ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Отделение программной инженерии

Кафедра Управления разработкой программного обеспечения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

НА ТЕМУ «ПРОГРАММА ПРОЕКТИРОВАНИЯ МЕХАНИЧЕСКОГО УСТРОЙСТВА ПО ЗАДАННЫМ ПАРАМЕТРАМ НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ»

Студента группы № 471

Пустовалова Михаила Юрьевича

_____

Научный руководитель

доцент кафедры УРПО,к.т.н.
Ахметсафина Р.З.

_____

Москва, 2013 г.

АННОТАЦИЯ

В данной работе представлено исследование в области генетических алгоритмов (ГА), по результатам которого является Windows приложение. Цель работы - проанализировать степень применимости ГА в задачах условной оптимизации и поиска решения. Задачи, поставленные в этой работе могут быть разделены на два типа: анализ принципов работы ГА и их практическое применение.

В работе приводится краткий обзор эволюционных алгоритмов, выбран механический объект для иллюстрации работы ГА, разработано Windows приложение, в котором реализован ГА и анимация механического устройства с учетом законов механики.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Генетический алгоритм — это алгоритм эвристического поиска, используемый для решения задач поиска и оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, повторяющих принципы природной эволюции. Ключевыми процедурами алгоритма являются *скрещивание, мутация и селекция*.

Генетический алгоритм оперирует данными, закодированными в двоичную последовательность. Комбинация всех параметров одного представителя популяции называется *хромосомой* или *индивидуумом*.

Скрещивание - процесс обмена битами среди хромосом двух родителей.

Мутация - процесс внесения случайных изменений в хромосому. Обычно, шанс внесения изменений задается крайне малым, т.к. цель мутации - избежать вырождения популяции в один локальный максимум.

Селекция - процесс отбора родителей для формирования нового потомства. Основывается на результате фитнес-функции, вычисленной для данного родителя. Чем ближе фитнес-функция к искомому значению, тем больший шанс того, что данный родитель создаст своего потомка.

Исследования в области генетических алгоритмов велись на протяжении последних 50 лет, начавшись еще в 60-х годах XX века. Пионером в этой области был John Holland [1]. Множество современных разработок в области ГА базируются на достижениях тех лет, развивая их и привнося все новые методы работы с данными.

Основной целью этой работы является исследование в области ГА, сравнительный анализ подходов к реализации ГА и оценка эффективности их работы. В основе ГА лежит нечеткий подход (в процессе поиска решения очень большую роль играет случай), что значит, что алгоритм не может быть применен для решения задач, в которых требуется высокая точность и четкость нахождения решения. Но в реальном мире существует очень много задач, где не требуется высокая точность решения. Например, поиск по текстам сайта и

нахождения максимального соответствия запросу. В этом случае ГА является хорошим подходом для поиска решения.

Взглянув на спектр проблем, которые решаются с помощью ГА, можно выделить наиболее популярные из них. Например, оптимизация сетей труб городского водо-/газо-провода [2], оптимизация компьютерных сетей. Так же ГА используются в машинном обучении [3], при написании искусственного интеллекта и т.д.

Еще одним примером, связанным с условием задачи данной работы являются гонки Формула-1, в которых аэродинамические показатели обсчитываются с помощью ГА [4].

У эволюционных алгоритмов много преимуществ. Из-за специфики реализации и работы с данными, ГА, как и любой эволюционный, может быть адаптирован практически под любую задачу. Для поиска решения ему на вход подаются только список варьируемых параметров для оптимизации и целевая функция. Еще одним преимуществом ГА можно назвать его неитеративный подход. Вычисление решений внутри одного поколения может производиться параллельно, по-максимуму используя вычислительные мощности машины.

На протяжении последних десяти лет ГА получили широкое распространение для решения оптимизационных задач. Более того, появилось множество вариаций самого ГА. С развитием облачных вычислений, ГА сейчас могут быть распараллелены для использования множества вычислительных мощностей одновременно с помощью алгоритма map/reduce [5]. Одним из последних применений ГА является использование его для наложения невидимых водяных знаков на JPEG-изображения для защиты их авторства [6].

Для реализации алгоритма, все параметры будут закодированы строками из бит. Это позволяет легко применять функции скрещивания и мутации к потомкам. Размер популяции будет ограничен на уровне 20 потомков, т.к. для тестирования каждого из потомков понадобится время, пока условия остановки

устройства не сработают. Комбинация этих ограничений позволит алгоритму найти приемлемое решение за приемлемое время.

Реализовав ГА для решения этой смоделированной задачи, в будущем реализацию алгоритма можно будет применить для решения других задач. Например, для вычисления точки входа в позицию для алгоритмического робота, торгующего на фондовой бирже, основываясь на истории сделок.

Вместе с реализацией алгоритма еще одной задачей является протестировать найденные решения. Для этого будет реализован модуль двухмерной спрайтовой анимации. Части механического устройства будут представлены спрайтами и взаимодействовать друг с другом и с поверхностью с учетом законов механики. Результирующим значением фитнес-функции ГА будет расстояние, пройденное устройством до срабатывания условия остановки.

1. ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ

1.1. Основные понятия.

С развитием цифровых технологий и проникновением их во все большее количество сфер жизни, возникает все большее количество задач поиска решения с различными критериями поиска и ограничениями. Идеальным решением для всех задач является искусственный интеллект (ИИ). Но разработка ИИ является сложной задачей, на данном этапе технического прогресса пока еще не решенной. Для построения работающего ИИ требуется полностью повторить механику работы головного мозга человека. Мозг человека является предметом крайне сложным, многогранным и до сих пор не изученным до конца.

Было разработано количество подходов для воспроизведения принципов работы головного мозга, один из которых идет по пути повторения естественных процессов, происходящих в природе. Одним из таких процессов является эволюция. Принципы естественного отбора и эволюции изучены, поэтому уже сейчас имеется множество примеров алгоритмизации процессов, действующих по принципу натуральной эволюции. Все алгоритмы из этой области получили название “эволюционных алгоритмов”.

История области эволюционных алгоритмов содержит в себе большое количество вариаций самих алгоритмов:

1) Генетический алгоритм
2) Генетическое программирование
3) Эволюционное программирование
4) Эволюционная стратегия
5) Дифференциальная эволюция
6) Нейроэволюция и т.д.

Но идея, лежащая в их основе, одна: на вход поступает популяция индивидуумов, которая подвергается некоторому внешнему воздействию,

заставляющее эти индивидуумы порождать более приспособленных к воздействию потомков. Таким образом улучшается приспособленность всего поколения, что в итоге приводит к индивидуумам, удовлетворяющим целевому условию. Имея целевую функцию, к которой мы стремимся, случайным образом задаем набор решений и применяем к ним функцию - результат и будет засчитан за значение целевой (фитнес-) функции данного кандидата. Чем оно ближе к целевой - тем лучше кандидат. На основе данной фитнес-функции выбираются кандидаты для производства следующего поколения (чем лучше значение функции - тем больше шанс, что кандидат даст потомство). Производство потомства состоит из двух частей:

1) рекомбинации, скрещивания

2) мутации

Рекомбинацией называется оператор скрещивания двух потомков для производства потомства со свойствами, присущими обоим предкам. Мутация - процесс внесения случайных изменений в потомство, применяется к одному кандидату, на выходе так же один кандидат. В итоге, после прохождения двух этапов получается новое поколение, с теоретически большим средним значением фитнес-функции, чем прошлое. Данный процесс продолжается до тех пор, пока не достигнуто желаемое значение, либо пока не достигнут предел вычисления (по времени или по ресурсам). Эволюционные алгоритмы часто называют “адаптационными”, т.к. значения кандидатов “адаптируются” под искомое значение.

Заметим, что все процессы в эволюционных алгоритмах являются стохастическими. Это значит, что более подходящие кандидаты имеют больший шанс дать потомство, но это не отрицает шанс очень слабых кандидатов так же породить новое поколение. Так же и процесс мутации происходит с некоторой долей вероятности. Заведомо неизвестно, к чему приведет мутация, на каких особях она будет произведена и в какую ветвь будет развиваться алгоритм далее.

Приведем псевдо-код работы всех эволюционных алгоритмов (ЭА):

НАЧАЛО
  ИНИЦИАЛИЗАЦИЯ поколения случайными кандидатами;
  ПРОТЕСТИРОВАТЬ каждого кандидата;
  ПОВТОРЯТЬ ПОКА (!*условие останова*)
    ВЫБРАТЬ предков;
    РЕКОМБИНИРОВАТЬ предков;
    МУТАЦИЯ полученного кандидата;
    ПРОТЕСТИРОВАТЬ кандидатов;
    ВЫБРАТЬ кандидатов для формирования нового поколения;
  КОНЕЦ ПОВТОРЕНИЯ
КОНЕЦ

Графическую схему ЭА можно представить в следующем виде:

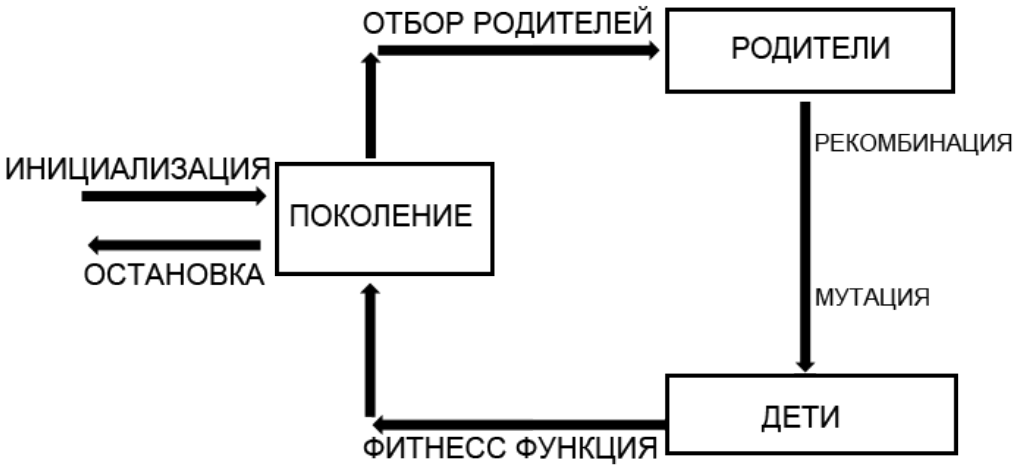

Рис. 1. Схема работы эволюционного алгоритма

Все диалекты ЭА работают по принципу, изображенному на рис. 1. Различия имеют лишь технический характер. Например, представление закодированных кандидатов. В генетических алгоритмах они представляются в виде строк, состоящих из бит (так называемых генов), в эволюционных стратегиях кандидаты представлены в виде целочисленных векторов. В эволюционном программировании исходные данные представляются в виде конечных автоматов. В генетическом программировании кандидаты хранятся в виде деревьев. Эти технические различия существуют лишь для удобства применения алгоритма для разных типов задач. Например, для решения задачи на проверку выполнимости логично кодировать данные строками из n бит, то есть использовать ГА. Если же стоит задача улучшения алгоритма игры в шашки, то наиболее подходящим выбором станет генетическое программирование. Важно помнить, что независимо от выбора представления эволюционного алгоритма, операции рекомбинации и мутации должны оперировать исходными данными именно в том виде, в котором они представлены, т.е. при генетическом программировании рекомбинация проводится на деревьях, в то время как в ГА рекомбинация осуществляется на строках из бит.

1.2. Описание работы эволюционного алгоритма

Как уже описывалось ранее, для работы ЭА требуется провести несколько шагов:

1) выбрать формат представления индивидуумов
2) Обозначить фитнес-функцию
3) Определить параметры формирования популяции
4) Задать функцию отбора родителей
5) Определить параметры операторов рекомбинации, мутации
6) Задать функцию определения живучести

Помимо этого должны быть определены условия старта и остановки алгоритма.

Чтобы продемонстрировать как работает ЭА и почему он является эффективным, можно показать алгоритм его работы визуально.

Рис. 2 показывает три этапа по времени работы ЭА.

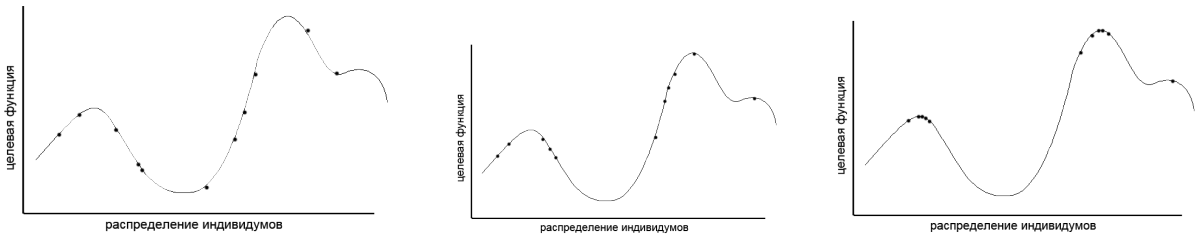

Рис. 2. Три этапа работы ЭА

На рис. 2 изображены 3 временных точки в работе ЭА, показывающих распределение кандидатов в начале работы алгоритма (слева), в середине (центр) и в конце работы алгоритма (справа). В начальной фазе, сразу после инициализации, хромосомы распределены случайным образом по всей области определения. После нескольких циклов работы алгоритма (селекции, рекомбинации и мутации) популяция отбросила самые неприспособленные (по фитнес-функции) кандидаты и произвела новых потомков из тех, что были ближе к вершине, тем самым улучшая среднее значение всей популяции. Ближе к концу работы алгоритма все кандидаты находятся в локальных максимумах.

Конечно имеется шанс, что все кандидаты выродятся в локальный максимум, который не будет являться глобальным, но шанс этого крайне мал. Изначально кандидаты случайным образом распределяются более равномерно по всей области определения, что не позволяет всем кандидатам прийти в одну точку (если она не является единственным максимумом). Для уменьшения шанса сведения всех кандидатов к локальному максимуму обычно заранее задается некоторая область на всем пространстве в пределах которой алгоритм будет искать максимумы.

Еще одним свойством эволюционно алгоритма, которое присуще всем итерационным алгоритмам, оперирующим заранее заданным набором решений, является эффективность в любой точке во времени. Это значит, что алгоритм может начинать работать с любого заданного набора и пытаться его улучшить, действуя бесконечно. Это свойство вызывает проблему нахождения оптимума между временем поиска и вычислительными ресурсами, требуемыми для дальнейшей работы алгоритма. На рис. 3 приведен график увеличения наилучшего значения фитнес-функции в поколении по мере работы алгоритма.

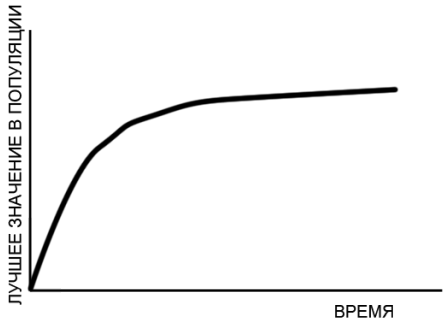

Рис. 3. Улучшение значения кандидата по мере работы ЭА

Как видно из графика (рис. 3), значение фитнес-функции растет логарифмически. Значит существует некоторая точка останова, в которой прирост величины фитнес-функции будет нерентабелен относительно затраченного времени для его вычисления. Эта точка имеет субъективный характер и выбирается относительно решаемой задачи.

Подводя итог рассмотрению ЭА, можно привести результаты исследования David E. Goldberg [3], который сравнил эффективности различных алгоритмов на широком спектре задач. Усредненные и абстрагированные результаты представлены на рис. 4.

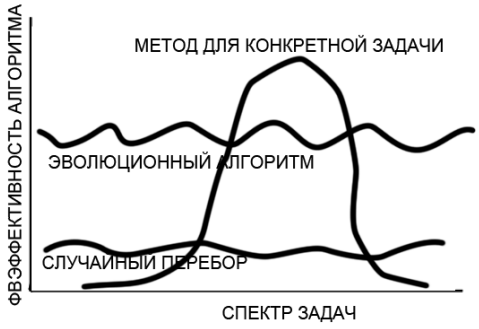

Рис. 4. Сравнение работы различных алгоритмов по David E. Goldberg

2. РАЗРАБОТКА АЛГОРИТМА ДЛЯ ЗАДАЧИ ПРОЕКТИРОВАНИЯ МЕХАНИЧЕСКОГО УСТРОЙСТВА

2.1. Представление задачи в терминах ГА

В представленной работе рассматривается проблема оптимизации основных параметров механического устройства (примитивной модели автомобиля). Модель устройства состоит из рамы (прямоугольник с изменяемыми параметрами: ширина, высота), двух колес (изменяемый параметр - радиус) и полезная нагрузка. Для каждого индивидуума модели также меняется скорость, но остается постоянной на протяжении жизни индивидуума . Схематически модель представлена на рис. 5.

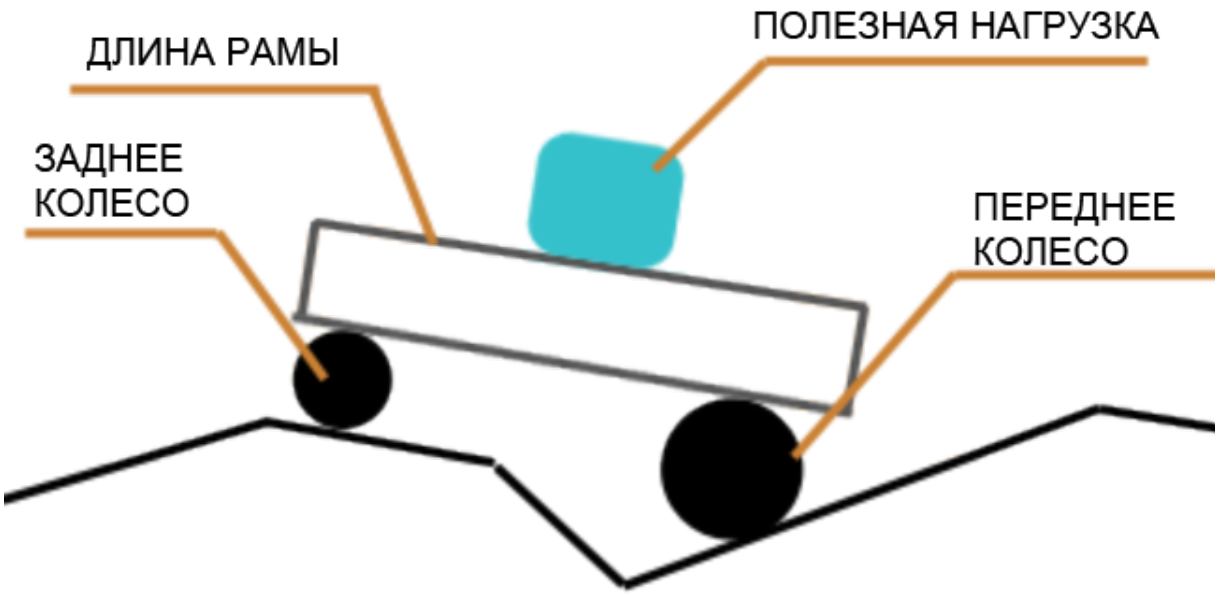

Рис. 5. Схема механического устройства

Все подбираемые параметры будут представлены в виде бинарных строк длиной 10 бит (2^10 < 1024). Далее, каждый потомок будет тестироваться и получать финальное значение фитнес-функции. Значение фитнес-функции есть расстояние, пройденное моделью с примененными параметрами даннойго индивидуума. Расстояние засчитывается, когда сработает одно из двух условий остановки: либо устройство застрянет и не сдвинется с места в течении 5 секунд, либо полезный груз коснется земли (это возможно, т.к. рама модели “мнимая”, то есть поверхность, по которой едет модель может свободно пересекаться с рамой и проходить сквозь неё).

После того, как все потомство одного поколения будет протестировано, оно породит новое потомство, дети которого будут наследовать от родителей биты параметров, вычисляемые с помощью процедур скрещивания и мутации. Вероятностные параметры процедур рекомбинации, мутации и скрещивания подбираются эмпирически.

2.2. Анимация механического устройства.

Исходя из условий задачи, имеется механическое устройство, состоящее из двух колес, груза и мнимой рамы. Устройство движется по некоторой поверхности. Взаимодействие происходит между двумя объектами: парой колес и поверхностью. Исключительная ситуация - груз касается поверхности.

Полагая, что поверхность и колеса имеют абсолютную жесткость, можно рассчитывать физику их взаимодействия, основываясь на импульсах тел. В двухмерной анимации все значения импульсов и нормалей к поверхности представлены в виде векторов.

В каждый момент времени на механическое устройство действуют несколько сил: сила притяжения, движущая сила и сила реакции опоры поверхности. Каждое колесо может находиться на разных участках поверхности, а значит и силы на них действуют разные. В то же время, рама компенсирует смещение колес друг относительно друга.

У каждого колеса всегда есть вектор импульса. Он показывает расстояние и направление на которое должно сдвинуться колесо в следующий момент времени. У каждого участка поверхности есть нормаль, показывающая угол наклона этого участка относительно горизонта.

Для обработки столкновения колеса с поверхностью следует вычислить скалярное произведение двух векторов - импульса колеса и нормали к поверхности (1).

$$dP = v \bullet N \qquad (1)$$

где

dP - проекция импульса колеса на нормаль к поверхности

v - вектор импульса колеса

N - вектор нормали к поверхности

Если расстояние между центром колеса и началом нормали к поверхности меньше радиуса и $dP < 0$, значит импульсы движения тел направлены друг к другу и произошло столкновение. Для разрешения этой ситуации положим, что масса поверхности бесконечно велика и она остается недвижимой, а значит при столкновении меняется только импульс колеса. Из физики известно, что по закону отражения новое значения импульса колеса вычисляется по формуле (2).

$$R = v - (1+e)*N*(v \bullet N) \qquad (2)$$

где

R - вектор отражения

e - коэффициент упругости

N - вектор нормали к поверхности

(v•N) - скалярное произведение импульса колеса и вектора нормали

Коэффициент упругости *e* в формуле показывает потерю энергии при столкновении. В идеальной ситуации $e = 1$, то есть это абсолютно эластичное колесо, так называемый идеальный мяч. Но в реальном мире такого быть не может, поэтому, учитывая потерю энергии, этот коэффициент эмпирическим путем принят равным 0.6.

Помимо силы реакции опоры на колесо действуют еще две силы - сила притяжения ($g = 9.8$ *м/с²)* и сила, приводящая модель механического устройства в действие (импульс, толкающий оба колеса вперед).

Для применения силы притяжения к колесу следует в каждый момент времени отнимать от значения ординаты вектора импульса колеса значение равное 9.8 / кол-во моментов времени в секунду.

Последней силой, действующей на наше колесо является рама устройства (балка), которая не дает двум колесам изменять расстояние друг относительно друга. Т.к. вся физика в программе основана на импульсах, то и это ограничение нужно преобразовать в термины импульсов тел.

Изменение расстояния между колесами может спровоцировать только ненулевое значение разности проекций векторов на ось, проложенную между центрами двух колес. Иными словами - следует вычислить проекции обоих векторов (для двух колес) на прямую, проходящую через центры окружностей (колеса имеют форму круга) и вычислить разность между ними.

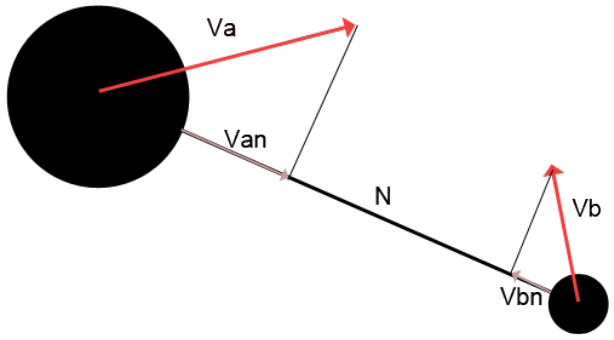

Рис. 6. Проекции импульсов

Как видно из рис. 6, два импульса Va и Vb спроецированы на прямую, лежащую между центрами колес. Вектора проекций направлены друг к другу, а значит в следующий момент времени колеса сблизятся друг относительно друга, что недопустимо. Значит, мы должны удалить из векторов эту компоненту, которая изменяет расстояние между центрами колес.

Таким образом, для моделирования устройства необходимо вычислять на каждом временном интервале четыре величины, влияющие на положение колес и автомобиля в целом

1) сила притяжения

2) движущая сила

3) импульсы столкновений

4) вычисление проекции импульса на нормаль к раме

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ЗАДАЧИ

3.1. Реализация анимации

Приложение разрабатывалось на языке C# под платформу .NET 3.5. Основной используемой технологией является WinForms. Библиотека WinForms использует функции библиотеки WinAPI, что в свою очередь означает, что вся графика обсчитывается программно, то есть на процессоре. Библиотека WinForms была разработана более десяти лет назад и являлась оберткой над Microsoft Foundation Class Library в управляемом коде. В настоящий момент существует более новая версия графического API для Windows, называемая WPF, которая полностью работает на DirectX и использует графический процессор для обсчета графики. Но WinForms - более универсальная и поддерживаемая технология, перенесенная под все операционные системы и работающая на всех версия операционной системы Windows.

Несмотря на отсутствие поддержки графического процессора, анимация обсчитывается со скоростью не менее 60 кадров в секунду даже на компьютере с одноядерным процессором.

В основе приложения лежит слой спрайтовой анимации. Он поддерживает два вида спрайтов: спрайт из растрового изображения и спрайт, построенный по точкам полигона. Иерархия классов спрайтов выглядит следующим образом:

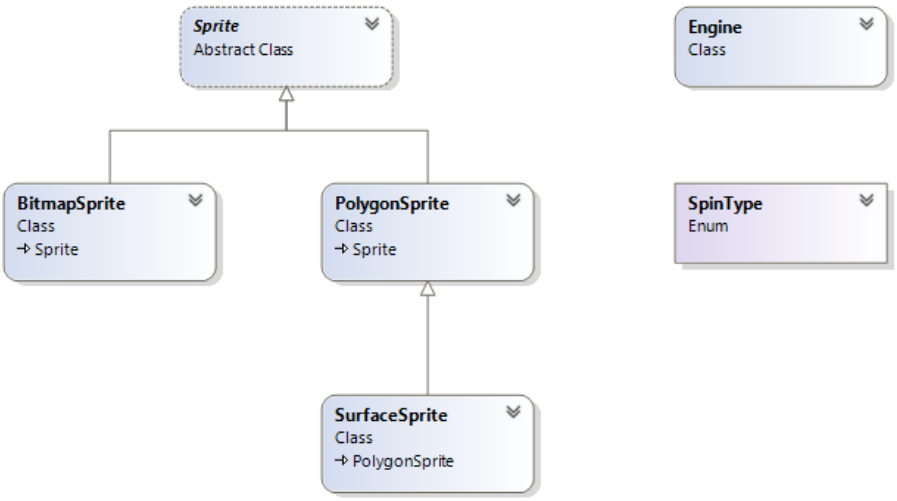

Рис. 7. Иерархия классов слоя-аниматора

В системе есть абстрактный класс *Sprite* с несколькими полями, от которого наследуются два типа спрайтов: *BitmapSprite* и *PolygonSprite.* От *PolygonSprite* наследуется класс *SurfaceSprite,* служащий для отображения поверхности, по которой движется модель механического устройства (т.к. поверхность представляет из себя набор точек, соединенных отрезками). Также в проекте присутствует класс *Engine*, с помощью которого и происходит анимация движения спрайтов и обработка их взаимодействия.

Аниматор работает по следующему алгоритму:

1) Создается холст на форме
2) На холст помещаются спрайты с некоторым значением скорости и направления движения
3) Процесс анимации отрабатывает каждый заданный интервал времени по таймеру (100 раз в секунду), задавая всем спрайтам новые координаты относительно их текущего положения и вектора скорости
4) Просчитываются столкновения и новые значения импульсов
5) Перерисовывается холст с применением технологии двойного буферинга для исключения мерцания

3.2. Реализация ГА

Схема классов проекта с реализацией ГА представлена на рис.8.

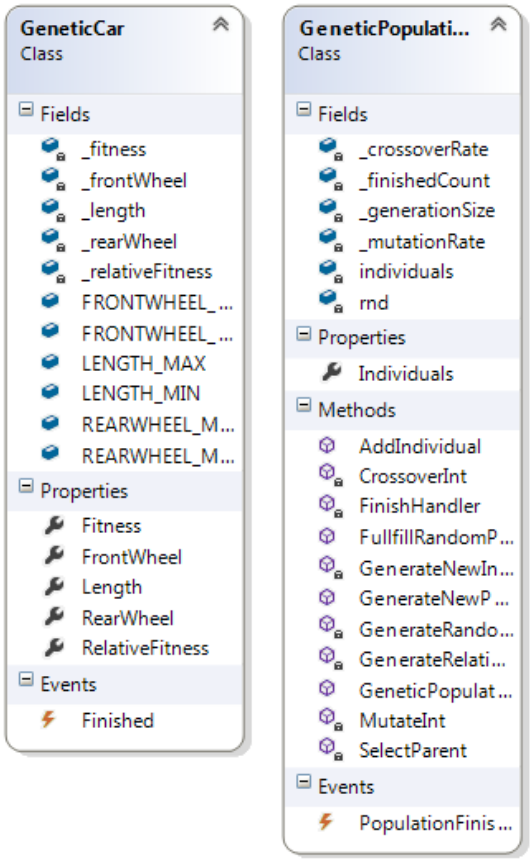

Рис. 8. Схема классов ГА

Класс *GeneticCar* представляет из себя набор полей со свойствами модели механического устройства, включая изначально заданные константы для определения минимальных и максимальных значений. Также класс содержит событие *Finished,* которые будет вызвано как только сработает условие остановки.

Класс *GeneticPopulation* отвечает за просчет алгоритма. В его публичном свойстве *Individuals* хранятся индивидуумы текущего поколения.

Алгоритм работы ГА следующий:

1) Инициализируется первое поколение случайным образом

2) Каждый индивид поколения тестируется

3) Когда все индивиды поколения будут протестированы, вызовется событие *PopulationFinished*

4) Из текущего поколения класс *GeneticPopulation* начнет создавать новое поколение, вызывая метод *GenerateNewIndividual()* ровно столько раз, сколько индивидов требуется в новом поколении.

5) Метод *GenerateNewIndividual()* создает экземпляр *GeneticCar,* присваивая его свойствам значения на основе работы методов селекции, скрещивания и мутации битов значений родительских экземпляров.

6) Когда количество индивидов будет равно нужному, цикл повторяется с пункта 2.

Метод *GenerateNewIndividual()* отберет двух родителей из поколения, затем применит к ним оператор скрещивания, вызвав метод *CrossoverInt()*. Данный метод принимает на вход два целых числа (int), возвращая одно целое число. Входные числа переводятся в битовое представление. Затем в выходное значение заносятся биты от родителей (случайным образом выбирая каждый бит от одного из родителей). Полученное битовое представление переводится в целочисленный тип. Далее, если будет сгенерирован вызов мутации, то вызывается метод *MutateInt()*. На вход ему подается одно целочисленное значение, на выходе возвращается также одно целочисленное значение. Как и в методе скрещивания, входное значение конвертируется в строку бит, далее с маленькой вероятностью биты могут быть инвертированы. Это привносит случайные мутации в значения свойств индивидуумов.

Во время выполнения алгоритма должно сработать одно из двух условий остановки:

1) в течение 5 секунд устройство не сдвинется с места

2) полезная нагрузка коснется земли

При срабатывании условия остановки, расстояние, пройденное индивидуумом, засчитывается за значение его фитнес-функции.

Из-за того что тестирование каждого индивидуума проходит довольно продолжительно, требуется большое количество времени для того, чтобы прийти к хорошим значениями индивидуумов. В среднем, каждое поколение тестируется в течение шести-семи минут, а для нахождения наиболее оптимального значения среди 20 индивидуумов требуется 20-30 поколений, что может занять около 100-200 минут.

ЗАКЛЮЧЕНИЕ

Разработанное приложение может использоваться для иллюстрации работы эволюционного алгоритма в задачах поиска и оптимизации.

Впоследствии, разработанная библиотека, реализующая генетический алгоритм на платформе .NET может быть абстрагирована от конкретной задачи с использованием *generic types,* что позволит применять ее для решения других задач.

Например, для задач реверсивного тестирования торговых роботов, работающих на фьючерсных и опционных рынках на Российской Торговой Системе. Подбор параметров и комбинаций индикаторов, на которых делает свое решение торговый робот разумно искать с помощью алгоритмического подхода, нежели вручную перебирать кажущиеся наиболее очевидными значения. Данный подход уже зарекомендовал себя на американских и европейских рынках ценных бумаг, но в России пока не получил широкого применения.

4. БИБЛИОГРАФИЯ

[1] Holland J. H. Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence / J. H. Holland. - University of Michigan Press. - 1975

[2] Dandy. G. C. An improved genetic algorithm for pipe network optimization / Dandy G.C., SImpson A., Murhpy L. // Water Resources Research. - 1996. - 32(2). - 449-458

[3] Goldberg D. E. Genetic Algorithms in Search, Optimization, and Machine Learning / Goldberg D. E. - Addison-Wesley. - 1989

[4] Piancastellia L. Revised KAD tool to optimize F1 cars through a combined-elitarian genetic-fuzzy algorithm / Piancastellia L., Frizzieroa L., Marcoppidoa S., Pezzutib E. // Engineering Sciences. - 2012. - 24(2). - 165–171

[5] Verma A. Scaling genetic algorithms using mapreduce / Llora X., Goldberg D., Campbell R. // Междунуародная конференция дизайна интелектуальных систем и приложений. - Пиза, Италия. - 2009.

[6] Edupuganti V. Authentication of JPEG Images Based on Genetic Algorithms / Edupuganti V., Shih F. // Multimedia Security and Stenography. - CRC Press. - 165-172