КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2024 Т. 16 № 1 С. 69–78
DOI: 10.20537/2076-7633-2024-16-1-69-78

Ки&М

СПЕЦИАЛЬНЫЙ ВЫПУСК

УДК: 519.8

Фреймворк sumo-atclib для моделирования адаптивного управления трафиком дорожной сети

В. И. Казорин[a], Я. А. Холодов[b]

Университет Иннополис,
Россия, 420500, г. Иннополис, ул. Университетская, д. 1

E-mail: [a] v.kazorin@innopolis.university, [b] ya.kholodov@innopolis.ru

Получено 25.12.2023
Принято к публикации 25.12.2023.

В данной статье предлагается фреймворк *sumo-atclib*, который предоставляет удобный единообразный интерфейс для апробации разных по ограничениям алгоритмов адаптивного управления, например ограничения на длительности фаз, последовательности фаз, ограничения на минимальное время между управляющими воздействиями, который использует среду микроскопического моделирования транспорта с открытым исходным кодом SUMO. Фреймворк разделяет функционал контроллеров (класс *TrafficController*) и систему наблюдения и детектирования (класс *StateObserver*), что повторяет архитектуру реальных светофорных объектов и систем адаптивного управления и упрощает апробацию новых алгоритмов, так как можно свободно варьировать сочетания разных контроллеров и систем детектирования транспортных средств. Также в отличие от большинства существующих решений добавлен класс дороги *Road*, который объединяет набор полос, это позволяет, например, определить смежность регулируемых перекрестков, в случаях когда на пути от одного перекрестка к другому количество полос меняется, а следовательно, граф дороги разбивается на несколько ребер. При это сами алгоритмы используют одинаковый интерфейс и абстрагированы от конкретных параметров детекторов, топологии сети, то есть предполагается, что это решение позволит транспортному инженеру протестировать уже готовые алгоритмы для нового сценария, без необходимости их адаптации под новые условия, что ускоряет процесс разработки управляющей системы и снижает накладные расходы на проектирование. В настоящий момент в пакете есть примеры алгоритмов MaxPressure и метода обучения с подкреплением Q-learning, база примеров также пополняется. Также фреймворк включает в себя набор сценариев SUMO для тестирования алгоритмов, в который входят как синтетические карты, так и хорошо верифицированные SUMO-сценарии, такие как Cologne и Ingolstadt. Кроме того, фреймворк предоставляет некоторый набор автоматически подсчитываемых метрик, таких как полное время в пути, время задержки, средняя скорость; также в фреймворке представлен готовый пример для визуализации метрик.

Ключевые слова: транспортное моделирование, обучение с подкреплением, адаптивное управление, микроскопическое моделирование

COMPUTER RESEARCH AND MODELING
2024 VOL. 16 NO. 1 P. 69–78
DOI: 10.20537/2076-7633-2024-16-1-69-78

Ки&М

SPECIAL ISSUE

UDC: 519.8

Framework sumo-atclib for adaptive traffic control modeling

V. I. Kazorin[a], Ya. A. Kholodov[b]

Innopolis University,
1 Universitetskaya st., Innopolis, 420500, Russia

E-mail: [a] v.kazorin@innopolis.university, [b] ya.kholodov@innopolis.ru

Received 25.12.2023
Accepted for publication 25.12.2023.

This article proposes the *sumo-atclib* framework, which provides a convenient uniform interface for testing adaptive control algorithms with different limitations, for example, restrictions on phase durations, phase sequences, restrictions on the minimum time between control actions, which uses the open source microscopic transport modeling environment SUMO. The framework shares the functionality of controllers (class *TrafficController*) and a monitoring and detection system (class *StateObserver*), which repeats the architecture of real traffic light objects and adaptive control systems and simplifies the testing of new algorithms, since combinations of different controllers and vehicle detection systems can be freely varied. Also, unlike most existing solutions, the road class *Road* has been added, which combines a set of lanes, this allows, for example, to determine the adjacency of regulated intersections, in cases when the number of lanes changes on the way from one intersection to another, and therefore the road graph is divided into several edges. At the same time, the algorithms themselves use the same interface and are abstracted from the specific parameters of the detectors, network topologies, that is, it is assumed that this solution will allow the transport engineer to test ready-made algorithms for a new scenario, without the need to adapt them to new conditions, which speeds up the development process of the control system, and reduces design overhead. At the moment, the package contains examples of MaxPressure algorithms and the Q-learning reinforcement learning method, the database of examples is also being updated. The framework also includes a set of SUMO scripts for testing algorithms, which includes both synthetic maps and well-verified SUMO scripts such as Cologne and Ingolstadt. In addition, the framework provides a set of automatically calculated metrics, such as total travel time, delay time, average speed; the framework also provides a ready-made example for visualization of metrics.

Keywords: transport modeling, reinforcement learning, adaptive control, microscopic modeling

Citation: *Computer Research and Modeling*, 2024, vol. 16, no. 1, pp. 69–78 (Russian).

Введение

Несмотря на большое количество результатов в области адаптивного управления транспортными потоками, эта задача все еще актуальна, что подтверждается регулярным появлением новых исследований. Сейчас также активно развиваются новые подходы, основанные на последних результатах в области обучения с подкреплением (Reinforcement Learning, далее RL) и искусственных нейронных сетей. Однако наблюдается некоторое несоответствие теоретических и практических результатов. Это связано с тем, что, как правило, в алгоритмы изначально строятся на модельных данных, которые в реальной ситуации получить затруднительно, особенно при управлении в режиме реального времени, также сам способ управления, например выбор следующей фазы в режиме реального времени, — довольно трудно реализуемый на практике способ из-за ограничений аппаратной части (контроллеры, связь со светофорным объектом) или существующих локальных нормативов.

Так, например, в абсолютном большинстве исследованных работ (таблица 1) предлагаемые алгоритмы управляют светофорными объектами методом выбора следующей фазы, то есть каждые несколько секунд алгоритм решает оставить текущую фазу или выбрать любую из списка фаз. Однако на практике такое управление не всегда доступно технически. Чаще самой доступной опцией является обновление длительностей фаз один раз в более длительный период времени, например 15 минут.

Таблица 1. Анализ статей по адаптивному управлению трафиком

Метод управления	Количество статей	Доля, %
распределение времени между фазами	5	12
сменить/продлить фазу	8	19
выбрать следующую фазу	26	62
определить длительность фазы	3	7

Такое смещение фокуса в разрабатываемых алгоритмах адаптивного управления повлекло и смещение в разработанных инструментах. Так, в основных открытых бенчмарках и готовых фреймворках доступен только один вид управления светофорными объектами. Все это усложняет прямое сравнение алгоритмов, так как в разных исследованиях они могли использовать разные методы управления. Также затрудняется моделирование работы выбранного алгоритма для конкретного перекрестка, так как, скорее всего, придется реализовывать его самостоятельно под существующие условия.

В данной работе предлагается унифицированный фреймворк для разработки, тестирования алгоритмов адаптивного управления трафиком, который дает больший набор доступных методов управления, легко адаптирует один раз написанный алгоритм к конкретной дорожной сети. Фреймворк написан на языке python и использует в качестве среды для моделирования транспорта пакет SUMO [Lopez et al., 2018].

Существующие решения

Существующие фреймворки разработки алгоритмов адаптивного управления используют готовые открытые пакеты для моделирования транспортных потоков, среди которых можно выделить SUMO, CityFlow [Zhang et al., 2019]. SUMO (Simulation of Urban MObility) — это пакет для моделирования дорожного трафика с открытым исходным кодом. Он наиболее широко распространен, имеет графический интерфейс, широкий функционал (вплоть до симуляции пешеходов,

общественного транспорта, выбор алгоритмов следования за лидером и т. д.), регулярно обновляется. Несмотря на то что написан на языке C++, имеет коннекторы для различных языков, которые позволяют управлять симуляцией в режиме реального времени.

CityFlow изначально строился как среда для построения RL-алгоритмов адаптивного управления. Заявлялось, что этот пакет имеет большее быстродействие по сравнению с SUMO в высоконагруженных сценариях [Zhang et al., 2019], что скорее было связано с TraCI — коннектором к SUMO, так как в статье [Mei et al., 2023] показано, что время симуляции одинаковых сценариев сравнимо. Также этот пакет не обновлялся с ноября 2021 года.

Для разработки и отладки RL-алгоритмов на текущий момент можно выделить два фреймворка: sumo-rl и LibSignal.

Фреймворк sumo-rl [Alegre, 2019] использует SUMO как среду для моделирования транспорта. В этом фреймворке также присутствует набор сценариев RESCO [Ault, Sharon, 2021], есть пример реализованного агента. Однако в нем реализовано только управление посредством выбора следующей фазы. Также пространство возможных действий в нем довольно бедно, оно состоит только из возможности выбора следующей фазы, но в нем отсутствует информация о смежных перекрестках и давлении, в терминах MaxPressure-алгоритма, считается только общее по перекрестку, что не дает возможности напрямую применить этот алгоритм без модификации фреймворка.

Фреймворк LibSignal [Mei et al., 2023] в качестве модуля моделирования может также использовать SUMO, но вместе с тем есть возможность использовать CityFlow. Однако этот фреймворк также не имеет возможности моделирования ситуации с управлением по расписанию. Однако он более гибкий и уже предлагает больше возможностей по кастомизации пространства состояний, однако это требует довольно глубокого погружения в код фреймворка.

Предлагаемый фреймворк

В предлагаемом фреймворке также было принято использовать SUMO как среду для моделирования транспортного потока. Это наиболее популярный на сегодняшний день пакет, регулярно обновляется, имеет открытый исходный код. Также в фреймворк входит набор сценариев для тестирования алгоритмов — RESCO.

Структурная схема предлагаемой библиотеки представлена на рис. 1.

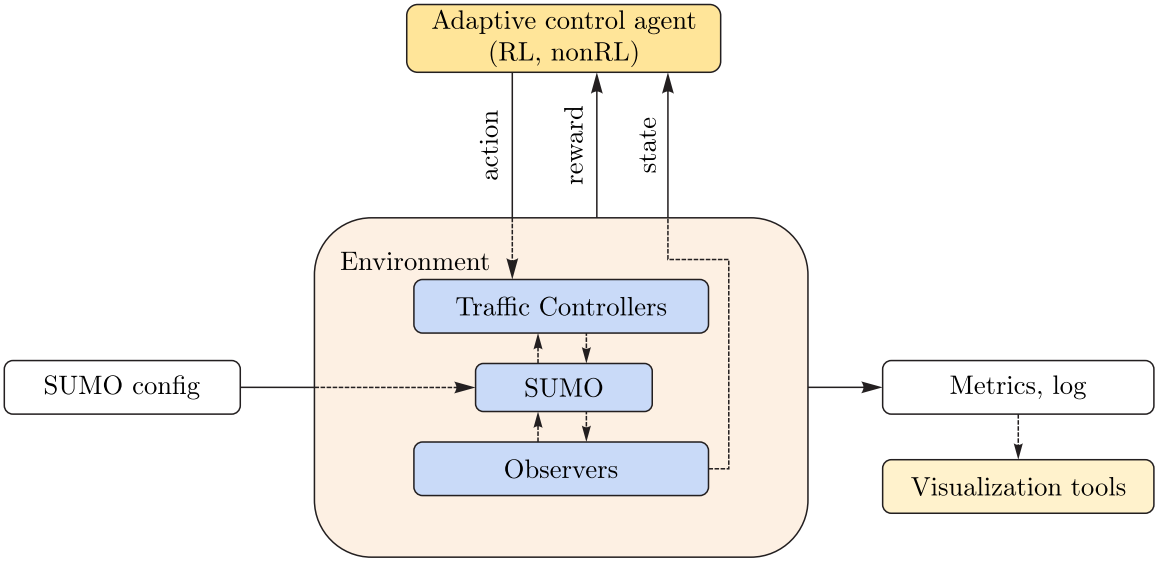

Рис. 1. Структурная схема фреймворка

Среда состоит из симуляции SUMO, контроллеров перекрестков (Traffic Controllers на схеме) и системы наблюдения (State Observers). Контроллеры управляют светофорами и реализованы в двух классах — TrafficLightsScheduleController и TrafficRealTimeController, первый позволяет имитировать управление по расписанию, дает возможность устанавливать допустимые длительности фаз, на некоторый промежуток времени, не подразумевает управление в реальном времени, в то время как второй класс контроллеров, наоборот, имитирует управление в режиме реального времени и запрашивает с некоторой периодичностью (например 5 секунд) действие, в зависимости от инициализации позволяет управлять в двух режимах: продление текущей фазы / включение следующей (циклический режим) или выбор следующей фазы из всех возможных (ациклический режим). Таким образом, доступны три способа управления — по расписанию, переключение на следующую фазу, выбор следующей фазы, причем в одной симуляции можно ставить на разные перекрестки разные контроллеры, что позволяет имитировать существующий аппаратный стек конкретной дорожной сети для отработки алгоритмов адаптивного управления.

Еще одной особенностью предлагаемого фреймворка является объединение полос (lane в SUMO) в «дороги», класс Road. Это связано с тем, что в SUMO, да и в других средах тоже, дорога от одного перекрестка до другого может состоять из нескольких участков (ребер на графе), то есть путь один, но он состоит из нескольких последовательно соединенных ребер, такое может быть, например, из-за сужений/расширений; таким образом, если автоматически строить граф для нового сценария, может получиться, что алгоритм управляет полосами несколько десятков метров на входе и на выходе и все светофоры между собой не соединены, хотя на самом деле они смежные. Для того чтобы бороться с такой проблемой, и был введен класс Road, который объединяет полосы, которые составляют одну ветвь.

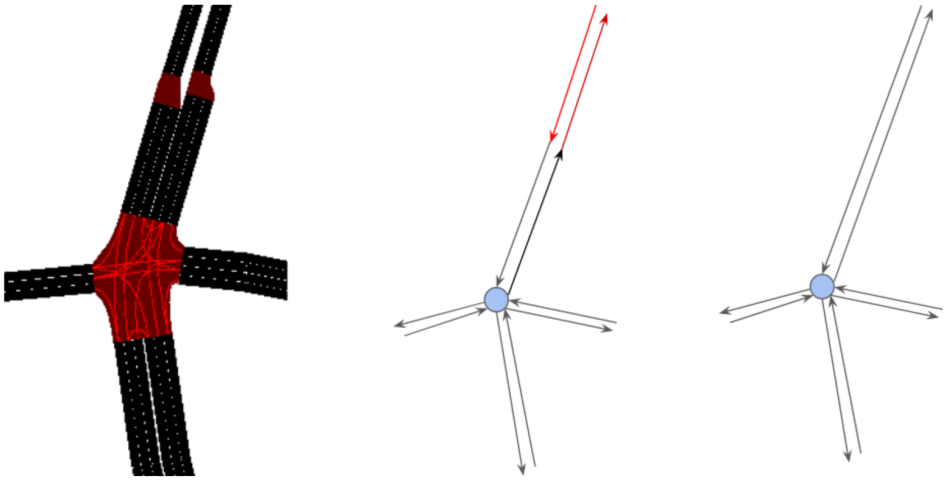

Рис. 2. Объединение полос в одну дорогу (Road)

Класс Observer имитирует работу систему наблюдения на дорожной сети, позволяет, например, задавать область наблюдения.

В данном фреймворке возможно тестировать не только алгоритмы, основанные на обучении с подкреплением, но и классические, основанные на модели, как, например, MaxPressure [Varaiya, 2013] или эволюционные алгоритмы в случае управления по расписанию.

Эксперименты

В данной секции демонстрируются возможности фреймворка. Для построения метрик были использованы методы MaxPressure в режиме цикла и Qlearning-подход, основанный на мето-

дах обучения с подкреплением, и также были получены метрики для расписания подобранного вручную. Методы протестированы на SUMO-модели двух связанных перекрестков Москвы, карта представлена на рис. 3.

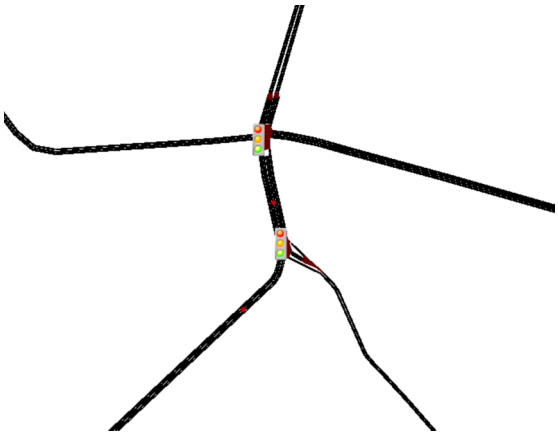

Рис. 3. На карте северный перекресток (пересечение Атласова – Москвитина – Чумакова) и южный (пересечение Атласова – Валуевское шоссе – Хабарова)

Описание алгоритмов

Max-pressure: алгоритм, предложенный в работе [Varaiya, 2013], минимизирует давление на перекрестке. Давление фазы, неформально говоря, определяется как разность количества транспортных средств в очереди на входящих дорогах перекрестка, проезд которым разрешает фаза, и количества авто, стоящих в очередях на смежных перекрестках исходящих дорог. Это децентрализованный алгоритм — он управляет только одним светофорным объектом, используя информацию со смежных пересечений, однако утверждается, что при некоторых условиях он гарантирует стабилизацию очередей на всех пересечениях сети.

Более формально: сначала считается *вес* направления по следующему правилу:

$$w(l, m) = q(l, m) - \sum_{n \in Out_m} p(m, n) \cdot q(m, n),$$

где $q(l, m)$ представляет длину очереди с ребра l на ребро m, $p(m, n)$ — пропорция ТС, направляющихся из m в n. *Давление* тогда считается как

$$\gamma_\phi = \sum_{(l,m) \in \phi} w(l, m),$$

где ϕ — набор разрешенных для фазы направлений проезда пересечения в виде пар «вход – выход» (l, m).

Стандартная реализация алгоритма, предложенного в работе [Varaiya, 2013]. Вариант с управлением в режиме реального времени, где последовательность фаз может не соблюдаться. Алгоритм описан ниже.

На рис. 4 представлен листинг программы, реализующей этот алгоритм с использованием интерфейса предлагаемого фреймворка.

Описанный алгоритм предполагает управление в режиме реального времени, однако это не всегда применимо на практике, поэтому часто используется его модификация, которая не меняет порядок фаз в цикле, а только распределяет время длительности каждой фазы в цикле. В отличие от исходного алгоритма здесь длительность фаз цикла просто устанавливается пропорционально

Алгоритм 1: Max-Pressure

Исходные параметры: Минимальная длительность фазы $t_{\min}$
Время с начала текущей фазы i-го пересечения t_i
ϕ_i, $\phi_{i,0}$ — текущая и начальная фаза i-го пересечения
Для каждого *i-го пересечения* **выполнять**
  $\phi_i \leftarrow \phi_{i,0}$;
  $t_i \leftarrow 0$;
конец
Для каждого *такта времени* **выполнять**
  Для каждого *i-го пересечения* **выполнять**
    Если $t_i \geqslant t_{\min}$ **тогда**
      Рассчитать давление пересечения i для всех фаз $\gamma_{i,\phi}$;
      Если $\arg\max_{\phi} \gamma_{i,\phi} \neq \phi_i$ **тогда**
        Текущая фаза меняется на $\arg\max_{\phi} \gamma_{i\phi}$;
        $t_i \leftarrow 0$;
      иначе
        Не менять фазу;
      конец
    конец
  конец
конец

```
env = SumoEnvironment(args)
initial_state, _, done, _ = env.reset()
actions = {}
while not done["__all__"]:
    obs, _, done, i = env.step(actions)
    actions = {}
    for ts in obs.keys():
        pressures = obs[ts]["pressure"]
        actions[ts] = np.argmax(pressures)
env.close()
```

Рис. 4. Max-Pressure со свободным выбором фаз, реализация в предлагаемом интерфейсе

```
env = SumoEnvironment(args)
initial_state, _, done, _ = env.reset()
actions = {}
while not done["__all__"]:
    obs, _, done, i = env.step(actions)
    actions = {}
    for ts in obs.keys():
        pressures = obs[ts]["pressure"]
        actions[ts] = np.argmax(pressures)
env.close()
```

Рис. 5. Циклический Max-Pressure

соответствующим давлениям. Реализовать в предлагаемом фреймворке этот вариант также не сложно, и на рис. 5 представлен листинг этого алгоритма в рамках фреймворка.

В отличие от классических алгоритмов в алгоритмах обучения с подкреплением (Reinforcement Learning) есть агент, который обучается в процессе взаимодействия со средой, об-

разно говоря, получая опыт и делая выводы. Для того чтобы продемонстрировать возможности фреймворка в поддержке методов RL, также был реализован пример обучения агента методом Q-learning. Псевдокод алгоритма представлен в алгоритме 2.

Алгоритм 2: Q-learning

Исходные параметры: α — параметр сглаживания
ϵ — параметр исследования
$Q(s, a)$ — таблица значений Q-функции
$s \in S$ — вектор состояния (очереди, давление, ...)
$a \in A$ — действие (номер фазы)
s_0 — начальное состояние
Для каждого *шага* k **выполнять**
  a_k с вероятностью ϵ выбирается равновероятно из A и с вероятностью $1 - \epsilon$ определяется как $a_k \leftarrow \arg\max_a Q(s_k, a)$
  r_k, s_{k+1} — результат шага среды после действия a_k
  Обновление Q: $Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha\left(r_k + \gamma \max_a Q(s_{k+1}, a) - Q(a_k, s_k)\right)$
конец

Это пример off-policy-метода, так как агент, вообще говоря, может учиться не только на примерах своего взаимодействия со средой, но и на сэмплах других агентов (политик). То есть можно использовать реплэй-буфер (experience replay) и, вообще говоря, отдельно собирать примеры и обновлять Q-таблицу.

Результаты

Результаты моделирования представлены на рис. 6 и 7, где *экспертное расписание* — длительности фаз — подобраны экспертно и фиксировано на протяжении всего эпизода; *MaxPressure* (*циклический*) — алгоритм *MaxPressure*, где длительности фаз устанавливаются один раз перед циклом, последовательность фаз остается неизменной; *MaxPressure* (*свободный выбор фаз*) — алгоритм *MaxPressure*, где каждые 5 секунд выбирается новая фаза или остается текущая, последовательность фаз меняется; *Q-agent* (*циклический*) — реализация подхода Q-learning, последовательность фаз не меняется, алгоритм каждые 5 секунд решает, переключить на следующую фазу или нет; *Q-agent* (*свободный выбор фаз*) — реализация подхода Q-learning, в которой алгоритм каждые 5 секунд выбирает следующую фазу, последовательность фаз меняется. Код экспериментов можно найти в репозитории проекта: https://github.com/zhelyazik/sumo-atclib/tree/main/experiments [Kazorin, 2023].

Заключение

Цель данной работы — разработать инструмент, который значительно ускорит разработку и исследование алгоритмов адаптивного управления трафиком. Изначально был взят готовый фреймворк (sumo-rl), который в результате правок был значительно переработан, и расширен его функционал. Фреймворк доступен на GitHub по ссылке https://github.com/zhelyazik/sumo-atclib [Kazorin, 2023].

Этот инструмент может быть полезен для инженеров и исследователей области управления трафиком для моделирования и тестирования разрабатываемых алгоритмов адаптивного управления.

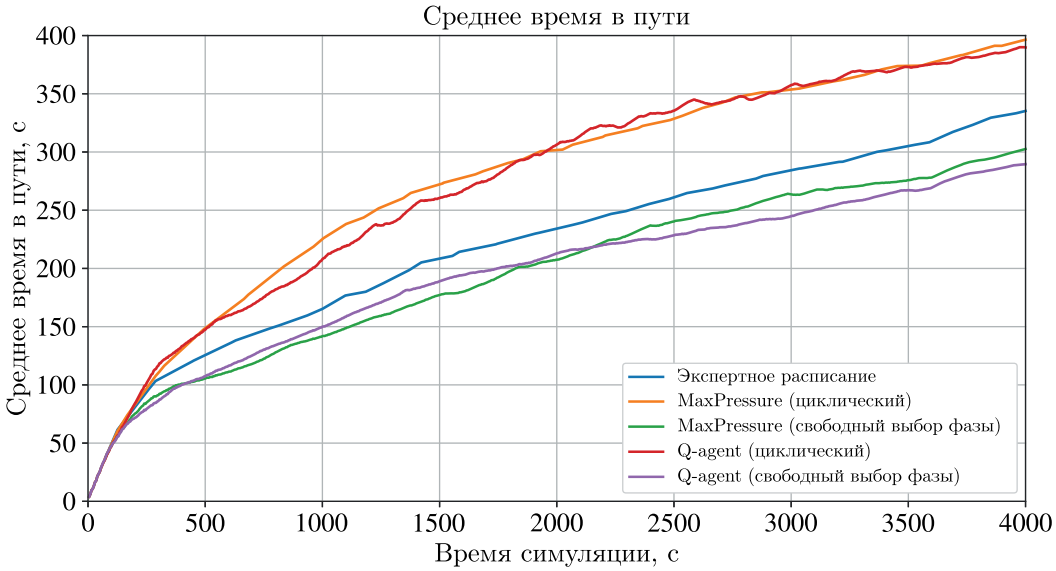

Рис. 6. График среднего времени поездки на один автомобиль в зависимости от шага симуляции, в расчете используются совместно автомобили, находящиеся в пути и прибывшие в пункт назначения

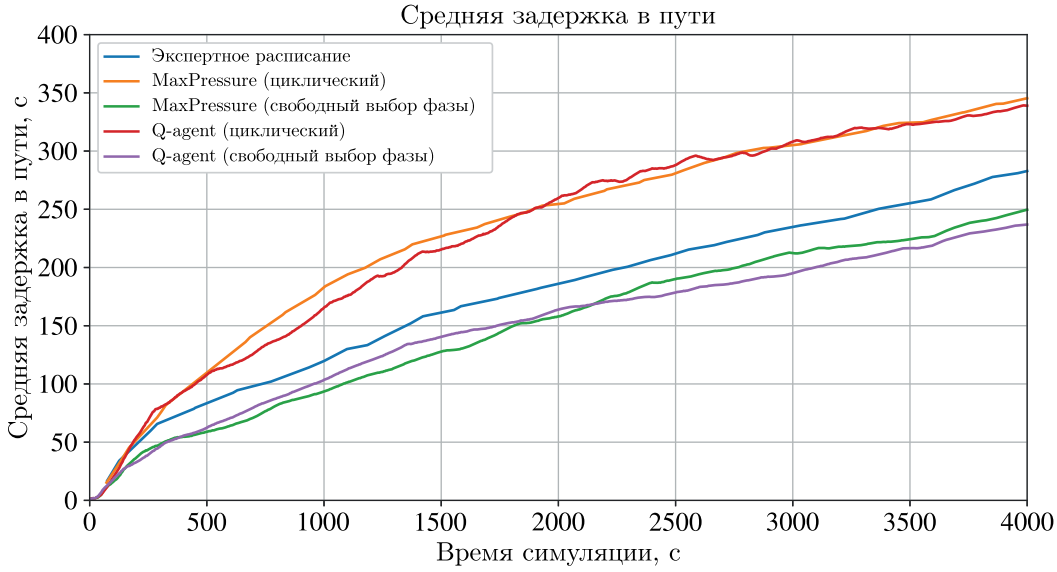

Рис. 7. График средней задержки в поездке на один автомобиль в зависимости от шага симуляции, в расчете используются совместно автомобили, находящиеся в пути и прибывшие в пункт назначения. Время задержки транспортного средства считается как суммарное время всех остановок

Однако в текущей версии есть несколько ограничений. Например, функция награды, которая используется для обучения RL-агентов, пока реализована внутри фрейиворка и нет возможности подключить пользовательскую версию. Аналогичная ситуация с классом StateObserver — пока что зафиксирован, и нет возможности дополнительной настройки, однако разработчики могут сами менять фреймворк при необходимости. В настоящий момент реализованы контроллеры, которые позволяют управлять светофорными объектами в режиме выбора следующей фазы (любой) в режиме реального времени, переключение на следующую фазу в цикле по порядку, установка распределения времени по фазам в цикле (расписание).

Несмотря на уже реализованный функционал, дальнейшая разработка необходима для того, чтобы сделать фреймворк более гибким и удобным для инженеров. Первоочередной задачей выглядит задача наполнения фреймворка, дополнительно к базе сценариев, набором готовых реализаций алгоритмов адаптивного управления.

Список литературы (References)

Alegre L. N. SUMO-RL // https://github.com/LucasAlegre/sumo-rl (accessed: 20.12.2023).

Ault J., Sharon G. Reinforcement learning benchmarks for traffic signal control // Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). — 2021.

Devailly F. X., Larocque D., Charlin L. Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control // IEEE Transactions on Intelligent Transportation Systems. — 2021. — Vol. 23, No. 7. — P. 7496–7507.

Kazorin V. SUMO-ATCLIB // https://github.com/zhelyazik/sumo-atclib (accessed: 25.12.2023).

Lopez P. A., Behrisch M., Bieker-Walz L., Erdmann J., Flötteröd Y.-P., Hilbrich R., Lücken L., Rummel J., Wagner P., Wiessner E. Microscopic traffic simulation using sumo // 2018 21st international conference on intelligent transportation systems (ITSC). — IEEE, 2018. — P. 2575–2582.

Mei H., Lei X., Da L., Shi B., Wei H. Libsignal: an open library for traffic signal control // Machine Learning. — 2023. — P. 1–37.

Sun X., Yin Y. A simulation study on max pressure control of signalized intersections // Transportation research record. — 2018. — Vol. 2672, No. 18. — P. 117–127.

Sutton R. S., Barto A. G. Reinforcement learning: An introduction. — MIT press, 2018.

Varaiya P. A universal feedback control policy for arbitrary networks of signalized intersections // Published online. — 2009. — http://paleale.eecs.berkeley.edu/varaiya/papers/ps.dir/090801-Intersectionsv5.pdf

Varaiya P. Max pressure control of a network of signalized intersections // Transportation Research Part C: Emerging Technologies. — 2013. — Vol. 36. — P. 177–195.

Wegener A., Piórkowski M., Raya M., Hellbrück H., Fischer S., Hubaux J.-P. TraCI: an interface for coupling road traffic and network simulators // Proceedings of the 11th communications and networking simulation symposium. — 2008. — P. 155–163.

Wei H., Zheng G., Gayah V., Li Zh. A survey on traffic signal control methods // arXiv preprint. — 2019. — arXiv:1904.08117

Zhang H., Feng S., Liu Ch., Ding Y., Zhu Y., Zhou Z., Zhang W., Yu Y., Jin H., Li Zh. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario // The world wide web conference. — 2019. — P. 3620–3624.