ЛОГИСТИКА

УДК 004.032.26 **DOI: 10.31799/2077-5687-2022-2-22-41**

СРАВНИТЕЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ И ОПТИМИЗАЦИИ ПРИ УПРАВЛЕНИИ ТРАНСПОРТНЫМИ ПРОЦЕССАМИ В СРЕДЕ ANYLOGIC

С. А. Андронов, М. С. Прокофьева
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Задача повышения пропускной способности участков транспортной сети мегаполиса при существующей инфраструктуре решается средствами автоматизированных систем управления дорожным движением, целью которых является формирование управляющих воздействий на объекты транспортной системы в реальном масштабе времени. Адекватная реакция на изменение нагрузки в транспортной сети реализуется за счет управления светофорными объектами с использованием встроенных адаптивных алгоритмов, в числе которых все шире применяются технологии искусственного интеллекта, в частности, нейросетевые. Отмеченные подходы конкурируют с широко применяемому методу адаптации на основе оптимизации (длительности фаз, смещению и т. д.).

В предлагаемой статье исследуется вопрос о сравнении эффективности оптимизационного алгоритма с алгоритмов машинного обучения с подкреплением по критерию времени нахождения транспортных средств в системе перекрестка в среде имитационного моделирования Anylogic. Данное исследование поможет определить более эффективный вариант решения проблемы задания длительности фаз светофорного регулирования.

Было показано, что в алгоритме обучения с подкреплением способность адаптации к входным интенсивностям в пределах расписания выше по сравнению с оптимизационным алгоритмом. Однако алгоритм с подкреплением в большей степени чувствителен к типу расписания по сравнению с оптимизационным, который превзошел последний примерно на 20% в оптимальной точке для расписания в будни. Преимущество алгоритма с подкреплением сильнее проявилось на 2-м расписании, особенностью которого является тенденция к нарастанию интенсивности движения, а именно: 61% по сравнению с оптимизационным и 70% по отношению к базовой настройке. К «расстройке» входных данных относительно оптимальной политики он оказался практически нечувствителен при изменении уровней интенсивностей в пределах этого расписания.

Таким образом, было показано, что результаты регулирования транспортным процессом на исследуемом реальном перекрестке, полученные при моделировании с использованием обучения с подкреплением превосходят оптимизационный подход, однако являются чувствительны к заданному расписанию интенсивностей.

Ключевые слова: обучение с подкреплением, оптимизация, искусственные нейронные сети, AnyLogic.

***Для цитирования**:*
Андронов С. А., Прокофьева М. С. Сравнительная оценка эффективности алгоритмов машинного обучения с подкреплением и оптимизации при управлении транспортными процессами в среде Anylogic // Системный анализ и логистика: журнал.: выпуск №2 (32), ISSN 2077-5687. – СПб.: ГУАП., 2022 – с. 22–41. РИНЦ. DOI: 10.31799/2077-5687-2022-2-22-41.

COMPARATIVE EVALUATION OF THE EFFICIENCY OF REINFORCEMENT MACHINE LEARNING ALGORITHMS AND COMPARATIVE EVALUATION OF TRANSPORT PROCESSES IN THE ENVIRONMENT ANYLOGIC

S. A. Andronov, M. S. Prokofieva
St. Petersburg State University of Aerospace Instrumentation

The task of increasing the throughput of sections of the transport network of the metropolis with the existing infrastructure is solved by means of automated traffic control systems, the purpose of which is to form control actions on the objects of the transport system in real time. An adequate response to load changes in the transport network is implemented by controlling traffic light objects using built-in adaptive algorithms, among which artificial intelligence technologies, in particular, neural networks, are increasingly being used. The noted approaches compete with the widely used optimization-based adaptation method (phase duration, displacement, etc.).

The proposed article examines the issue of comparing the efficiency of an optimization algorithm with reinforcement machine learning algorithms by the criterion of the time spent by vehicles in the intersection system in the Anylogic simulation environment. This study will help determine a more efficient solution to the problem of setting the duration of traffic light control phases.

It was shown that in the reinforcement learning algorithm, the ability to adapt to input intensities within the schedule is higher compared to the optimization algorithm. However, the reinforcement algorithm is more sensitive to the type of schedule than the optimization algorithm, which outperformed the latter by about 20% at the optimal point for the weekday schedule. The advantage of the reinforcement algorithm was more pronounced on the 2nd schedule, which features a tendency to increase traffic intensity, namely: 61% compared to the optimization and 70% compared to the base setting.

It turned out to be practically insensitive to the "detuning" of the input data relative to the optimal policy when changing the intensity levels within this schedule.

Thus, it was shown that the results of the regulation of the traffic process at the studied real intersection, obtained by modeling using reinforcement learning, are superior to the optimization approach, but are sensitive to the given intensity schedule.

Keywords: machine learning, optimization method, neural networks, artificial intelligence.

For citation:

Andronov S. A., Prokofieva M. S. Comparative Evaluation of the Efficiency of Reinforcement Machine Learning Algorithms and Comparative Evaluation of Transport Processes in the Environment // Systems analysis and logistics: №2 (32), ISSN 2077-5687. – Russia, SaintPetersburg.: SUAI., 2022 – p. 22–41. DOI: 10.31799/2077-5687-2022-2-22-41.

Введение

Повышение пропускной способности транспортной сети мегаполиса в условиях постоянно возрастающей транспортной нагрузки - важная задача при создании современных интеллектуальных транспортных систем (ИТС). Одним из путей балансировки нагрузки транспортной сети является внедрение автоматизированных систем управления дорожным движением (АСУДД), задача которых состоит в повышении эффективности регулирования дорожного движения.

Перспективным направлением в борьбе с неравномерностью транспортных потоков являются системы транспортно-зависимого управления, способные в режиме онлайн подстроиться под текущую дорожную ситуацию. При этом, например, автономный светофорный объект с адаптивным алгоритмом управления способен самостоятельно определять динамику включения сигнальных групп.

При исследовании адаптивных алгоритмов управления на транспорте важную роль играет имитационное моделирование, поскольку процессы в транспортных системах имеют принципиально вероятностных характер.

Сравнительному анализу адаптивных алгоритмов посвящено множество работ, например [1-5]. Все шире встречаются исследования в части адаптивного управления с использованием элементов искусственного интеллекта (ИИ). В случае дорожного регулирования это позволяет гибко приспособить задачу к особенностям трафика конкретного перекрестка. Так, вариант адаптации на основе нечеткой логики, рассмотренный в [2], исследует применение «мягкого» регулирования, при котором дискретные уровни интенсивностей движения представляются непрерывной величиной. При этом функции принадлежности нечеткой интеллектуальной системы можно интерпретировать как функции активации нейронов в нейронной сети. В работе [3] было показано, что из перечня рассмотренных адаптивных алгоритмов наиболее результативным является оптимизационных подход. В работе [4] исследуется алгоритмы управления на основе искусственных нейронных сетей (ИНС).

Исследования, производящиеся в данной публикации, в большей степени основаны на работе, где рассматривается схожая ИМ, однако в [5] акцент сделан на демонстрации технологии.

Настоящая статья посвящена исследованию в части сравнения эффективности одного из направлений машинного обучения – обучения с подкреплением (ниже алгоритм rl) с лидирующим сейчас оптимизационным подходом (далее opt) к управлению перекрестком. Также затронут вопрос чувствительности названных алгоритмов к изменению расписания интенсивностей в течение дня.

В качестве примера в работе рассматривался простой четырехсторонний перекресток с типом движения север-юг и запад-восток, без возможности поворотов, перестроений и т. д. Так же, следует отметить, что в отличии от данной статьи в [5], режим работы светосигнальной установки (ССУ) задается через стоп-линии, что позволяет использование более упрощённой схемы дорожного движения.

Исследование выполняется на имитационной модели (далее ИМ) реального т-образного перекрестка в г. Санкт-Петербург, имеющего проблемы с переключением светофорных фаз (рис. 1), режим работы ССУ задается при помощи задания направления полос движения (рис.9).

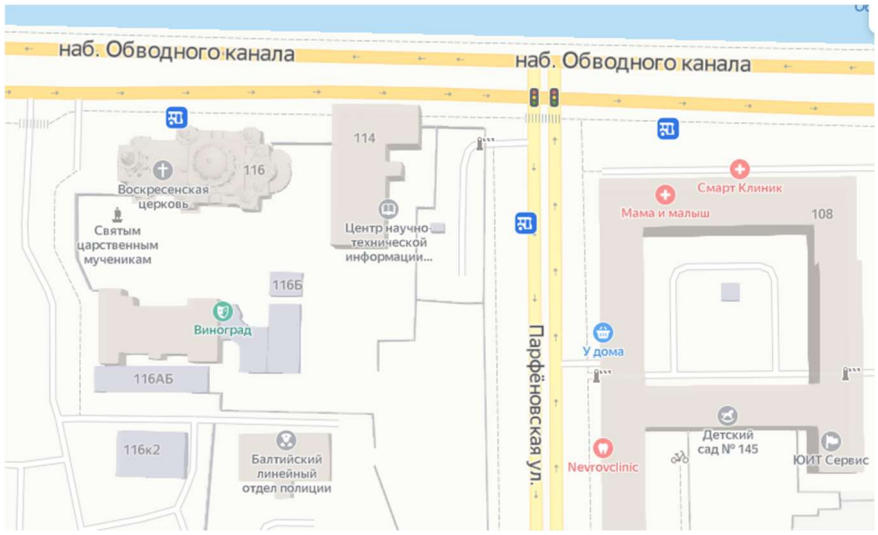

Рис. 1. Перекресток наб. Обводного канала – ул. Парфеновская

Предполагается, что данное исследование поможет определить более эффективный механизм управления длительностью фаз светофорного регулирования по сравнению с существующим.

Построение модели

Логика дорожного движения и перекрестка моделируется с помощью стандартных блоков в AnyLogic Road Traffic Library (RTL) [10] или библиотеки дорожного движения. Геометрия дорог и перекрестков моделируется с помощью пространственных разметок из RTL.

Результаты измерения базовых параметров:

a) Длина главных отрезков, выполняющих функцию перегонов до и после перекрестка, $L_{ГЛ_1} = 160 м$, $L_{ГЛ_2} = 114 м$:

b) Количество полос, со стороны юга («S»), $n_{SE} = n_{WS} = 1 => n_{SE\_WS} = 2$.Количество полос, для направлений восток («E») и запад («W»): $n_{EW} = 2$, $n_{ES} = 1 => n_{EW\_ES} = 3$; $n_{WE} = 2$, $n_{WS} = 1 => n_{WE\_WS} = 3$.

c) Ширина всех полос принимается равной $L_{ШП} = 3,5 м$.

d) Масштаб растровой основы задается путем измерения края дома = 30м (см. Рис. 2).

Соответственно, после произведения всех вышеперечисленных замеров, была составлена итоговая ИМ:

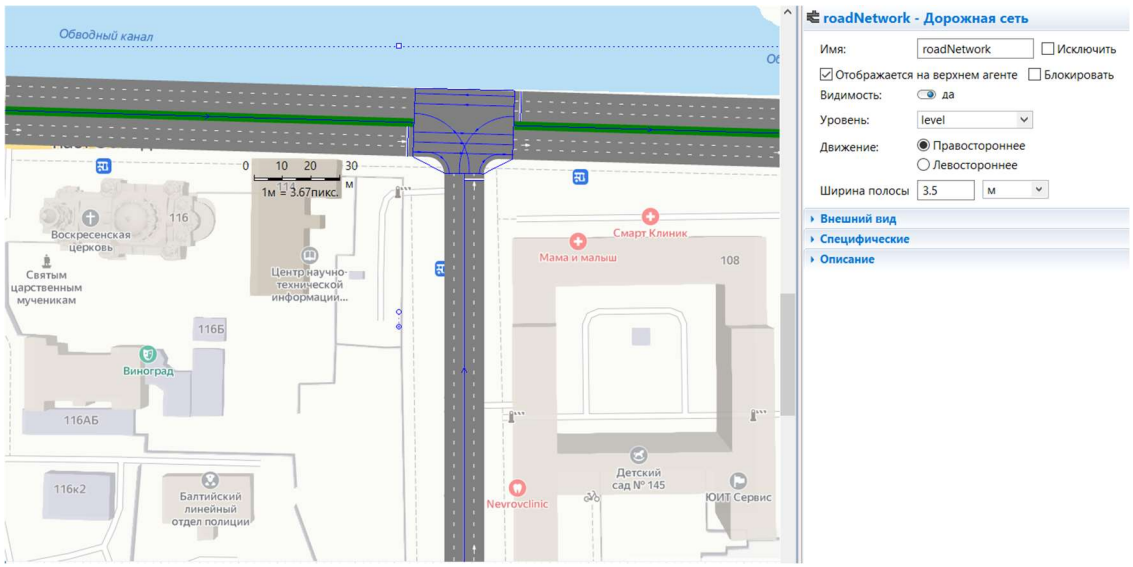

Рис. 2. Итоговая ИМ перекрестка, где

«W»– обозначен участок дороги до перекрестка, соответствует направлению «запад»: roadW; «E» – рассматривается участок дороги после перекрестка, соответствует направлению «восток»: roadE; «S» – позиционируется участок дороги, соответствует направлению «юг»: roadS;

Задание интенсивности движения.

Автомобили добавляются в систему с помощью шести блоков CARSOURCE (рис. 4).

Три элемента SCHEDULE используются для определения скорости прибытия (рис. 3):

a) rateSchedSE_SW - для направления юг/восток (S/E) и юг/запад (S/W);
b) rateSchedES_WS - для направления восток/юг (E/S) и запад/юг (W/S);
c) rateSchedEW_WE - для направления восток/запад (E/W) и запад/восток (W/E).

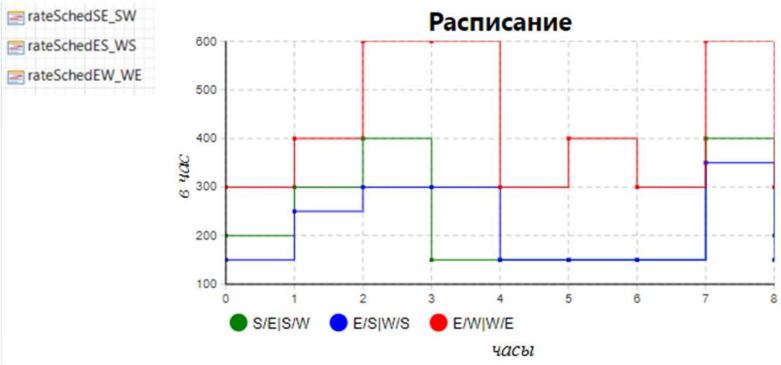

Рис. 3. Расписание интенсивностей (скорости прибытий)

Значения, используемые в расписании, представляют собой усредненные показатели интенсивностей для любого подобного т-образного перекреста в рассматриваемый интервал реального времени с 8:00 и до 16:00.

Транспортные средства (ТС), которые добавляются в блок-схему, представляют собой настраиваемый тип агента под названием Car.

В этом пользовательском типе агента есть две переменные с именами isNS и birthminutes. Параметр isEW — логическое значение, которое принимает значение true, если автомобиль движется в направлении восток/запад или запад/восток, иначе false. Значение этой переменной устанавливается после сборки автомобиля и при выходе из блока CARSOURCE, в котором он

был создан. Код (car.isEW = true или car.isEW = false):

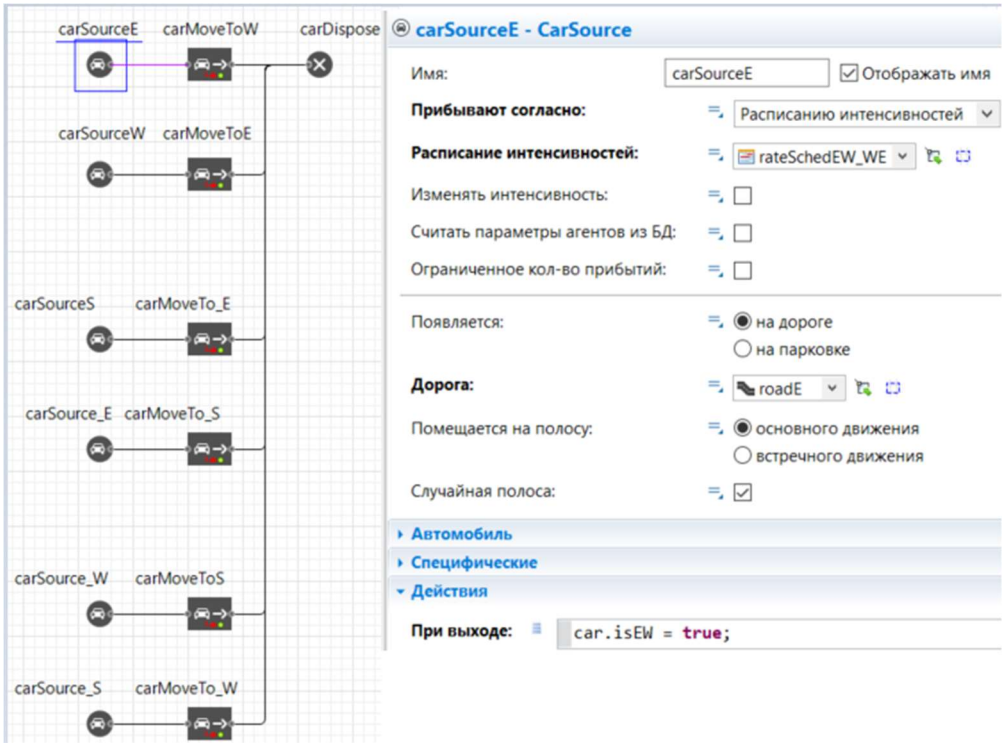

Рис. 4. Блок-схема логики дорожного движения

Другая переменная в типе агента Car, birthminutes, записывает время (в минутах), в течение которого создается агент автомобиля. Код (time(TimeUnits.MINUTE)) устанавливается в качестве начального значения переменной.

В блок-схеме также есть четыре блока CARMOVETO, которые предписывают прибывающим автомобилям установить пункт назначения как конец встречной дороги:

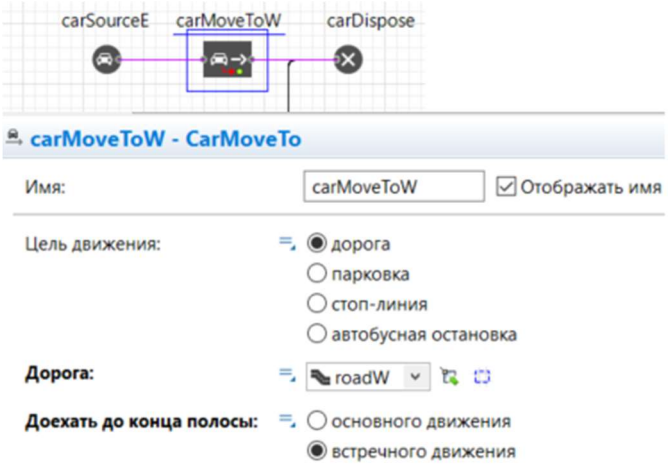

Рис. 5. Демонстрация работы блока carMoveTo

Параметры для задания светофорных циклов.

Настройка регулирования движения осуществляется путем внедрения светофорного регулирования. В данном случае используется светосигнальная установка с классическим распределением цветов: красный, зеленый, мигающий желтый. Общий цикл работы светофора $T_{ц}$ = 125 с., подсчет был составлен по таблице 1 (с учетом желтых фаз $t_{ж} = 3$). Значения

длительностей фаз были заданы, исходя из визуального наблюдения за реальной дорожной ситуацией, на рассматриваемом т-образном перекрестке (см. Рис. 1).

Таблица 1 – Время работы фаз

№ Фазы	т. Фазы
1	30
2	51
3	44
	125

Всего, для рассматриваемой светосигнальной установки, используется 4 групп сигналов. Во избежание путаницы требуется составить схему пофазного разъезда по каждому маршрутному направлению:

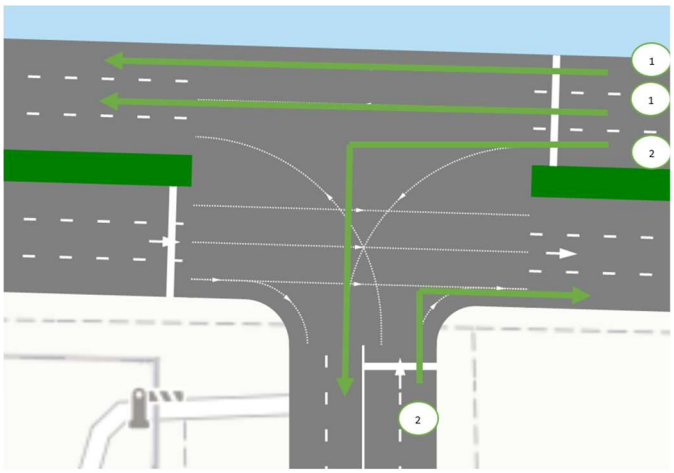

Рис. 6. Фаза 1

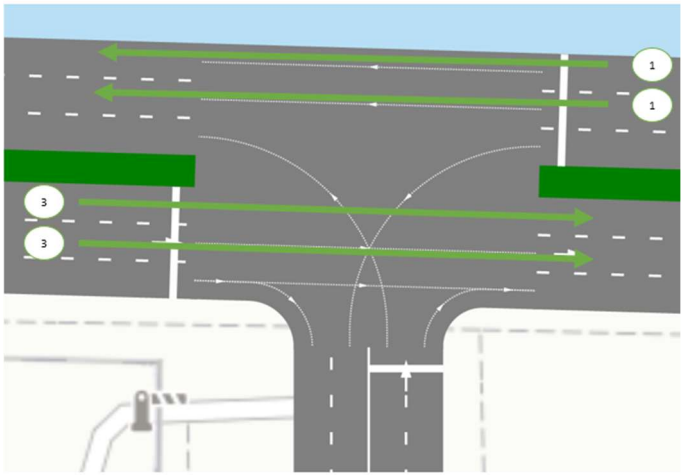

Рис. 7. Фаза 2

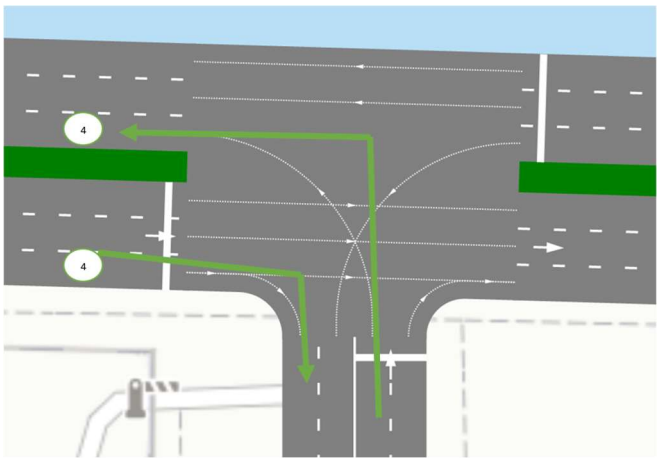

Рис. 8. Фаза 3

Далее, в соответствии с рисунками 6-8 формируется светосигнальная установка в Anylogic за это отвечает блок TRAFFICLIGHT :

Рис. 9. Общий вид светосигнальной установки

Этот блок контролирует восемь соединительных полос вокруг перекрестка и имеет шесть фазы (пронумерованные от 0 до 5, как показано на рис. 10). Первая фаза (#0) дает право проезда автомобилям в направлении восток/запад, восток/юг и юг/восток. За этим следует (в фазе №1) короткая желтая фаза. Затем на третьем этапе (#2) право проезда предоставляется автомобилям в направлении запад/восток и восток/запад. Фаза (#3) — еще одна короткая желтая фаза. После нее идет (#4) фаза, которая разрешает проезд в направлении запад/юг и юг/запад. После чего идет последняя желтая фаза (#5), как только эта она заканчивается, светофор возвращается к началу [7].

Чтобы измерить, как долго каждое направление имеет зеленый свет, добавляются четыре набора данных: phaseEW, phaseES-SE, phaseWE, phaseWS-SW

Переменная с именем lastPhaseChangeTS записывает метку времени, когда свет в последний раз менял фазы. Всякий раз, когда происходит изменение фазы, индекс новой фазы используется для определения того, следует ли добавить продолжительность последней фазы в один из наборов данных [7].

Продолжительность вычисляется путем взятия текущего модельного времени и вычитания из него значения переменной lastPhaseChangeTS. Если новый индекс равен 1, то последняя фаза была с индексом 0 (т. е. автомобили E/W, E/S и S/E получили право проезда); следовательно, продолжительность должна быть добавлена к PhaseLensEW и phaseES_SE. Та же логика применяется, если новый индекс равен 3 (но со значением, добавляемым к PhaseLensEW и PhaseLensEW), и т.д. Независимо от нового индекса переменная lastPhaseChangeTS всегда обновляется и устанавливается на текущее модельное время [8,9]:

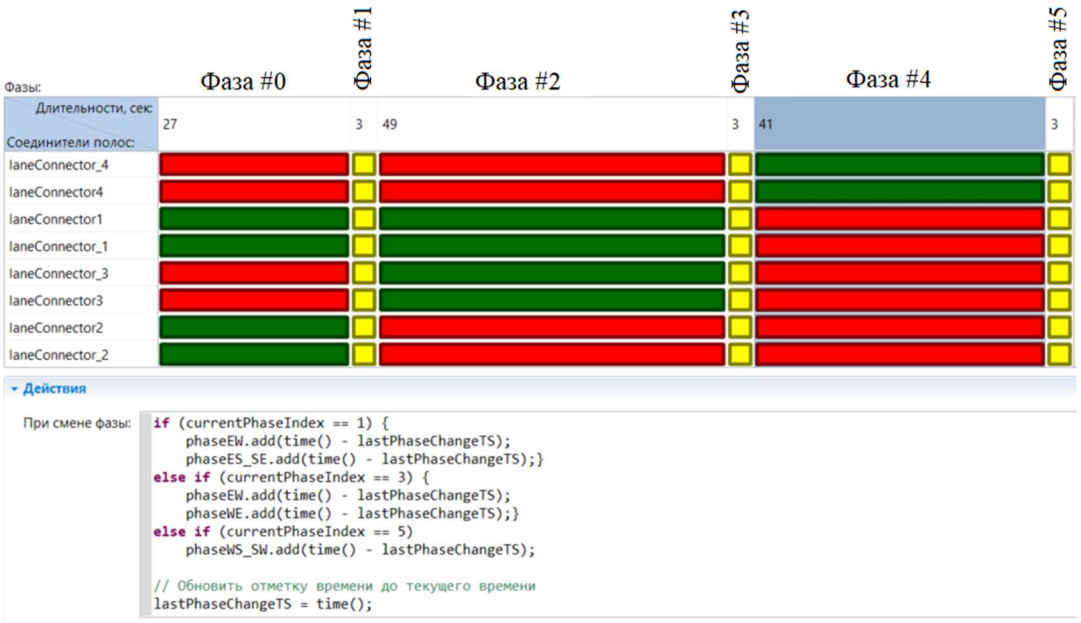

Рис. 10. Блок светофора и код для обновления наборов данных длины фазы зеленого в поле «При изменении фазы»

Анализ базовой модели

Ключевой показатель эффективности перекрестка определяется как среднее время пребывания автомобиля в системе, который отслеживается с помощью набора данных с именем tisDS. Когда автомобиль выезжает из системы (через блок CARDISPOSE), к набору данных добавляется время пребывания автомобиля в модели. Это значение рассчитывается как разница между текущим модельным временем и значением переменной «birthminutes» автомобиля [8]:

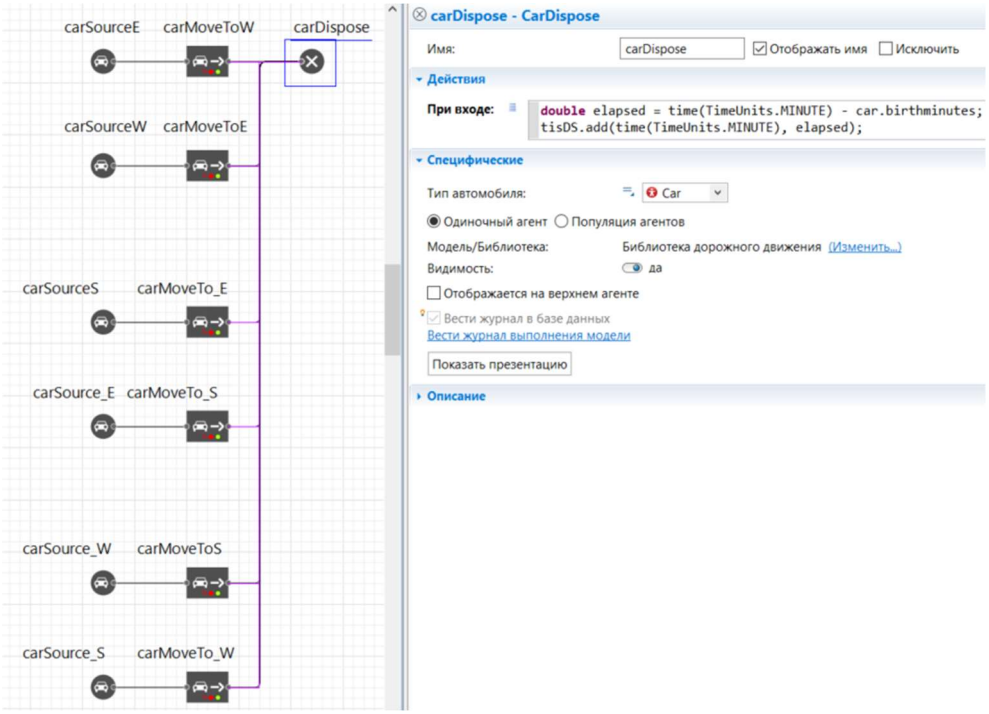

Рис. 11. Блок CarDispose и обновление времени в системном наборе данных, когда автомобиль покидает модель

Следует запустим модель для определения среднего времени автомобилей в системе (до применения обучения с подкреплением и оптимизации):

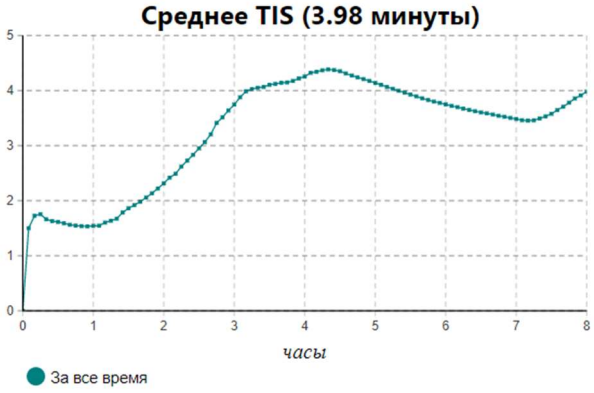

Рис. 12. График среднего времени автомобилей в системе

Результаты простого эксперимента, с базовыми настройками фаз светофорного объекта, которые, как говорилось выше, были заданы, исходя из визуального наблюдения за реальной дорожной ситуацией (см. Рис. 1), и наглядно показали необходимость в уменьшении среднего времени нахождения всех автомобилей в системе.

Алгоритм построения оптимизационного эксперимента

Построения оптимизационного эксперимента, в программе Anylogic, выполняется на основе приведенной математической модели целевой функции [6]:

$$\sum_{1}^{8}\frac{\sum_{1}^{N_{npWE\_EW}} t_{max}-t_{k_1}-t_{k_2}}{N_{np_{WE\_EW}}}+\frac{\sum_{1}^{N_{npEW\_WE}} t_{max}-t_{k_2}}{N_{np_{EW\_WE}}}+\frac{\sum_{1}^{N_{npES\_WS}} t_{max}-t_{k_1}}{N_{np_{ES\_WS}}}+\frac{\sum_{1}^{N_{npES\_WS}} t_{max}-t_{k_3}}{N_{np_{ES\_WS}}}+\frac{\sum_{1}^{N_{npSE\_SW}} t_{max}-t_{k_1}}{N_{np_{SE\_SW}}}+\frac{\sum_{1}^{N_{npSE\_SW}} t_{max}-t_{k_3}}{N_{np_{SE\_SW}}}\to \min, \quad (1)$$

Где t_{max} – максимальное время, за которое ТС проходит заданный путь, с;

$t_{k_{1,2,3}}$ – продолжительность красной фазы светофора, с;

$$N_{np_{1,2,3}}=\sum_{m_1,m_2m_3}^{1} N_{m_1,m_2,m_3}\cdot k_{m_1,m_2,m_3}$$

– приведенная к легковому автомобилю расчётная часовая интенсивность движения в одном направлении, данные значения задаются расписанием (Рис. 3), авт/ч, где:

N_{m_1,m_2,m_3} – количество автомобилей, k_m – коэффициент приведения для легковых автомобилей $k_{m_1,m_2,m_3}=1$. Таким образом,

$$N_{np_{1,2,3}}=\sum_{m_1,m_2,m_3}^{1} N_{m_1,m_2,m_3}$$

Для удобства была произведена замена $N_{np_1}=N_{np_{EW\_WE}}$, $N_{np_2}=N_{np_{ES\_WS}}$, $N_{np_3}=N_{np_{SE\_SW}}$, [5].

Граничные условия:

$$t_{k_1} \geq 20; t_{k_1} \leq 35;$$
$$t_{k_2} \geq 35; t_{k_2} \leq 60;$$
$$t_{k_3} \geq 30; t_{k_3} \leq 50.$$

Варьируемые параметры: t_{k_1,k_2,k_3};

Ограничения:

$$N_{np_{1,2,3}} = \frac{n_{1,2,3} \cdot z_{1,2,3} \cdot 1000 \cdot V_{потоk_{1,2,3}} \cdot Z}{S_{1,2,3} \cdot (Z + \frac{V^2_{пер}}{26 \cdot a} + \frac{V^2_{пер}}{26 \cdot b} + \frac{(\frac{t_{k_{1,2,3}} - t_{ж}}{2}) \cdot V_{пер}}{3,6})} - \quad (2)$$

где $z_{1,2,3} = \frac{N_{ф_{1,2,3}} \cdot \varepsilon}{N_{n_{1,2,3}} \cdot n_{1,2,3}}$

- уровень загрузки движения, где $N_{ф_{1,2,3}}$ – фактическая интенсивность движения, авт/ч, приведенная к легковому автомобилю,

ε – коэффициент сезонной неравномерности движения, $\varepsilon = 1,26$,

$N_{n_{1,2,3}}$ – типичная пропускная способность полосы движения, авт/ч, величина варьируется в зависимости от времени наблюдения. В данной работе $N_{ф_{1,2,3}} = N_{np_{1,2,3}}$, значения задаются расписанием (Рис. 3), соответственно,

$$N_{n_{1,2,3}} = \frac{N_{пр}}{n_{1,2,3}}.$$

Как правило, $z_{1,2,3} = 0,7$; $n_{1,2,3}$ – количество полос, $n_{EW\_WE} = n_1 = 4$, $n_{ES\_WS} = n_2 = 2$, $n_{SE\_SW} = n_3 = 2$; $V_{потоk_{1,2,3}}$ – скорость потока, км/ч; Z – расстояние между перекрестками, для удобства в расчетах будет использоваться среднее значение между перекрестками $Z = 402м$, величина была определена при помощи замера. Соответственно,

$$Z = \frac{241 + 353 + 618}{3} = 402$$

$$S_{1,2,3} = \frac{V_{потоk_{1,2,3}}}{3.6} + \frac{V_{потоk_{1,2,3}}{}^2 K}{254 \cdot (\phi \pm i)} + l_0 + l_a$$

- динамический габарит автомобиля ТС, где l_o – дистанция безопасности между остановившимися ТС, $l_o = 2м$, l_a – длина расчетного ТС, $l_a = 6м$, ϕ – коэффициент сцепления, $\phi = 0,5$, K – коэффициент эксплуатационного состояния тормозов, $K = 1,2$,

i – продольный уклон, $i = 0$.

Усреднённое значение динамического габарита автомобиля принимается равным $S_{1,2,3} = 64м$;

$V_{пер}$ – расчетная скорость перед перекрестком, $V = 30км/ч$,

a – ускорение при разгоне ТС, $a = 1,2м/с^2$;

b – ускорение при разгоне ТС, $b = 1,5м/с^2$;

$t_{ж}$ – время продолжительность желтого сигнала светофора, $t_{ж} = 3c$.

Ограничение было выведено из формулы приведенной интенсивности: $N_{np_{1,2,3}} = z \cdot P_{1,2,3} \cdot n_{1,2,3}$ (9), где $P_{1,2,3} = \frac{1000 \cdot V_{поток_{1,2,3}}}{S_{1,2,3}}$ - пропускная способность [6].

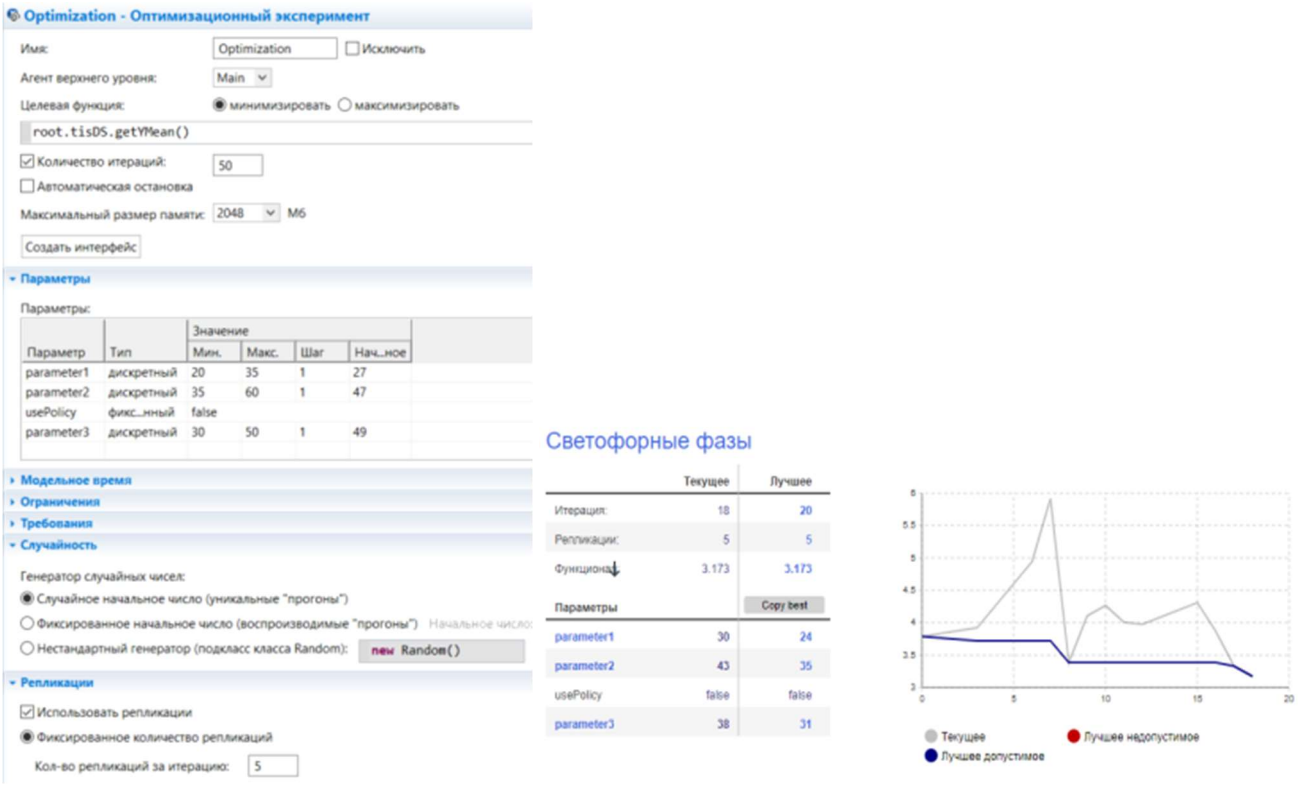

Рис. 13. Настройки и итоги эксперимента

Данные значения будут использоваться при проведении основанного эксперимента, который в AniLogic называется Simulation, над построенной ранее ИМ.

Алгоритм построения нестандартного эксперимента для обучения с подкреплением

В документации Anylogic отмечено, что библиотека RL4J — это фреймворк для обучения с подкреплением, интегрированный с deeplearning4j и выпущенный под лицензией с открытым исходным кодом Apache 2.0.

В данном исследовании используется алгоритм DQN (Deep Q Learning), т. е. Q-обучение в сочетании с глубокими нейронными сетями, реализованное в R L4J.

Упрощенный алгоритм процесса обучения можно описать следующим образом [7]:

a) На каждом временном шаге обучающийся агент (ИНС) наблюдает за текущим состоянием среды, т.е ИМ;
b) на основе этой информации (которая может не включать полностью всю информацию, содержащуюся в состоянии), он выбирает действие, которое нужно предпринять.
c) После этого происходит временной шаг, и последствия этого действия применяются к системе.
d) Затем агент получает числовое вознаграждение и по прошествии времени снова начинает процесс в новом состоянии.

Соответственно, математическая постановка задачи будет выглядеть следующим образом:

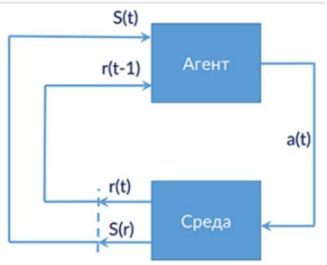

Рис. 14. Схема взаимодействия агента со средой.

Среда описывается марковским процессом принятия решений (МППР) , который включает $S(states)$ – множество состояний среды; $A(actions)$ – множество возможных действий; условное распределение $P(s'|s,a)$ следующего состояния при условии текущего и действия; $R(rewards)$ – вознаграждения при переходе в новое состояние R(s,s'); коэффициента дисконтирования $0<\gamma<1$,который определяет предпочтение текущего вознаграждения по сравнению с будущим.

В произвольный момент времени t агент характеризуется состоянием $s_t \in S$ и множеством возможных действий $A(s_t)$. Выбирая действие $a \in A(s_t)$, он переходит в состояние s_{t+1} и получает выигрыш r_t. Основываясь на таком взаимодействии с окружающей средой, агент, обучающийся с подкреплением, должен выработать оптимальную политику, т. е. стратегию $\pi: S \to A$, которая максимизирует функцию ценности действия Q – математическое ожидание совокупного вознаграждения:

$$Q=E\sum_t \gamma^t r_t \tag{3}$$

Рекуррентная формула для функции ценности Q имеет вид

$$Q(s,a) = E(r_t + \gamma Q(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a) \tag{4}$$

В алгоритме DQN функция Q аппроксимируется при фиксированных параметрах сети w_t:

$y_t = r_t$, если состояние s_{t+1} терминальное, иначе $y_t = r_t + \gamma max_a Q(s_{t+1}, a; w_t)$,

где второе слагаемое определяет оптимальную функцию Q^* на основе уравнения Беллмана.

Функция потерь $D(w)$ для обучения $Q(s,a,w)$ является стандартной квадратичной ошибкой:

$$D(w) = (Q(s_t, a_t; w_t) - y_t)^2. \tag{5}$$

Награда в модели определяется как

$$r_t = (d_0 - d_1) / max(d_0, d_1), \tag{6}$$

где в числителе разница задержек до и после действия.

Поскольку r_t всегда лежит в диапазоне между -1 и 1, то значение функции Q определяется в процессе поиска точки в интервале между двумя экстремумами [-1,1].

Чтобы настроить этот процесс, необходимо выполнить три шага:

1 Библиотека RL4J импортируется как зависимость AnyLogic.

2 Создается ИМ с добавлением необходимых функций, обеспечивающих связь между моделью и структурой обучения с подкреплением. В частности, в модель AnyLogic добавлены две функции: одна для получения наблюдения (getObservation), а другая для выполнения

действия (doAction).

3 ИНС настраивается с использованием фреймворка RL4J и подключается к ИМ. Вся эта настройка реализована внутри пользовательского эксперимента.

Для правильного построения нестандартного эксперимента более детально описать шаги:

a) Импорт библиотеки RL4J. Библиотека RL4J была импортирована путем добавления файла Jar в свойства модели:

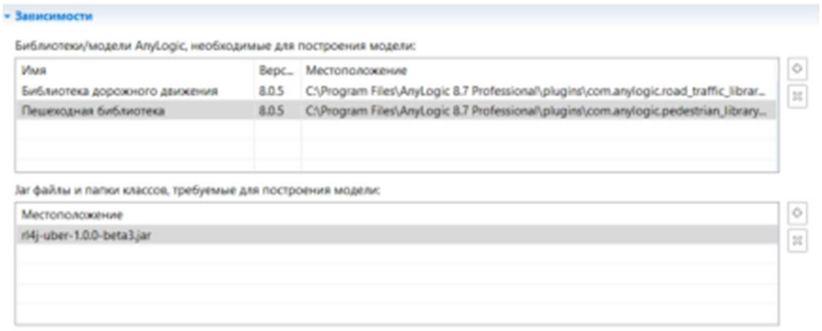

Рис. 15. Импорт библиотеки RL4J в AnyLogic

b) Добавление необходимых функций в модель AnyLogic, для обеспечения взаимодействия со структурой обучения с подкреплением. Чтобы ИНС могла наблюдать за текущим состоянием ИМ и предпринимать какие-либо действия, в имитационной модели заданы две функции: doAction и getObservation.

doAction – требует числового ввода и на основе этого значения выполняет некоторое действие в модели. В ИМ она либо ничего не сделает (если аргумент действия равен 0), либо переключит фазу светофора на следующую (если аргумент действия равен 1); единственное предостережение в последнем случае заключается в том, что она переключится на следующую фазу, только если свет не находится в одной из двух желтых фаз. Этот параметр не позволяет обучающему агенту обрезать желтые фазы, поскольку полные три секунды необходимы для обеспечения плавного перехода между фазами:

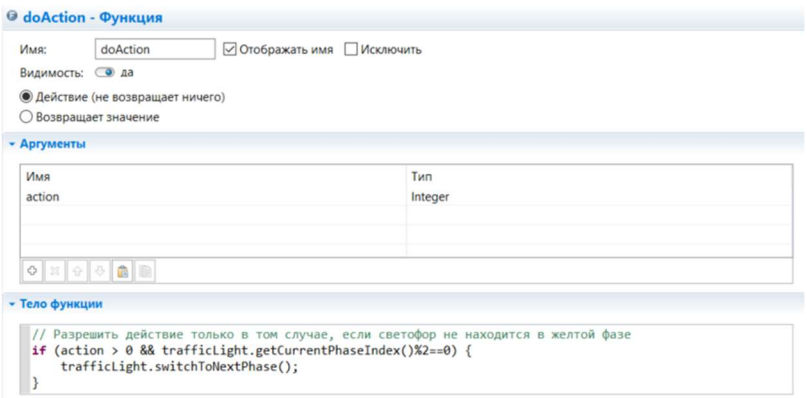

Рис. 16. Определение функции doAction

getObservation — это функция (наблюдение), которая суммирует всю информацию, содержащуюся на заданном перекрестке (Рис. 1), в массиве из 8 элементов, который содержит информацию о состоянии системы для ИНС, чтобы найти оптимальную политику. Каждый из первых восьми элементов (индексы от 0 до 5) представляют собой общее количество времени, проведенное в модели для всех автомобилей на заданной полосе и в непосредственной

близости от перекрестка. Седьмой элемент (индекс 6) работает аналогично первым восьми элементам, но фокусируется только на автомобилях на перекрестке. Восьмой элемент (индекс 7) — это текущий фазовый индекс светофора (число от 0 до 5):

```
double[] state = new double[8];
double time = time(TimeUnits.MINUTE);

// Получить длину всех дорог (в метрах); используется для определения близости к перекрестку.
double lenS = scale.toLengthUnits(roadS.getSegment(0).length(), LengthUnits.METER);
double lenE = scale.toLengthUnits(roadE.getSegment(0).length(), LengthUnits.METER);
double lenW = scale.toLengthUnits(roadW.getSegment(0).length(), LengthUnits.METER);

// Суммарный TIS всех автомобилей в радиусе 150 м от перекрестка
// * getCars(true)  -- все автомобили в передней полосе (полосы движутся в сторону перекрестка)
// * getCars(false) -- все автомобили в полосе заднего хода (полосы уходят от перекрестка)
// * Смещение начинается в конце дороги, поэтому фильтр проверяет,
// * находятся ли автомобили в последних 150 м от дороги.
state[0] = roadE.getCars(true).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenE-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();
state[1] = roadE.getCars(false).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenE-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();
state[2] = roadS.getCars(true).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenS-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();
state[3] = roadS.getCars(false).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenS-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();
state[4] = roadW.getCars(true).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenW-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();
state[5] = roadW.getCars(false).stream()
                .filter(c -> ((Car)c).getRoadOffset(LengthUnits.METER) >= lenW-150)
                .mapToDouble(c -> time - ((Car)c).birthminutes).sum();

// Общий TIS автомобилей на перекрестке
state[6] = intersection.getCars().stream().mapToDouble(c -> time - ((Car)c).birthminutes).sum();

// Индекс фазы светофорае
state[7] = trafficLight.getCurrentPhaseIndex();
```

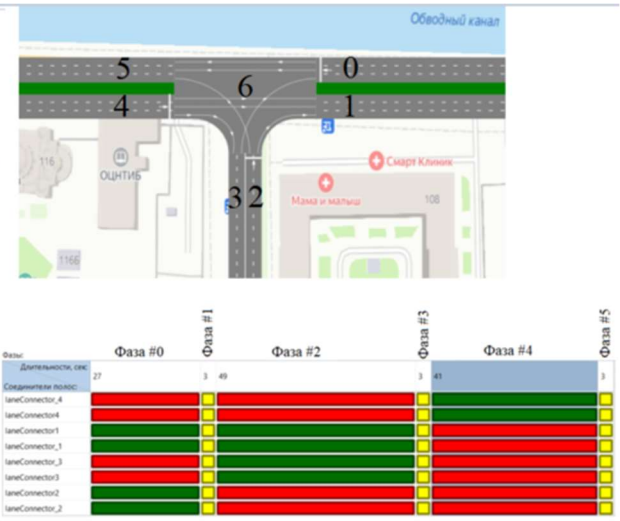

Рис. 17. Функция getObservation

c) Пользовательский эксперимент служит для контроля выполнения и Q-обучения на прогонах модели.

Отдельно следует продемонстрировать процесс получения окончательной политики, так как важно убедиться в том, что предыдущая политика (если она существует) не перезаписалась (Рис.18).

```
    // Получить окончательную политику
    DQNPolicy<Encodable> pol = dql.getPolicy();
    // Сериализировать и сохранить (демонстрация сериализации, но не обязательно).
    // Убедитесь, что предыдущая политика не перезаписывается.
    String fn = "PhasePolicy";
    String ext = ".zip";
    int n = 0;
    File outfile = new File(fn + ext);
    while (outfile.exists())
        outfile = new File(fn + "_" + (n++) + ext);
    if (n == 0)
        pol.save(fn + ext);
    else
        pol.save(fn + "_" + n + ext);


    //закрыть mdp (закрыть двигатель)
    mdp.close();

} catch (IOException e) {
    e.printStackTrace();
}
```

Рис. 18. Код получения окончательной политики

На рис.19 показан этап прохождения обучения политики:

```
public StepReply<Encodable> step(Integer action) {
    double[] s0 = root.getObservation();
    // {изменена пошаговая функция, чтобы игнорировать действие, если в желтой фазе}
    root.doAction(action);
    // Приращение времени на 10 секунд
    engine.runFast(root.time() + 1.0/6.0);
    double[] s1 = root.getObservation();
    // Изменение в прямом направлении + задержка пересечения
    double delay0 = s0[0] + s0[2] + s0[4] + s0[6];
    double delay1 = s1[0] + s1[2] + s1[4] + s1[6];
    double reward = delay0 - delay1;
    if (delay0 > 0 || delay1 > 0) {
        reward /= Math.max(delay0, delay1);
    }
    return new StepReply(getObservation(), reward, isDone(), null);
}
```

Рис. 19. Код, разъясняющий этапы обучения политики

Код на рис. 19 указывает, что должно произойти на каждом этапе обучения, состоящем из наблюдения, действия, временного шага и вознаграждения. Точнее, перед вызовом этой функции политика просматривает свое наблюдение и решает, какое действие предпринять; это передается в качестве аргумента функции шага. Код, показанный в этой функции, вызывает функцию getObservation модели, которая будет использоваться для расчета вознаграждения. Затем предпринимается действие, и симуляция запускается для заданного (фиксированного) временного шага. После чего проводится еще одно наблюдение за моделью после того, как она подверглась воздействию предпринятого действия.

На основе наблюдений, снятых с модели до и после действия, рассчитывается вознаграждение. Функция вознаграждения представляет собой числовое значение от -1 до 1. Для обоих наблюдений сумма берется для автомобилей в передних полосах (движущихся к центру) и автомобилей на перекрестке. Изначально вознаграждение задается как разница между этими двумя значениями. Затем он нормализуется путем деления максимума значений. Наконец, возвращается объект StepReply, содержащий новое наблюдение, вознаграждение, логический набор, основанный на том, находится ли модель в конечном состоянии, и любую дополнительную информацию.

После произведения всех вышеописанных манипуляций производится запуск пользовательского эксперимента и начнется обучение.

Чтобы протестировать уже обученную политику, нужно проверить, ее существование в папке модели перед началом моделирования (Рис. 21).

Циклическое событие также добавляется в ИМ для использования политики. Его время повторения устанавливается равным той же величине, которая использовалась в качестве временных шагов, используемых во время обучения. Выполняемый код должен сначала проверить, не является ли переменная «policy» нулевой (т. е. политика успешно загружена). Если это оценивается как истинное, логика отправляет текущее наблюдение обученной политике; затем он получает действие, которое политика определяет как оптимальное, и затем это действие выполняется(рис.20).

Additional class code:

```
// Load the previous agent
DQNPolicy<Encodable> policy = null;
String filename = "PhasePolicy.zip";
{
    try {
        policy = DQNPolicy.load(filename);
        traceln("Policy import succeeded!");
    } catch (IOException e) {
        traceln("Policy import failed...");
    }
}
```

Рис. 20 – Импорт обученной политики при условии ее существования

В настоящей статье событие будем использовать политику для управления светофором. Каждые 10 секунд из модели берется наблюдение, которое политика использует для определения правильного действия (рис. 21).

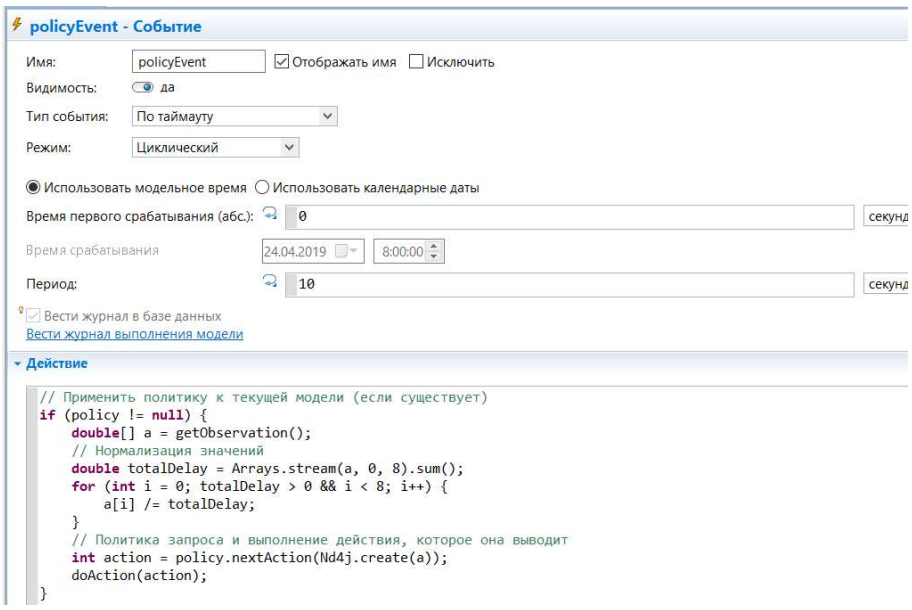

Рис. 21. Циклическое событие для развертывания изученной политики

Если в папке модели нет политики или файл, указанный в разделе «Дополнительный код класса» свойств *Main*, неверен, вычислительный эксперимент запустит модель, используя значения по умолчанию для временных интервалов фаз светофора.

После проведения всех вышеперечисленных действий обученная политика будет готова к применению для осуществления простого эксперимента с применением фактических фиксированных длительностей фаз.

Анализ результатов и выводы

В данной разделе осуществлены два простых эксперимента, где изменяются исходные значения длительностей светофорных фаз:

a) Первое исследование берет за основу значения, полученные в результате оптимизационного эксперимента;

b) Второе исследование оперирует итогами, взятыми из обученной политики.

После проведения двух, вышеописанных, экспериментов проводиться сравнение их результатов, экспериментов на основе обучения с подкреплением и методом оптимизации;

Затем, подводятся общие итоги, где обосновывается выбор в пользу одного из двух имеющихся методов.

После замены значений по умолчанию для длин фаз в блоке светофора на оптимизированные, модель была запущена. В представлении статистики модели графики дают представление о производительности системы (рис. 22)

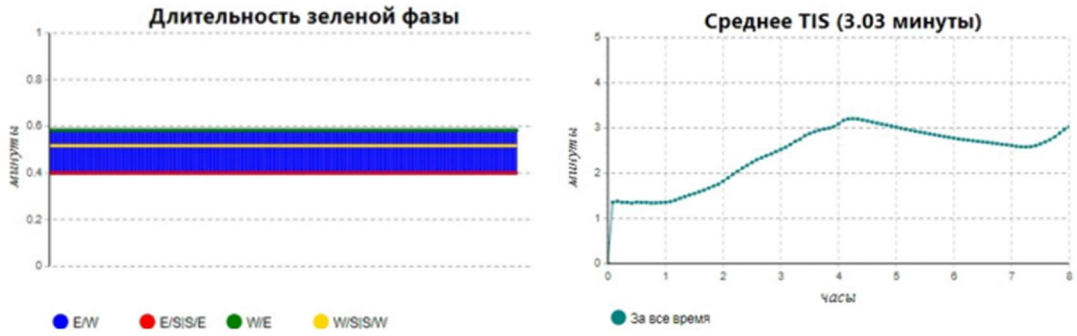

Рис. 22. Графики с оптимизированными значениями

Проведение исследования со значениями, взятыми из обученной политики(рис.23).

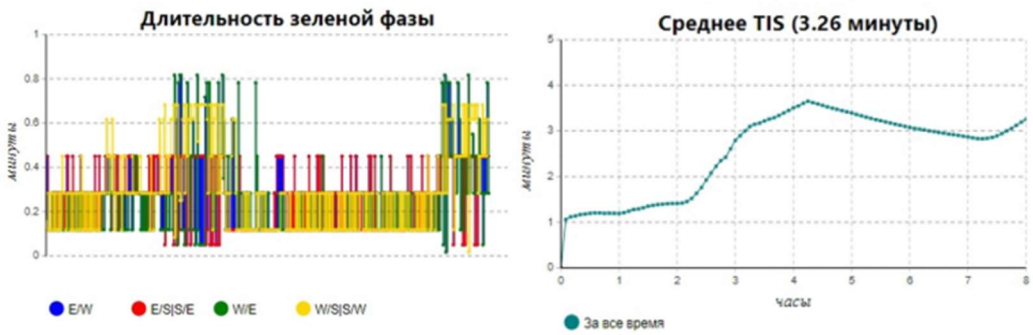

Рис. 23. Графики с использованием обучающей политики

Были выполнены эксперименты при разных расписаниях интенсивности. На рис. 3 расписание соответствует движению в будние дни. На рис. 24 показано расписание интенсивностей в выходные.

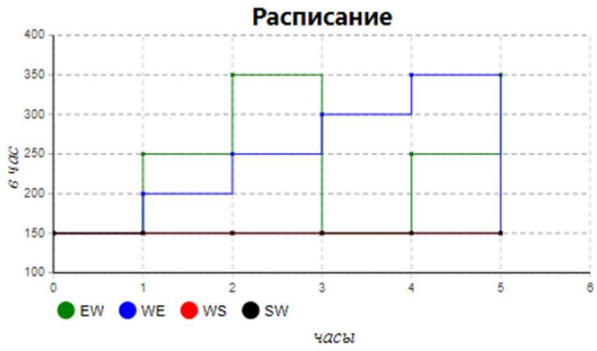

Рис. 24. Расписание интенсивностей в выходные

На рис.25 и 26 показаны соответствующие результаты для измененного расписания.

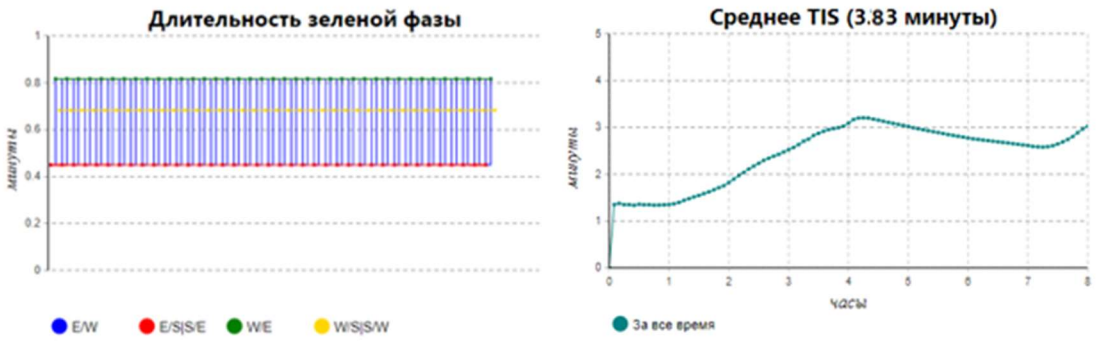

Рис. 25. Графики с оптимизационными значениями

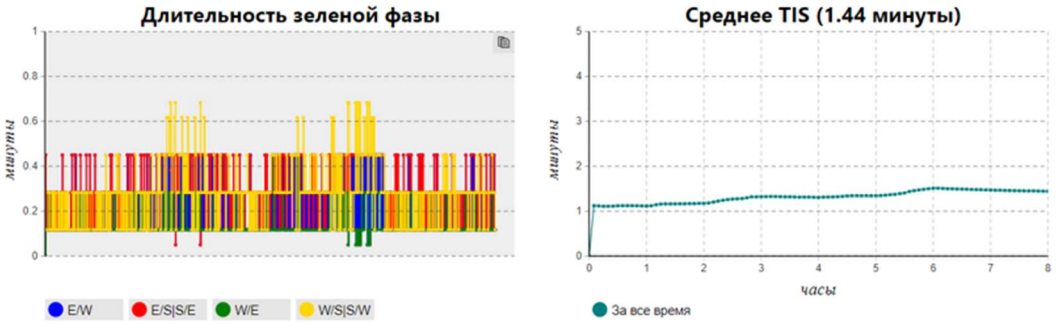

Рис. 26. Графики с использованием обучающей политики

Оценка адаптивности алгоритмов к изменению уровней интенсивности движения ТС

Для 2-х расписаний интенсивностей, приведенных выше выполнялась настройка политики управления и оптимизация для сравниваемых алгоритмов *opt* и *dqn* соответственно, а также приведены результаты времен нахождения ТС в системе *base* (реальный регулятор).

При фиксированных настройках в расписание вносились возмущения по уровням интенсивностей движения от 0 до 40 процентов от «оптимальных». В результате модельного эксперимента определялись времена нахождения ТС в системе (табл. 2). Строка 0% соответствует оптимальной настройке. В таблице также приведены показатели относительной эффективности алгоритма обучения с подкреплением относительно оптимизационного δ_1 и относительно базовых времен δ_2 по формулам:

$$\delta_1 = (opt - dqn)*100/opt; \quad \delta_2 = (base - dqn)*100/base.$$

Таблица 2 – Оценка адаптивности алгоритмов

Будни

Расстройка,%	*opt*,мин	*dqn*,мин	*base*,мин	δ1,%	δ2,%
0	3,03	3,68	3,98	-21,5	7,5
10	2,5	2,71	3,56	-8,4	23,9
20	2,1	1,58	2,43	24,8	35,0
30	1,76	1,28	2,21	27,3	42,1
40	1,38	1,16	1,86	15,9	37,6

Выходные

Расстройка,%	*opt*,мин	*dqn*,мин	*base*,мин	δ1,%	δ2,%
0	3,67	1,44	4,78	60,8	69,9
10	2,95	1,27	3,83	56,9	66,8
20	1,98	1,21	2,94	38,9	58,8
30	1,45	1,17	2,44	19,3	52,0
40	1,44	1,14	1,54	20,8	26,0

На рис. 27 и 28 приведены соответствующие диаграммы. Хорошо видно, что с увеличением «расстройки» относительно оптимальных параметров кривые времен сравниваемых алгоритмов сближаются.

Способность алгоритмов к «обобщению» в данной задаче соответствует адаптации к измененным входным значениям интенсивностей. В алгоритме *dqn* она оказалась выше, что видно из таблицы и диаграмм. Однако этот алгоритм в большей степени чувствителен к типу расписания по сравнению с оптимизационным, который превзошел *dqn* примерно на 20% в оптимальной точке для расписания в будни. Преимущество *dqn* алгоритма сильнее проявилось

на 2-м расписании, особенностью которого является тенденция к нарастанию интенсивности движения, а именно: 61% по сравнению с оптимизацией и 70% по отношению к базовой настройке. Также отметим, что к «расстройке» входных данных относительно оптимальной политики он оказался практически нечувствителен при изменении уровней интенсивностей в пределах этого расписания.

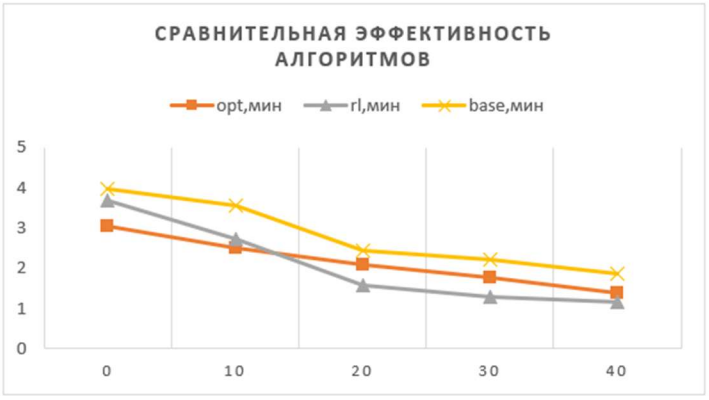

Рис. 27. Иллюстрация эффективности сравниваемых алгоритмов для расписания в будни

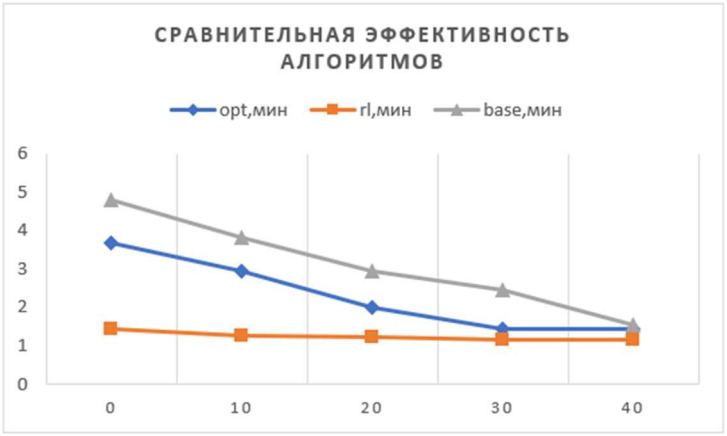

Рис. 28. Иллюстрация эффективности сравниваемых алгоритмов для расписания в выходные

Было показано, что результаты регулирования транспортным процессом на исследуемом реальном перекрестке, полученные при моделировании с использованием обучения с подкреплением превосходят оптимизационный подход, однако являются чувствительны к заданному расписанию.

СПИСОК ЛИТЕРАТУРЫ

1. Кретов А.Ю. Обзор некоторых адаптивных алгоритмов светофорного регулирования перекрестков. Известия Тульского государственного университета 2013, Технические науки Выпуск 7 Часть 2, Тула: Издательство ТулГУ. — 390 с;
2. Андронов, С. А. Разработка и исследование имитационной модели светофорного регулирования на основе нечеткой логики в среде AnyLogic // Седьмая всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика» ИММОД-2015: Труды конф., 21–23 окт. 2015 г., Москва: в 2 т.

— Т. 2. — М.: ИПУ РАН, 201, С. 443–449;
3. Андронов, С. А. Сравнение эффективности адаптивных алгоритмов светофорного регулирования в среде AnyLogic / С.А.Андронов // Международный научно-практический журнал. Программные продукты и системы, №1 за 2019 год, с.150-158;
4. Седых И.А., Демахин Д.С. Гибкое управление светофорной системой перекрестка на основе нейронных сетей / И.А. Седых, Д.С. Демахин // Автоматизация процессов управления N 1(47),2017,с.94-100.
5. I. Arel, C. Liu, T. Urbanik and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," in IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128-135, June 2010;
6. ОДМ 218.1.001-2005 – отраслевой дорожный методический документ;
7. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction;
8. Lapan, M. (2018). Deep reinforcement learning hands-on: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more;
9. Patterson, J., & Gibson, A. (2017). Deep learning: A practitioner's approach;
10. Справка Anylogic [Электронный ресурс]: AnyLogic Road Traffic Library. URL: https://anylogic.help/anylogic/index.html (дата обращения: 30.03.2022).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Андронов Сергей Александрович –
кандидат технических наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Россия, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: andronov_00@mail.ru

Прокофьева Марина Сергеевна –
магистрант
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Россия, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: m4riprokofjeva@yandex.ru

INFORMATION ABOUT THE AUTHORS

Andronov Sergey Alexandrovich –
Candidate of Technical Sciences, Associate Professor
Saint-Petersburg State University of Aerospace Instrumentation
SUAI, 67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: andronov_00@mail.ru

Prokofeva Marina Sergeevna –
master
Saint-Petersburg State University of Aerospace Instrumentation
SUAI, 67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: m4riprokofjeva@yandex.ru