Аннотация

Двадцать первый век, ставший периодом быстрого развития городов, выявил потребность в разработке и использовании компьютерных программ и приложений, которые могут быть использованы в области градостроительства и территориального планирования.

Развитие города во многом зависит от устойчивости транспортной системы города, которая в свою очередь определяется правильной работой светофоров. Под правильной работой светофоров я подразумеваю оптимально рассчитанные интервалы времени, в течение которых горит зеленый свет для каждого из участников дорожного движения. Рассчитать такие интервалы в некоторых случаях (обычно в местах пересечения сразу нескольких автомобильных дорог и, возможно, плотного пешеходного потока) представляется достаточно сложной задачей, которую, тем не менее, можно решить методом имитационного моделирования.

Предметом исследования стала имитационная модель дорожного движения, на отдельных реально существующих в Москве дорожных развязках в период их наибольшей загруженности, которая возникает зачастую в результате нерациональной работы светофоров, а также неправильно выбранного места их расположения. Моя задача заключается в том, чтобы оптимизировать дорожное движение путем корректирования времени существующих светофоров и добавления новых.

Методом исследования стал метод агентного имитационного моделирования в графической среде моделирования AnyLogic.

Предложенный метод имитационного моделирования и последующего оптимизационного эксперимента в теории позволяет значительно сократить загруженность проблемных участков. Полученные мною результаты рекомендуем к использованию дорожно-транспортными службами.

Перспектива развития работы заключается в расширении объекта моделирования от отдельных участков дороги до масштабов района или целого города.

Содержание

ВВЕДЕНИЕ

Двадцать первый век, ставший периодом быстрого развития городов, выявил потребность в разработке и использовании компьютерных программ и приложений, которые могут быть использованы в области градостроительства и территориального планирования.

Развитие города во многом зависит от устойчивости транспортной системы города, которая в свою очередь определяется правильной работой светофоров. Под правильной работой светофоров подразумеваются оптимально рассчитанные интервалы времени, в течение которых горит зеленый свет для каждого из участников дорожного движения. Рассчитать такие интервалы в некоторых случаях (обычно в местах пересечения сразу нескольких автомобильных дорог и, возможно, плотного пешеходного потока) представляется достаточно сложной задачей, которую, тем не менее, можно решить методом имитационного моделирования.

Предметом исследования стала имитационная модель дорожного движения на отдельных реально существующих в Москве дорожных развязках в период их наибольшей загруженности, которая возникает зачастую в результате нерациональной работы светофоров, а также неправильно выбранного места их расположения.

Цель данной работы заключалась в попытке добавить в модель дорожного движения в существующее на данный момент место нерегулируемого пешеходного перехода светофор и найти оптимальный режим его работы.

Актуальность работы заключается в том, что на сегодняшний день во многих крупных городах существует серьезная проблема с загруженностью дорог, вызванной неоптимальным функционированием светофоров или их отсутствием. Данная работа предлагает один из возможных способов решения данной проблемы.

ИМИТАЦИОННАЯ МОДЕЛЬ

Постановка задачи и выбор метода ее решения.

Чтобы построить имитационную модель, а затем провести с ней оптимизационный эксперимент, был выбран участок дороги в Москве, на котором наблюдается проблема организации дорожного движения. Рассматриваемым участком стала дорога рядом с Черкизовским мостом (Восточный административный округ, район Преображенское, Большая Черкизовская улица, съезд с моста).

На этом участке дороге существует проблема организации дорожного движения, по нашему мнению, связанная с отсутствием светофора. В час пик наблюдается большое скопление людей, идущих от остановки, находящейся на мосту, ко входу в метро «Черкизовская», при этом пересекающих проезжую часть в месте съезда с моста. Из-за этого в часы наибольшей загруженности наблюдается скопление большого числа машин, в результате чего образуется пробка (рис. 1).

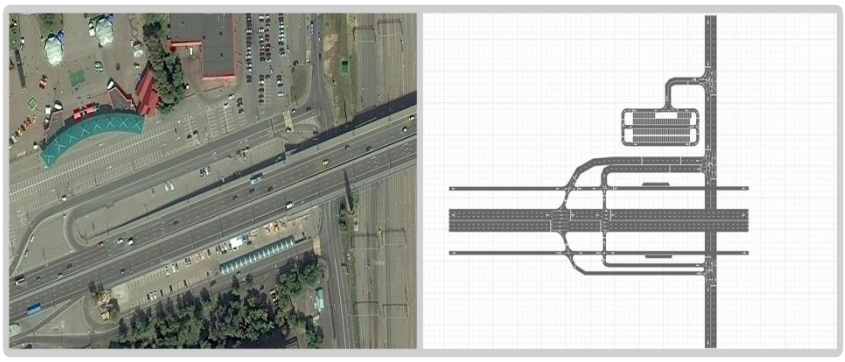

Рис. 1

Данную задачу можно решить следующими методами:

- метод подбора места и режимов работы светофоров сразу на практике без предварительного моделирования;
- аналитическое решение системы большого количества уравнений, описывающих всех участников движения;
- имитационное агентное моделирование.

Можно сразу сказать, что первый метод не будет нас устраивать, потому выбрать правильное место для светофора будет очень трудно, так как придется несколько раз переставлять светофор и многократно проверять ситуацию на дороге, чтобы найти нужное время работы светофора.

Вторым способом является аналитическое решение большого множества уравнений (математической модели), которые нам нужно будет долго решать, чтобы найти оптимальное время для светофора. Уравнения могут получиться сложными, и решать их придется несколько месяцев, поэтому это вариант нас не совсем устраивает.

Третий метод – метод имитационного агентного моделирования. *Имитационное моделирование* (англ. *simulation modeling*) — метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе. Такую модель можно «проиграть» во времени, как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику. Экспериментирование с моделью называют имитацией (имитация — это попытка постичь суть явления, не прибегая к экспериментам на реальном объекте) [1].

Под агентным моделированием понимается метод имитационного моделирования, исследующий поведение децентрализованных агентов и то, как оно определяет поведение всей системы в целом. Под агентами понимаются различные по своей природе элементы (например, социальные агенты: люди, социальные группы, покупатели; технические агенты: автомобили, роботы, самолеты). [1]

В агентном моделировании сначала устанавливаются параметры активных объектов (агентов) и определяется их поведение. В виде агентов может быть представлено что угодно, что имеет значение для исследуемой системы: люди, домохозяйства, автомобили, оборудование, даже продукты и

компании. Затем устанавливаются связи между агентами, задается окружающая среда и запускается моделирование. Индивидуальные действия каждого из агентов образуют глобальное поведение моделируемой системы.

Понятие агентного моделирования близко к объектно-ориентированному программированию. В ООП мы работаем с объектами, являющимися экземплярами классов. В агентном моделировании экземплярами классов являются агенты (агенты-машины, агенты-пешеходы), для которых задаются алгоритмы поведения в случаях взаимодействия с другими агентами и просто другими элементами модели (в нашем случае данные алгоритмы соответствуют действующим правилам дорожного движения).

Таким образом, имитационное моделирование — универсальный метод, обладающий следующими достоинствами:

1) позволяет легко решать сложные задачи;

2) дает возможность исследовать особенности функционирования реальной системы в разнообразных условиях;

3) существенно сокращает стоимость и продолжительность испытаний по сравнению с натурным экспериментом, то есть экономит ресурсы;

4) позволяет включать результаты натурных испытаний компонентов реальной системы;

5) позволяет достигать лучшие решения за счет гибкости и легкости варьирования структуры, алгоритмов и параметров;

6) является единственным практически реализуемым методом для исследования сложных систем. [2]

К недостатку можно отнести то, что каждое решение носит частный характер, так как оно соответствует фиксированным элементам структуры, алгоритмам, значениям параметров — требуется многократное повторение имитационного эксперимента при вариации исходных данных. Для автоматизации подобных процессов обычно прибегают к так называемому оптимизационному эксперименту. В качестве метода исследования был

выбран метод агентного имитационного моделирования, потому что этот вариант нам больше всего подходит, исходя из его особенностей. В качестве средства для построения модели использовалась среда имитационного моделирования AnyLogic.

AnyLogic — первый и единственный инструмент имитационного моделирования, объединивший методы системной динамики, "процессного" дискретно-событийного и агентного моделирования в одном языке и одной среде разработки моделей. Гибкость AnyLogic позволяет отражать динамику сложных и разнородных экономических и социальных систем на любом желаемом уровне абстракции. AnyLogic включает набор примитивов и библиотечных объектов для эффективного моделирования производства и логистики, бизнес-процессов и персонала, финансов, потребительского рынка, а также окружающей инфраструктуры в их естественном взаимодействии. Объектно-ориентированный подход, предлагаемый AnyLogic, облегчает итеративное поэтапное построение больших моделей. Плюсами данной среды являются:

- наличие бесплатной версии для учебного использования;
- наличие встроенных библиотек, позволяющих моделировать движение автомобилей и пешеходов, а также взаимодействие между ними;
- простота использования и легкость освоения;
- возможность проведения оптимизационного эксперимента.

По своей сути система AnyLogic представляет собой графическую надстройку над языком программирования Java, т.е. основные блоки модели добавляются в нее простым перетаскиванием из палитры инструментов, но многие частные задачи, характерные для конкретной модели, могут быть решены лишь написанием фрагментов программного кода. Таким образом, первой проблемой, с которой мы столкнулись, было освоение на элементарном уровне языка программирования Java и его сопоставление с изучаемым в школе языком Pascal.

Построение модели.

1. Для построения модели выбранного участка дороги был использован раздел «*Библиотека дорожного движения*» – «*Разметка пространства*».

2. Чтобы пустить автомобили по всей дорожной системе, используем тот же раздел, в котором нам понадобится блок «*Car Source*». В нем указываем место, откуда агент (в данном случае автомобиль) начнет свое движение. Чтобы указать это место, в блоке «*Car Source*» нужно выбрать название дороги, на которой будут появляться машины.

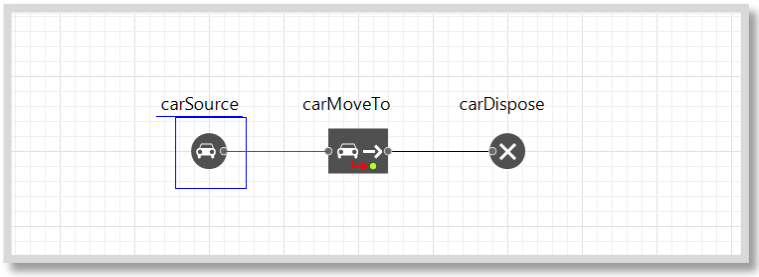

Рис. 2

Следующий блок, который нам нужен, называется «*Car Move to*». Этот блок отвечает за конечную точку автомобиля. В нем нужно установить название дороги, до которой должен будет двигаться автомобиль.

Если говорить простым языком, то автомобиль должен двигаться из пункта А, в котором он начинает свое движение, в пункт Б, в котором он заканчивает свое движение. «*Car Source*» устанавливает местонахождение пункта А, а «*Car Move to*» устанавливает местонахождение пункта Б.

Рассматриваемую схему заканчивает блок «*Car Dispose*», который удаляет автомобиль, достигший заданную цель (рис. 2).

Чтобы движение выглядело более реалистично, используем раздел «*Библиотека моделирования процессов*», в котором применим блок «Select Output». С помощью него можно создать несколько конечных пунктов движения автомобиля, один из которых он выберет сам. Выбор определяется с помощью вероятности, которая указывается пользователем (рис. 3).

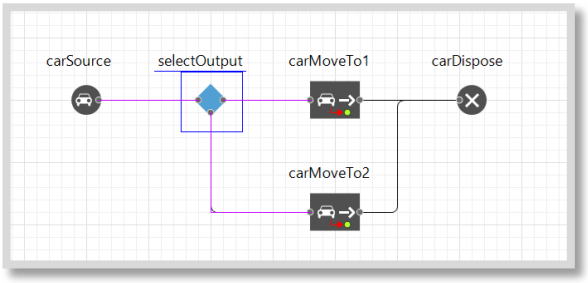

Рис. 3

Также, отдельно была создана имитация движения автобусов с помощью раздела «*Библиотека дорожного движения*», только в этот раз нам понадобилось использовать два блока «*Car Move to*»: в одном из них мы указываем автобусную остановку, которую мы поставили рядом с нужной нам дорогой, а в другом — пункт окончания движения.

Чтобы автобусы останавливались на автобусной остановке надо в разделе «*Библиотека моделирования процессов*» выбрать блок «*Delay*», в котором можно указать время остановки автобуса (рис. 4).

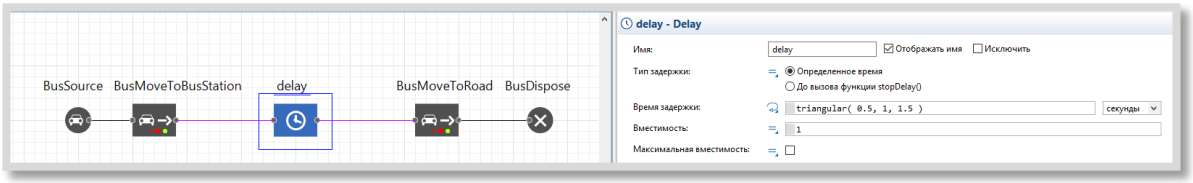

Рис. 4

Таким же способом можно сделать так, чтобы машины могли отправляться на парковку, предварительно создав парковочное место.

3. Следующим этапом станет создание движения пешеходов. Для этого понадобится раздел «*Пешеходная библиотека*». Блоки: «*Ped Source*», «*Ped Go To*», «*Ped Sink*», – аналогичны блокам: «*Car Source*», «*Car Move to*», «*Car Dispose*». Чтобы пешеходы ходили по определенной области, нужно использовать раздел «*Пешеходная библиотека*» – «*Разметка пространства*» – «*Прямоугольная стена*». Это функция создает стены для пешеходов, через которые они не могут проходить. Для использования блоков «*Ped Source*» и «*PedGoTo*» нужна «*целевая линия*». Эта линия указывается в блоках

движения пешеходов так же, как и название дороги в блоке «*Car Source*» (рис. 5).

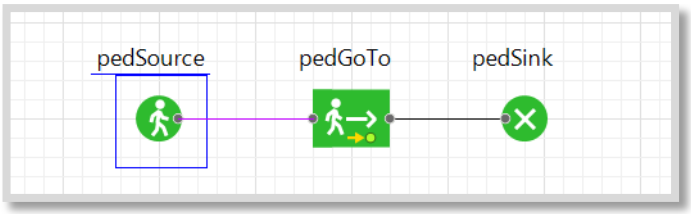

Рис. 5

4. На данном этапе построения модели мы столкнулись с определенной проблемой, заключавшейся в том, что пешеходы и автомобили в системе AnyLogic являются агентами из двух разных библиотек и, следовательно, ведут себя в процессе симуляции так, будто не замечают друг друга: автомобили едут сквозь переходящих дорогу пешеходов, не уступая им дорогу, а пешеходы в свою очередь не видят автомобили. Рассматриваемая проблема может быть решена следующим образом.

Для начала свяжем появление пешеходов с прибытием автобуса на остановку. Для этого перейдем в раздел «Библиотека моделирования процессов» и используем блок «*Split*». Этот блок разделяет логическую схему модели на два условия, но в отличие от «*Select Output*», будут выполняться оба условия, а не одно из них на выбор.

Поставим блок «*Split*» после блока движения, отвечающего за движение автобуса до остановки. После прибытия автобуса на остановку, идет выполнение двух условий: 1) условие остановки на определенное время, заданное блоком «*Delay*»; 2) условие выхода пассажиров из автобуса, которое будет задано блоком «*Ped Enter*» (это блок выполняется при выполнении предыдущего условия, а блок «*Ped Source*» независим от других условий, поэтому используем «*Ped Enter*»). Создадим «*целевую линию*» рядом с остановкой и укажем ее в блок «*Ped Enter*» . В блоке «*Split*» мы должны указать количество выходящих пассажиров. Для этого в поле *«Количество копий»* введем функцию «*uniform_discr(x,y)*» (где «*x*» - минимальное количество людей, выходящих из автобуса, а «*y*» - максимальное количество

людей). После этого мы можем использовать остальные блоки, такие как «*PedGoTo*» и другие. Таким образом, пешеходы появляются на автобусной остановке только тогда, когда на нее приезжает автобус (рис. 6).

Рис. 6

Последним шагом объединения пешеходов с автомобилем является создание правила дорожного движения, которое заключается в том, что автомобили должны пропускать пешеходов, находящихся на проезжей части.

В начале создадим две «*стоп-линии*» на участке дороги, где пешеходы переходят проезжую часть. «*Стоп-линия*» в данной библиотеке играет роль, как стоп-линии, так и светофора.

Далее создадим на дороге для автомобилей «*прямоугольную область*», которая находится в разделе «*Пешеходная библиотека*». В любое место модели поместим блок «*Ped Area Description*», который будет управлять заданной *областью*, и, перейдя в свойства данного элемента, пропишем программный код в поле «*При входе*»:

```
{
        pedestrians1.setSignal(SIGNAL_RED);
        pedestrians2.setSignal(SIGNAL_RED);
}
```

(где *pedestrians1* и *pedestrians2* – названия стоп-линий)

Эта команда задает условие: если в заданную область входит пешеход, то мы закрываем проезд автомобилю.

В поле «*При выходе*» запишем следующее условие:

```
if (self.size()==0)
{
        pedestrians1.setSignal(SIGNAL_GREEN);
        pedestrians2.setSignal(SIGNAL_GREEN);
}
```

Это команда задает условие: если в заданной области нет пешеходов, то область можно открыть.

Таким образом, мы создали дорожную сеть с нерегулируемым пешеходным переходом.

5. Последним этапом перед оптимизацией является создание переключателя, который будет превращать нерегулируемый пешеходный переход в регулируемый и обратно.

Установим светофор, добавив блок «*Traffic Light*» из раздела «*Библиотека дорожного движения*» (рис. 7).

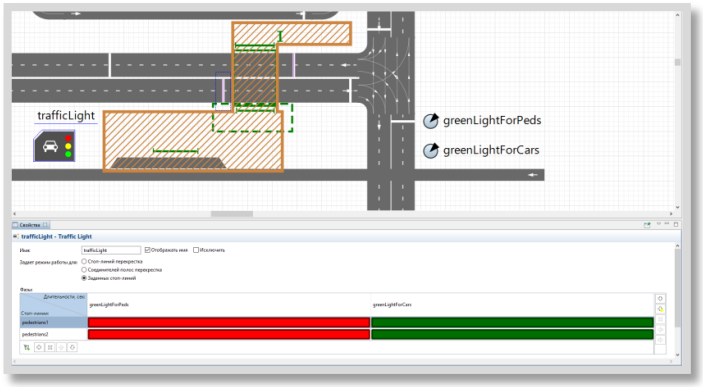

Рис. 7

Этот блок будет задавать режим работы для созданных нами стоп-линий, которые мы указываем в светофоре. Также, нам нужно будет добавить два параметра, которые можно взять в разделе «*Агент*». Назовем их «*greenLightForPeds*» и «*greenLightForCars*» (тип – «int», начальное значение 10 и 60 соответственно).

В том же разделе добавим «*переменную*», которая будет отвечать за включение и выключение светофора. Назовем ее «*light*», в качестве типа укажем «*int*», а начальное значение зададим равным *1*. Теперь добавим «*Переключатель*» из раздела «*Элементы управления*» (рис. 8). В нем создадим два элемента: нерегулируемый переход; регулируемый переход. Свяжем «*Переключатель*» с переменной «*light*». Он будет менять значение этой переменной, которая в свою очередь будет определять режим работы светофора. В поле «*Действие*» добавим следующий программный код:

```
if (light==0)
{
        trafficLight.turnOff();
        pedestrians1.setVisible(false);
        pedestrians2.setVisible(false);
}
if (light==1)
{
        trafficLight.turnOn();
        pedestrians1.setVisible(true);
        pedestrians2.setVisible(true);
}
```

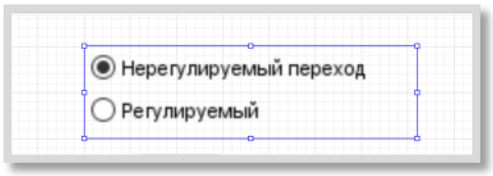

Рис. 8

Теперь при смене значения переменной «*light*» светофор будет включаться или выключаться.

В блоке «*Traffic Light*», в поле «*Действие*» напишем программный код:

```
if (light == 1)
        pedAreaDescriptor.setOpen (currentPhaseIndex == 0);
if (light == 0)
        pedAreaDescriptor.setOpen(true);
```

При нерегулируемом светофоре пешеходы могут переходить дорогу, когда они захотят, но при регулируемом светофоре осуществить переход можно строго по сигналу светофора. Поэтому при значении переменной «*light*» равном *1* переход пешеходов разрешается только по сигналу светофора, а при значении *0* переход разрешен всегда.

Еще были добавлены два «*бегунка*» из раздела «*Элементы управления*», с помощью которых можно менять время работы красного (зеленого света для пешеходов) и зеленого света светофора при запуске имитационной модели. Для этого «*бегунки*» нужно связать с переменными, отвечающими за длительность горения определенного сигнала светофора

(«*greenLightForPeds*», «*greenLightForCars*»), и задать минимальное и максимальное значения. Пусть значение «*greenLightForPeds*» может меняться в пределах от 10 до 25 секунд, а для «*greenLightForCars*» — от 40 до 120 секунд (рис. 9).

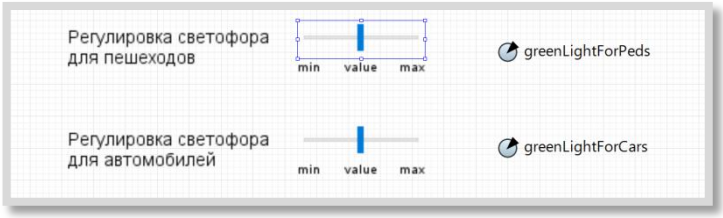

Рис. 9

ОПТИМИЗАЦИОННЫЙ ЭКСПЕРИМЕНТ

Постановка задачи оптимизации.

Очевидно, что выставить оптимальный режим работы светофора для того, чтобы автомобили могли проезжать данный участок дороги за кратчайшее время, очень трудно. Мы можем выделить большой промежуток времени для проезда автомобилей и немного времени для перехода дороги пешеходами, но тогда на пешеходном переходе будет скапливаться большое количество людей. Если же мы дадим автомобилям мало времени для проезда, то может образовываться пробка. Наша задача заключается в том, чтобы найти оптимальное время работы светофора, при условии, что на пешеходном переходе не должно скапливаться людей больше определенного количества. Чтобы решить данную задачу мы будем проводить оптимизационный эксперимент.

Выбор целевой функции.

Проанализировав схему движения автомобилей, найдем пути, проходящие через нужную нам дорогу. В нашей схеме есть 4 пути. При попадании на нужную нам дорогу используем блоки «*Time Measure Start*» и «*Time Measure End*», находящиеся в разделе «*Библиотека моделирования*

процессов», и поставим их до и после блоков «Car Move to», которые отправляют автомобиль на нужную нам дорогу. Блоки «*Time Measure Start*» и «*Time Measure End*» будут считать среднее время проезда машин на данном участке. В каждом блоке «*Time Measure End*» в строчке «*Объект Time Measure Start*» укажем соответствующие блок, а в поле «*При входе*» напишем программный код:

```
averageTime1=timeMeasureEnd1.distribution.mean();
n1++;
totalAverage=(averageTime1*n1+averageTime2*n2+averageTime3*n3+averageTime4*n4)/(n1+n2+n3+
n4);
```

(этот код для блока «Time Measure End1», аналогичные коды должны быть и в других блоках «Time Measure End», только нужно изменить цифры для «averageTime», «timeMeasureEnd» и «n»)

Перед этим нам нужно создать новые переменные: 1) переменные (*n1-n4)*, которые будут записывать количество проехавших машин, присвоим им тип «*int*» и начальное значение *0*; 2) переменные (*averageTime1-averageTime4*), которые будут считать среднее время проезда машин, движущихся по определенной полосе (тип — «*double*» и начальное значение *1000*)

Теперь нам нужно посчитать общее среднее время проезда автомобилей на данном участке дороги для всех машин. Для этого создадим новую переменную и назовем ее «*totalAverageTime*». Формулу расчета значения данной переменной мы записали в блоки «*Time Measure End*».

Последнее, что нам нужно будет создать, это счетчик количества людей, ожидающих разрешения на переход. Для этого создадим новую область «*Прямоугольная область*» и переменную, в которой будет храниться максимальное количество людей, одновременно ждавших зеленного сигнала светофора (тип – «*int*», начальное значение – *0*), и добавим новый блок «*Ped Area Description*». В блоке выберем область, которую мы создали, в полях действия «*При входе*» запишем условие:

```
if(pedAreaDescriptor2.size()>NumberOfPeds)
        NumberOfPeds=pedAreaDescriptor2.size()
```

«При выходе»:

```
if(pedAreaDescriptor2.size()>NumberOfPeds)
        NumberOfPeds=pedAreaDescriptor2.size()
```

Теперь в переменной будет запоминаться максимальное количество людей, одновременно ожидавших зеленого сигала светофора, за все время работы модели.

Подготовка оптимизационного эксперимента.

Теперь можно приступать к созданию самого эксперимента. Для этого необходимо щелкнуть правой кнопкой в левом окне на названии модели и выбрать «*Создать*» – «*Эксперимент*» – «*Оптимизация*».

Для проведения эксперимента необходимо указать целевую функцию, которую мы хотим минимизировать или максимизировать. В нашем случае в качестве целевой функции будет рассматриваться переменная «*totalAverageTime*». Поэтому на вкладке *Основные* свойств эксперимента в поле *Целевая функция* укажем значение «*root.totalAverageTime*». Указатель *root* необходим, так как параметр расположен не в окне оптимизационного эксперимента, а в активном классе модели. Чуть ниже зададим изменение двух параметров «*greenLightForCar*» и «*greenLightForPed*». Параметр «*greenLightForCar*» будет изменяться от 40 до 89 с шагом 1, а «*greenLightForPed*» от 10 до 19 с шагом 1. Это означает, что эксперимент будет выполняться снова и снова, но время работы светофора будет меняться, и для каждого нового значения параметров будет измеряться среднее время проезда машин, то есть переменная «*totalAverageTime*».

При подготовке к оптимизационному эксперименту возникла проблема с указанием количества итераций. Данная опция определяет, сколько раз будет запускаться имитационная модель с разными значениями параметров «*greenLightForPed*» и «*greenLightForCar*», для нахождения наименьшего значения оптимизируемой функции. Используемая версия бесплатная версия

AnyLogic предназначена только для обучения и поэтому максимальное количество итераций ограничено числом 500. По этой причине был выбран следующий диапазон изменяемых параметров: 40-89 для «*greenLightForCar*» и 10-19 для «*greenLightForPed*», т.е. для «*greenLightForCar*» существует 50 различных значений, а для «*greenLightForPed*» – 10. Таким образом, для перебора всех возможных комбинаций значений рассматриваемых параметров модель необходимо запустить ровно 500 раз.

Суть нашего эксперимента заключается в том, что необходимо найти то допустимое время работы светофора в час пик, при котором среднее время проезда машин будет минимальным, а количество людей, ожидающих разрешение на переход, не должно превышать определенного значения. Величина, которой мы ограничиваем количество пешеходов, ждущих разрешения перейти дорогу, в нашем ограничивается размерами тротуара вблизи перехода. Замеры, проведенные на рассматриваемом участке, показали, что число ожидающих пешеходов не должно превышать 25. Перейдем во вкладку «*Требования*» и потребуем, чтобы значение «*root.NumberOfPeds*» в процессе эксперимента не превышало 25.

Часы пик в Москве являются часами с 8:00 до 9:00 и с 17:00 – 18:00. Поэтому и мы ограничим время выполнения эксперимента примерно 1 часом. Для этого перейдем во вкладку «*Модельное время*» и укажем «*Начальное время*» – «0.0», а «*Конечное время*» – «*3599.0*».

После этого возвращаемся во вкладку «*Основные*» и нажимаем на кнопку «*Создать интерфейс*».

Расчет оптимального режима работы светофора.

Обратите внимание, что теперь, если щелкнуть на кнопке запуска модели, для запуска будут доступны два эксперимента. Один из них – «*Simulation*» – простой эксперимент по симуляции модели, который мы запускали ранее. Второй же эксперимент и есть оптимизационный. Запустим его.

Перед нами откроется новая форма, в которой щелкнем на кнопку «Запустить оптимизацию». На рисунке 10 показан результат выполнения такого эксперимента. На графике справа отображена зависимость значений оптимизируемого параметра от номера итерации (одна итерация одно выполнение эксперимента при определенных значениях «перебираемого» параметра «*greenLightForCar*» и «*greenLightForPed*»). Причем красным отображено лучшее недопустимое значение, т.е. значение, полученное без учета требований, наложенных на оптимизируемую модель. А синим - лучшее допустимое. Мы видим, что с увеличением количества итераций, значение оптимизируемого параметра стремится к 34,5. Это же значение показано в окошке, слева от графика (Функционал – 34.746). Следовательно, для заданных нами начальных условий наименьшее возможное среднее время проезда машин данного участка дороги будет приблизительно равно 35 секундам, при этом зеленый сигнал светофора для машин должен будет гореть 61 секунд, а красный свет – 10 секунд.

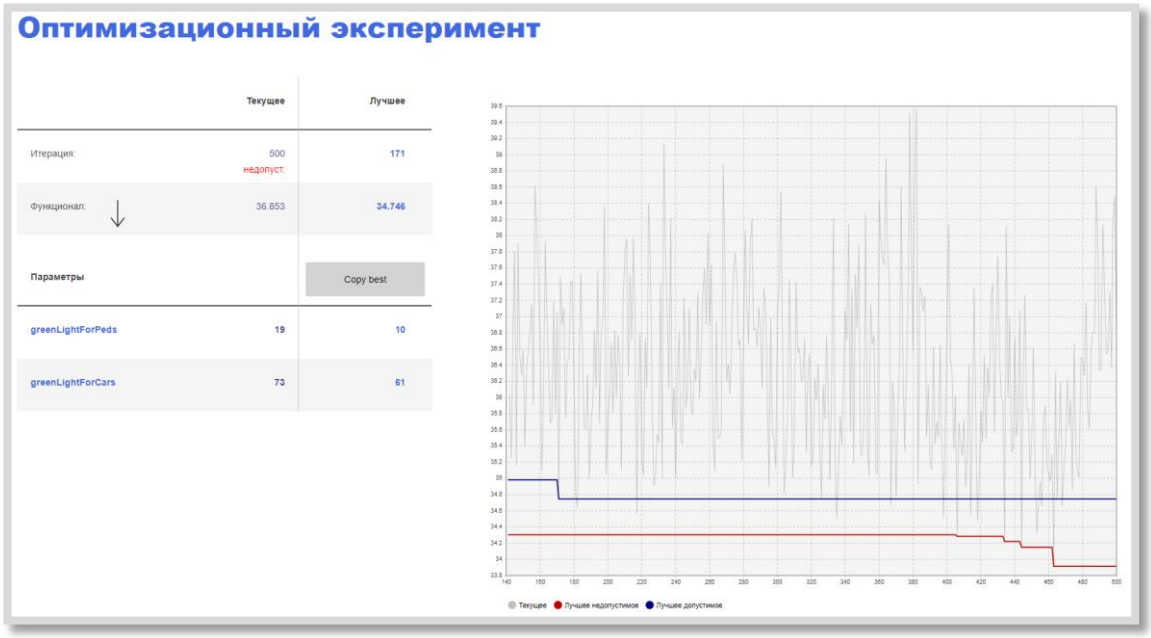

Рис. 10

ЗАКЛЮЧЕНИЕ

1. Была построена модель существующего участка дороги с нерегулируемым пешеходным переходом, для чего были произведены измерения интенсивностей потока машин и пешеходов в часы пик.

2. Было измерено среднее время, затрачиваемое автомобилями на прохождение рассматриваемого участка дороги в случае отсутствия светофора. Это время оказалось приблизительно равным 250 секундам.

3. Был добавлен в модель светофор и проведен оптимизационный эксперимент, в результате которого было установлено оптимальное время «зеленого» и «красного» сигналов для автомобилей. В среднем зеленый сигнал светофора для автомобилей должен гореть 70 секунд, а красный – 12 секунд.

4. Таким образом, эксперимент показал, что добавление светофора с оптимально выбранным режимом работы сокращает время движения машин по рассматриваемому участку приблизительно в 7 раз.

5. Результаты нашего исследования можем рекомендовать дорожным службам, занимающимся оптимизацией дорожного движения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сайт AnyLogic: [Электронный ресурс]. URL: https://www.anylogic.ru. (Дата обращения: 17.02.2019).

2. Сайт Образовательный блог – все для учебы: [Электронный ресурс]. URL https://all4study.ru (Дата обращения: 17.02.2019).

3. Борщов А. Java for Anylogic users. Интернет издание: 2016.

4. Григорьев И. AnyLogic за три дня. Практическое пособие по имитационному моделированию. Интернет издание: 2016.

5. Калугин А.И. Оптимизационный эксперимент в среде Anylogic. Наука и школа. — 2015 — №4.— С. 168.