включающих поиск, оптимизацию и обучение. В то же время аналогия с естественным отбором позволяет этим алгоритмам преодолевать некоторые препятствия, встающие на пути традиционных алгоритмов поиска и оптимизации, особенно в задачах с большим числом параметров и сложными математическими представлениями.

Цель генетических алгоритмов – найти оптимальное решение некоторой задачи. Если дарвиновская эволюция развивает популяцию отдельных особей, то генетические алгоритмы развивают популяцию потенциальных решений данной задачи, называемых индивидуумами. Эти решения итеративно оцениваются и используются для создания нового поколения решений. Те, что лучше проявили себя при решении задачи, имеют больше шансов пройти отбор и передать свои качества следующему поколению. Так постепенно потенциальные решения совершенствуются в решении поставленной задачи.

В случае генетических алгоритмов каждому индивидууму соответствует хромосома, представляющая набор генов. Например, хромосому можно представить двоичной строкой, в которой каждый бит соответствует одному гену.

1. **Определение популяции**

В любой момент времени генетический алгоритм хранит популяцию индивидуумов – набор потенциальных решений поставленной задачи. Поскольку каждый индивидуум представлен некоторой хромосомой, эту популяцию можно рассматривать как коллекцию хромосом. Популяция всегда представляет текущее поколение и эволюционирует со временем, когда текущее поколение заменяется новым.

1. **Базовая структура**

На следующей блок-схеме показаны основные этапы типичного генетического алгоритма.

Тут кратко характеризуем каждый из этапов. Подробнее мы их разберем в рамках следующих слайдов.

1. **Создание начальной популяции**

Начальная популяция состоит из случайным образом выбранных потенциальных решений (индивидуумов). Поскольку в генетических алгоритмах индивидуумы представлены хромосомами, начальная популяция – это, по сути дела, набор хромосом. Формат хромосом должен соответствовать принятым для решаемой задачи правилам, например это могут быть двоичные строки определенной длины.

Если имеется априорная информация, то для увеличения скорости сходимости поиска она может быть использована при формировании начальной популяции

1. **Функция приспособленности**

Разбор определения.

Для каждого индивидуума вычисляется функция приспособленности. Это делается один раз для начальной популяции, а затем для каждого нового поколения после применения операторов отбора, скрещивания и мутации. Поскольку приспособленность любого индивидуума не зависит от всех остальных, эти вычисления можно производить параллельно. Так как на этапе отбора, следующем за вычислением приспособленности, более приспособленные индивидуумы обычно считаются лучшими решениями, генетические алгоритмы естественно «заточены» под нахождение максимумов функции приспособленности. Если в какой-то задаче нужен минимум, то при вычислении приспособленности следует инвертировать найденное значение, например умножив его на –1.

1. **Отбор. Правило рулетки**

После того как вычислены приспособленности всех индивидуумов в популяции, начинается процесс отбора, который определяет, какие индивидуумы будут оставлены для воспроизводства, т. е. создания потомков, образующих следующее поколение. Процесс отбора основан на оценке приспособленности индивидуумов. Те, чья оценка выше, имеют больше шансов передать свой генетический материал следующему поколению. Плохо приспособленные индивидуумы все равно могут быть отобраны, но с меньшей вероятностью. Таким образом, их генетический материал не полностью исключен.

Рассмотрим некоторые методы реализации отбора. Правило рулетки.

Метод отбора по правилу рулетки, или отбор пропорционально приспособленности (fitness proportionate selection – FPS), устроен так, что вероятность отбора индивидуума прямо пропорциональна его приспособленности. Тут можно провести аналогию с вращением колеса рулетки, где каждому индивидууму соответствует сектор, стоимость которого равна приспособленности индивидуума. Шансы, что шарик остановится в секторе индивидуума, пропорциональны размеру этого сектора. Пусть, например, имеется популяция из шести индивидуумов с такими значениями приспособленности, как в таблице ниже. По этим значениям вычисляются доли, занимаемые секторами каждого индивидуума.

После каждого запуска рулетки отбор индивидуума из популяции производится в точке отбора. Затем рулетка запускается еще раз для выбора следующего индивидуума, и так до тех пор, пока не наберется достаточно индивидуумов для образования следующего поколения. В результате один и тот же индивидуум может быть выбран несколько раз.

1. **Отбор. Стохастическая универсальная выборка**

Стохастическая универсальная выборка (stochastic universal sampling – SUS) – немного модифицированный вариант правила рулетки. Используется та же рулетка с такими же секторами, но вместо одной точки отбора и многократного запуска рулетки мы вращаем колесо только один раз, а отбор индивидуумов производим в нескольких точках, равномерно расставленных по окружности. Тем самым все индивидуумы выбираются одновременно, как показано на рисунке.

Этот метод отбора не дает индивидуумам с особенно высокой приспособленностью заполнить все следующее поколение в результате повторного выбора. Поэтому более слабым индивидуумам предоставляется шанс, а несправедливость чистого правила рулетки в какой-то мере сглаживается.

1. **Отбор. Ранжированный отбор**

Метод ранжированного отбора похож на правило рулетки, но значения приспособленности используются не для вычисления вероятностей выбора, а просто для сортировки индивидуумов. После сортировки каждому индивидууму назначается ранг, соответствующий его позиции в списке, а вероятности секторов рулетки вычисляются на основе этих рангов. Возьмем ту же самую популяцию из шести индивидуумов, что и раньше. И добавим в таблицу столбец с рангом индивидуума. Поскольку размер популяции равен 6, наивысший возможный ранг тоже равен 6, следующий по порядку – 5 и т. д. Каждому индивидууму сопоставляется сектор рулетки, вычисленный по этим рангам, а не по значениям функции приспособленности.

Ранжированный отбор полезен, когда есть несколько индивидуумов, гораздо лучше приспособленных, чем все остальные. Использование ранга вместо самой приспособленности мешает этим индивидуумам захватить всю популяцию в следующем поколении, поскольку ранжирование сглаживает значительные различия. Кроме того, когда все индивидуумы обладают почти одинаковой приспособленностью, ранжированный отбор позволяет разделить их, отдавая преимущество лучшим, даже когда различия малы.

1. **Отбор. Турнирный отбор**

В каждом раунде турнирного отбора из популяции выбираются два или более индивидуумов, и тот, у кого приспособленность больше, выигрывает и отбирается в следующее поколение. Рассмотрим тех же индивидуумов с такими же приспособленностями, что и ранее. На рисунке показан результат случайного выбора трех из них (A, B и F) с последующим объявлением F победителем, поскольку у него приспособленность максимальная из трех.

Количество индивидуумов, участвующих в каждом раунде турнирного отбора (в нашем примере – три), называется размером турнира. Чем больше размер турнира, тем выше шансы, что в раундах будут участвовать лучшие индивидуумы, и тем меньше шансов у слабых участников победить в турнире и отобраться. У этого метода отбора есть интересная особенность: если мы умеем сравнивать любых двух индивидуумов и определять, какой из них лучше, то сами значения функции приспособленности и не нужны.

1. **Скрещивание**

Для создания пары новых индивидуумов родители обычно выбираются из текущего поколения, а части их хромосом меняются местами (скрещиваются), в результате чего создаются две новые хромосомы, представляющие потомков. Эта операция называется скрещиванием, или рекомбинацией.

Как правило, оператор скрещивания применяется не всегда, а с некоторой (высокой) вероятностью. Если скрещивание не применяется, то копии обоих родителей переходят в следующее поколение без изменения.

Простейший способ – одноточечное скрещивание.

В этом случае позиция в хромосомах обоих родителей выбирается случайным образом. Эта позиция называется точкой скрещивания, или точкой разреза. Гены одной хромосомы, расположенные справа от этой точки, обмениваются с точно так же расположенными генами другой хромосомы. В результате мы получаем двух потомков, несущих генетическую информацию обоих родителей.

1. **Двухточечное и k-точечное скрещивание**

При двухточечном скрещивании случайным образом выбираются по две точки скрещивания в каждой хромосоме. Гены одной хромосомы, расположенные между этими точками, обмениваются с точно так же расположенными генами другой хромосомы. Метод двухточечного скрещивания можно реализовать с помощью двух одноточечных скрещиваний с разными точками скрещивания. Его обобщением является метод k-точечного скрещивания, где k – целое положительное число.

1. **Равномерное скрещивание**

При равномерном скрещивании каждый ген обоих родителей определяется независимо путем случайного выбора с равномерным распределением. Когда выбирается 50 % генов, оба родителя имеют одинаковые шансы повлиять на потомков. Заметим, что в этом примере гены обоих потомков меняются местами, но в принципе потомков можно создавать и независимо. Поскольку в этом методе не производится обмен целых участков хромосом, потенциально он может повысить разнообразие потомков.

Существуют и другие методы, но разобрать все в рамках лекции не успеть. Более того, можно придумывать свои методы скрещивания.

1. **Мутация**

Цель оператора мутации – периодически случайным образом обновлять популяцию, т. е. вносить новые сочетания генов в хромосомы, стимулируя тем самым поиск в неисследованных областях пространства решений. Операция мутации вероятностная, обычно она выполняется изредка, с очень низкой вероятностью, поскольку может ухудшить качество индивидуума, к которому применена. В некоторых вариантах генетических алгоритмов вероятность мутации постепенно увеличивается, чтобы предотвратить стагнацию и повысить разнообразие популяции. С другой стороны, если частота мутации слишком велика, то генетический алгоритм выродится в случайный поиск.

Мутация может проявляться как случайное изменение гена. Мутации реализуются с помощью внесения случайных изменений в значения хромосом, например инвертирования одного бита в двоичной строке (отображено на слайде).

Другой пример – мутация обменом.

Мутация обращением – изменение порядка генов на противоположный

Мутация перетасовкой - в этом случае выбирается случайная последовательность генов, и порядок генов в ней изменяется случайным образом (тасуется).

Но не забывайте, что вы всегда можете придумать свой собственный метод, отвечающий специфике конкретной задачи.

1. **Критерии останова**

Может существовать несколько условий, при выполнении которых процесс останавливается. Сначала отметим два самых распространенных:

* достигнуто максимальное количество поколений;
* достигнуто целевое значение функции приспособленности.

Перечислим также другие возможные условия:

* с момента начала прошло заранее определенное время;
* превышен некоторый лимит затрат, например процессорного времени или памяти;
* наилучшее решение заняло часть популяции, большую заранее заданного порога;
* на протяжении нескольких последних поколений не наблюдается заметных улучшений. Это можно реализовать путем запоминания наилучшей приспособленности, достигнутой в каждом поколении, и сравнения наилучшего текущего значения со значениями в нескольких предыдущих поколениях. Если разница меньше заранее заданного порога, то алгоритм можно останавливать.

1. **Теоретическая основа – понятие схемы**

Пусть используется простой генетический алгоритм, то есть ГА с одноточечным кроссинговером и одноточечной мутацией. Будем считать, что особью в популяции является бинарная строка длины l. Если это не так, всегда можно закодировать ее нужным образом. Разбираем определение схемы, порядка схемы и определяющей длины схемы. Подкрепляем все примерами со слайда. Каждая хромосома в популяции соответствует нескольким схемам – точно так же, как заданная строка соответствует разным регулярным выражениям.

Подробнее - <http://www.machinelearning.ru/wiki/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D1%8B>

1. **Теоретическая основа – теорема Холланда о схемах**

Разбираем понятия со слайда.

Суть теоремы. Теорема о схемах утверждает, что частота схем низкого порядка с малым определяющим расстоянием и приспособленностью выше средней экспоненциально возрастает в последующих поколениях. Иными словами, генетический алгоритм увеличивает частоту в популяции небольших и простых структурных элементов, представляющих атрибуты, благодаря которым решение становится лучше.

Подробнее - <http://www.machinelearning.ru/wiki/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D1%8B>

1. **Теоретическая основа – теорема Холланда о схемах**

Разбираем компоненты из формулы, их смысловую нагрузку.

1. **Отличия ГА от традиционных алгоритмов**

Комментарии по каждому пункту.

Популяция решений. Целью генетического поиска является популяция потенциальных решений (индивидуумов), а не единственное решение. В любой точке поиска алгоритм сохраняет множество индивидуумов, образующих текущее поколение. На каждой итерации генетического алгоритма создается следующее поколение индивидуумов. С другой стороны, в большинстве других алгоритмов поиска хранится единственное решение, которое итеративно улучшается. Например, алгоритм градиентного спуска итеративно сдвигает текущее решение в направлении наискорейшего спуска, которое определяется антиградиентом заданной функции.

Генетическое представление. Генетические алгоритмы работают не с самими потенциальными решениями, а с их кодированными представлениями, которые часто называют хромосомами. Простым примером хромосомы является двоичная строка фиксированной длины. Хромосомы позволяют определить генетические операции скрещивания и мутации. Скрещивание реализуется обменом частей родительских хромосом, а мутация – изменением частей хромосом. Побочный эффект генетического представления – отделение поиска от исходной предметной области. Генетические алгоритмы не знают, что именно представляют хромосомы, и не пытаются их интерпретировать.

Функция приспособленности. Функция приспособленности представляет проблему, которую мы пытаемся решить. Цель генетического алгоритма – найти индивидуумов, для которых оценка, вычисляемая функцией приспособленности, максимальна. В отличие от традиционных алгоритмов поиска, генетические алгоритмы анализируют только значение, возвращенное функцией приспособленности, их не интересует ни производная, ни какая-либо другая информация. Поэтому они могут работать с функциями, которые трудно или невозможно продифференцировать.

Вероятностное поведение. Многие традиционные алгоритмы по природе своей детерминированы, тогда как правила, применяемые генетическими алгоритмами для перехода от предыдущего поколения к следующему, вероятностные. Например, вероятность отбора индивидуума для создания следующего поколения тем выше, чем больше значение функции приспособленности, но элемент случайности все равно присутствует. Слабо приспособленные индивидуумы могут быть отобраны, хотя вероятность этого ниже. Мутации тоже имеют вероятностный характер, обычно их вероятность мала, а изменению подвергаются случайные позиции в хромосоме. Случайность присутствует и в операторе скрещивания. В некоторых генетических алгоритмах скрещивание происходит лишь с некоторой вероятностью. Если скрещивания не было, то оба родителя дублируются в следующем поколении вообще без изменений. Несмотря на вероятностную природу процесса, поиск, основанный на генетическом алгоритме, нельзя назвать случайным; случайность используется, чтобы направить поиск в сторону тех областей пространства поиска, где выше шансы улучшить результаты. Теперь рассмотрим преимущества генетических алгоритмов.

1. **Преимущества генетических алгоритмов**

Особенности генетических алгоритмов, рассмотренные в предыдущих разделах, определяют их преимущества по сравнению с традиционными алгоритмами поиска.

Комментарии по каждому пункту.

Глобальная оптимизация. Во многих задачах оптимизации имеются точки локального максимума и минимума, которые представляют решения, лучшие, чем те, что находятся поблизости, но необязательно лучшие в глобальном смысле. Большинство традиционных алгоритмов поиска и оптимизации, а особенно те, что основаны на вычислении градиента, могут застревать в локальном максимуме, вместо того чтобы найти глобальный. Это связано с тем, что в окрестности локального максимума всякое небольшое изменение решения ухудшает оценку. Генетические алгоритмы менее подвержены этой напасти и имеют больше шансов отыскать глобальный максимум. Объясняется это тем, что используется популяция потенциальных решений, а не единственное решение, а операции скрещивания и мутации зачастую порождают решения, далеко отстоящие от ранее рассмотренных. Это остается справедливым при условии, что мы поддерживаем разнообразие популяции и избегаем преждевременной сходимости, о чем поговорим далее.

Применимость к сложным задачам. Поскольку генетическим алгоритмам нужно знать только значение функции приспособленности каждого индивидуума, а все остальные ее свойства, в частности производные, несущественны, их можно применять к задачам со сложным математическим представлением, включающим функции, которые трудно или невозможно продифференцировать. К сложным случаям, когда достоинства генетических алгоритмов раскрываются во всем блеске, относятся также задачи с большим числом параметров или со смешанными параметрами, например непрерывными и дискретными.

Применимость к задачам, не имеющим математического представления. Генетические алгоритмы применимы и к задачам, вообще не имеющим математического представления. Один из таких случаев, представляющий особый интерес, – когда оценка приспособленности основана на мнении человека. Пусть, например, требуется найти наиболее привлекательную цветовую палитру для веб-сайта. Мы можем попробовать разные комбинации цветов и попросить пользователей оценить привлекательность сайта. А затем применить генетический алгоритм, чтобы найти лучшую комбинацию, используя функцию приспособленности, основанную на оценках пользователей. Алгоритм будет работать, несмотря на то что никакого математического представления нет и невозможно вычислить оценку заданной комбинации непосредственно.

Устойчивость к шуму. Для некоторых задач характерно присутствие шума. Это означает, что даже при близких истинных значениях входных параметров результаты их измерений могут довольно сильно различаться. Например, так бывает, когда данные считываются с датчиков или когда оценка основана на мнении человека. Подобное поведение может сделать непригодными многие традиционные алгоритмы поиска, но генетические алгоритмы в общем случае устойчивы к нему благодаря повторяющимся операциям сборки и оценивания индивидуумов.

Распараллеливание. Генетические алгоритмы хорошо поддаются распараллеливанию и распределенной обработке. Функция приспособленности независимо вычисляется для каждого индивидуума, а это значит, что все индивидуумы в популяции могут обрабатываться одновременно. Кроме того, операции отбора, скрещивания и мутации могут одновременно выполняться для индивидуумов и пар индивидуумов. Поэтому подход, основанный на генетических алгоритмах, естественно адаптируется к распределенным и облачным реализациям.

Непрерывное обучение. Если окружающие условия изменяются, популяция приспосабливается к ним. Так и генетические алгоритмы могут непрерывно работать в постоянно изменяющихся условиях, и мы всегда можем получить и использовать лучшее на данный момент решение. Но это возможно, только если окружающая среда изменяется медленно по сравнению со скоростью смены поколений в генетическом алгоритме.

1. **Ограничения генетических алгоритмов**

Чтобы получить максимум пользы от генетических алгоритмов, мы должны знать об их ограничениях и потенциальных подвохах.

Комментарии по каждому пункту.

Специальные определения. Пытаясь применить генетические алгоритмы к некоторой задаче, мы должны создать подходящее представление – определить функцию приспособленности и структуру хромосом, а также операторы отбора, скрещивания и мутации. Зачастую это совсем не просто и занимает много времени.

Настройка гиперпараметров. Поведение генетических алгоритмов контролируется набором гиперпараметров, например размером популяции и скоростью мутации. Точных правил для выбора значений гиперпараметров не существует. Однако так обстоит дело практически со всеми алгоритмами поиска и оптимизации. Опыт – основной пункт к подбору разумных гиперпараметров.

Большой объем счетных операций. Работа с потенциально большими популяциями и итеративный характер генетических алгоритмов обусловливают большой объем вычислений, поэтому на получение приемлемого результата может уйти много времени. Проблему можно сгладить за счет хорошего выбора гиперпараметров, распараллеливания и в некоторых случаях кеширования промежуточных результатов.

Преждевременная сходимость. Если приспособленность какого-то индивидуума гораздо больше, чем у всей остальной популяции, то не исключено, что он продублируется так много раз, что в конечном счете, кроме него, в популяции ничего не останется. В результате генетический алгоритм может застрять в локальном максимуме и не найдет глобального. Чтобы предотвратить такое развитие событий, важно поддерживать разнообразие популяции.

Отсутствие гарантированного решения. Использование генетических алгоритмов не гарантирует нахождения глобального максимума. Однако это типично для всех алгоритмов поиска и оптимизации, если только у задачи не существует аналитического решения.

1. **Для решения каких задач подходят ГА?**

Резюмируя изложенное в предыдущих слайдах, можно сказать, что генетические алгоритмы лучше применять для решения следующих задач.

Задачи со сложным математическим представлением. Поскольку генетическим алгоритмам нужно знать только значение функции приспособленности, их можно использовать для решения задач, в которых целевую функцию трудно или невозможно продифференцировать, задач с большим количеством параметров и задач с параметрами разных типов.

Задачи, не имеющие математического представления. Генетические алгоритмы не требуют математического представления задачи, коль скоро можно получить значение оценки или существует метод сравнения двух решений.

Задачи с зашумленной окружающей средой. Генетические алгоритмы устойчивы к зашумленным данным, например прочитанным с датчика или основанным на оценках, сделанных человеком.

Задачи, в которых окружающая среда изменяется во времени. Генетические алгоритмы могут адаптироваться к медленным изменениям окружающей среды, поскольку постоянно создают новые поколения, приспосабливающиеся к изменениям.

1. **Фреймворк DEAP**

Для работы с генетическими алгоритмами создан целый ряд фреймворков на Python, например GAFT, Pyevolve и PyGMO. Но мы остановимся на фреймворке DEAP, поскольку он прост в использовании и предлагает широкий набор функций, поддерживает расширяемость и может похвастаться подробной документацией. DEAP (сокращение от Distributed Evolutionary Algorithms in Python – распределенные эволюционные алгоритмы на Python) поддерживает быструю разработку решений с применением генетических алгоритмов и других методов эволюционных вычислений. DEAP предлагает различные структуры данных и инструменты, необходимые для реализации самых разных решений на основе генетических алгоритмов.

Рассмотрим два основные модуля toolbox и creator. Разбираем примеры на слайде.

Подробнее о фреймфорке и его возможностях - <https://deap.readthedocs.io/en/master/>

1. **Примеры применения в ИБ**

Одной из задач при построении системы защиты информации является составление набора защитных мер. При этом в необходимо учитывать не только требования безопасности, но и затраты на внедрение и поддержание системы защиты информации. Обычно при составлении набора защитных мер используется метод, основанный на экспертном подходе, который сводится к предпочтениям экспертов, но так как количество возможных вариантов конфигурации системы защиты превышает мыслительные возможности, то в итоге к типовым схемам. В качестве критериев используются суммарная величина риска и стоимость системы защиты информации.

Хромосома – набор защитных мер, закодированных в форме двоичного числа.

Функция приспособленности - оценка эффективности системы защиты для заданного профиля атаки с учетом ее стоимости, которую необходимо максимизировать.

Оценка рисков выполняется с помощью специальной модели (на слайде приведен пример таблицы для расчета рисков).

Мутация – случайное инвертирование двух бит.

Начальная популяция – два случайных индивидуума

Селекция – отбор K лучших.

Детальный пример: <http://itids.ugatu.su/index.php/itids/itids2018/paper/view/51>

1. **Примеры применения в ИБ**

Формирует именно набор правил, а не одно правило, выявляющее все атаки (такое правило не достижимо).

Описание алгоритма: <https://www.researchgate.net/publication/288345210_Using_Genetic_Algorithm_in_Network_Security>

1. **Примеры применения в ИБ**

Задача: выявить какие параметры конфигурации ведут к тем или иным уязвимостям. Существуют алгоритмы, которые позволяют на основе уязвимых конфигураций выявлять параметры, которые ведут к тем или иным уязвимостям. Для качественной работы алгоритма нужно получить большой набор уязвимых конфигураций. Эту задачу можно решить с использованием генетических алгоритмов.

Подробнее - <https://wakespace.lib.wfu.edu/bitstream/handle/10339/59317/Odell_wfu_0248M_10904.pdf>

1. **Примеры применения в ИБ**

Пример шифрования изображения из статьи <https://docsdrive.com/pdfs/ansinet/itj/2006/516-519.pdf>

Подробнее познакомимся в рамках практического занятия

1. **Финальный слайд**

Спасибо за внимание!

**Перечень рекомендованной литературы.**

1. Э. Версански. Генетические алгоритмы на Python
2. Описание теоремы о схемах на портале machine learning.ru <http://www.machinelearning.ru/wiki/index.php?title=%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D1%8B>
3. Mohammed A.F. Al- Husainy. Image Encryption Using Genetic Algorithm. <https://docsdrive.com/pdfs/ansinet/itj/2006/516-519.pdf>
4. C.A. Odell. Using genetic algorithms to detect security related software parameter chains. A Thesis for the Degree of MASTER OF SCIENCE.
5. B. Ehab, H. O. Nasereddin, Hebah. (2010). Using Genetic Algorithm in Network Security. <https://www.researchgate.net/publication/288345210_Using_Genetic_Algorithm_in_Network_Security>
6. С. О. Иванов, Д. В. Ильин, Л. А. Ильина. Генетический алгоритм подбора оптимальной конфигурации системы защиты информации. <http://itids.ugatu.su/index.php/itids/itids2018/paper/view/51>
7. Официальная страница фреймворка DEAP. <https://deap.readthedocs.io/en/master/>