Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ДЕЦЕНТРАЛИЗОВАННЫЙ АЛГОРИТМ УПРАВЛЕНИЯ КОНВЕЙЕРНОЙ СИСТЕМОЙ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МУЛЬТИАГЕНТНОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Автор: Мухутдинов Дмитрий Вадимович _____

Направление подготовки: 01.04.02 Прикладная математика и информатика

Квалификация: Магистр

Руководитель: Фильченков А.А., к.ф-м.н. _____

К защите допустить

Руководитель ОП Парфенов В.Г., проф., д.т.н. _____

«____» _____ 20___ г.

Санкт-Петербург, 2019 г.

Студент Мухутдинов Д.В. Группа М4239 Факультет ИТиП

Направленность (профиль), специализация
Технологии разработки программного обеспечения

Консультанты:

а) Вяткин В.В., проф., д.т.н. _____

ВКР принята «_____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «_____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Конвейерные системы широко применяются для автоматизированной транспортировки объектов и материалов. Они являются неотъемлемой частью комплексов промышленного оборудования, пунктов сортировки грузов, багажных систем в аэропортах и т. д.. Оптимизация работы таких системы имеет высокую практическую и экономическую значимость, что мотивирует поиск эффективных методов управления конвейерными системами.

Для управления большинством современных конвейерных систем применяются централизованные статические стратегии управления, специально разрабатываемые под конкретную систему одновременно с проектированием ее физической топологии в целях решения задач конкретного предприятия [9]. Преимуществами такого подхода являются высокая производительность работы и предсказуемость поведения системы. Недостатками такого подхода являются высокая стоимость и долгие сроки разработки кастомизированной стратегии управления, неспособность системы адаптироваться к изменениям во внешний условиях, не учтенных во время проектирования (таким как неожиданный отказ отдельных элементов системы), а также наличие централизованного контроллера как критической точки отказа.

В связи с обозначенными недостатками использования кастомизированных стратегий управления существует интерес к разработке обобщенных систематических подходов к управлению конвейерными системами. Наиболее популярным подходом из используемых является управление с прогнозируемыми моделями (*англ.* model predictive control, MPC) [29]. Управление с прогнозируемыми моделями изначально разрабатывалось для задач химической промышленности и нефтепереработки [13], и на данный момент широко используется в этих сферах. В последние несколько лет повысился интерес к применению данного подхода к управлению другими типами киберфизических систем, в том числе промышленными конвейерными линиями [7, 22] и системами распределения багажа на основе рельсовых тележек (*англ.* destination coded vehicles, DCVs) [31, 36]. Существующие алгоритмы на основе управления с прогнозируемыми моделями обобщаются на различные конкретные конфигурации физических систем, но все еще предусматривают наличие централизованного контроллера. Кроме того, модификация существующего алгоритма управления (например, добавления учета энергопотребления системы)

во многих случаях является нетривиальной задачей, так как оптимизируемая функция во фреймворке MPC должна выражаться как задача линейного (*англ.* linear programming, LP) или квадратичного программирования (*англ.* quadratic programming, QP). В связи с этим существует интерес к разработке иных, децентрализованных подходов к управлению конвейерными сетями, требующих менее строгих ограничений к характеру решаемой задачи.

При рассмотрении штучных конвейеров, перемещающих отдельные объекты, как в случае систем для перемещения багажа в аэропортах, задача управления конвейерной системой по большей части сводится к задаче пакетной маршрутизации (*англ.* packet routing). Задача пакетной маршрутизации — это задача поиска пути наименьшей стоимости в графе из текущей вершины n в вершину назначения d. В контексте конвейерных систем топология конвейерной сети может быть представлена в виде ориентированного графа, а перемещаемые по конвейерам объекты могут быть абстрагированы как «пакеты».

Задача пакетной маршрутизации впервые обрела актуальность с появлением компьютерных сетей. Первые алгоритмы сетевой маршрутизации появились в процессе разработки сети ARPANet. Именно тогда были изобретены такие подходы к пакетной маршрутизации, как дистанционно-векторный (*англ.* distance-vector) [24] и состояния каналов связи (*англ.* link-state) [23], которые и по сей день лежат в основе таких стандартных и широко применяемых алгоритмов сетевой маршрутизации, как Routing Information Protocol (RIP) [16] и Open Shortest Path First (OSPF) [25]. Преимуществами алгоритмов сетевой маршрутизации являются обусловленные характером задачи децентрализованность, низкая требовательность к вычислительным ресурсам и устойчивость к отказам маршрутизаторов и обрывам соединений. В работе [35] было продемонстрировано, что простой дистанционно-векторный алгоритм маршрутизации может быть напрямую применен к задаче управления конвейерной системой для транспортировки багажа, что позволяет достигнуть устойчивости системы к отказам отдельных конвейеров. Однако, в силу своей простоты, дистанционно-векторный алгоритм решает исключительно задачу направления перемещаемых объектов вдоль кратчайших путей в конвейерной сети, что ограничивает его применимость в тех случаях, когда необходимо учитывать дополнительные критерии оптимизации (такие как энергопотребление системы).

Существуют и другие подходы к решению задачи пакетной маршрутизации. Одним из таких подходов является подход на основе идеи обучения с подкреплением (*англ.* reinforcement learing, RL). Первым таким алгоритмом стал алгоритм Q-routing [6], основанный на методе Q-learning [34]. Этот алгоритм, как и его модификации [8, 19], благодаря обучению с подкреплением оказался способен лучше адаптироваться к изменениям в интенсивности сетевого трафика, чем алгоритмы, основанные на вычислении кратчайшего пути. Но из-за использования большого количества служебных сообщений, использующих те же каналы передачи данных, что и целевые пакеты, применение таких алгоритмов в реальных компьютерных сетях ограничено.

В контексте конвейерных сетей, однако, целевые «пакеты» являются физическими объектами (например, чемоданами), перемещаемыми по конвейерной ленте, в то время как служебные сообщения передаются по проводным соединениям между контроллерами. Таким образом, время распространения служебных сообщений по системе пренебрежимо мало по сравнению с временем перемещения целевых объектов, что нивелирует озвученный недостаток алгоритмов маршрутизации, основанных на обучении с подкреплением.

В данной работе будет предложено несколько модификаций децентрализованного алгоритма управления конвейерной системой, основанного на методе Q-routing, но использующего нейронную сеть в качестве обучающегося агента. Для демонстрации способности работы алгоритма в различных постановках задачи маршрутизации он будет исследован как в имитационной модели конвейерной системы, так и в имитационной модели абстрактной компьютерной сети.

В главе 1 будут рассмотрены существующие подходы к управлению конвейерными системами и маршрутизации. Также будет сформулирована обобщенная постановка задачи маршрутизации в терминах мультиагентного обучения с подкреплением, к которой будет сведена задача управления конвейерной системой. Будут рассмотрены существующие алгоритмы маршрутизации, их сильные и слабые стороны. Также будут рассмотрены существующие методы обучения нейронных сетей с подкреплением, в том числе в мультиагентном случае.

В главе 2 будет рассмотрен предложенный алгоритм и его модификации и обоснованы решения, принятые в ходе его разработки.

В главе 3 будут приведены экспериментальные результаты работы алгоритма для задач пакетной маршрутизации в моделях компьютерной сети и управления системой багажных конвейеров. Также будет проведено экспериментальное сравнение алгоритма с существующими алгоритмами маршрутизации и управления конвейерной системой.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Методы управления киберфизическими системами

Определение 1. Управление с прогнозируемыми моделями (*англ.* model predictive control, MPC) — это...

TBD

1.2. Алгоритмы пакетной маршрутизации

Почти все существующие алгоритмы пакетной маршрутизации были разработаны для маршрутизации пакетов в компьютерных сетях. Большинство алгоритмов маршрутизации в компьютерных сетях, применяемых на практике, относятся к одному из двух семейств алгоритмов — дистанционно-векторные (distance-vector) [24] или состояния каналов связи (link-state) [23]. Концептуально все алгоритмы внутри каждого из этих семейств одинаковы, и различаются только техническими деталями реализации, обусловленными спецификой конкретной узкой сферы применения. Поэтому мы не будем рассматривать алгоритмы каждого семейства по отдельности, а рассмотрим только концепции в целом.

1.2.1. Дистанционно-векторные алгоритмы

Идея дистанционно-векторных алгоритмов (distance-vector algoritms) заключается в следующем:

- Каждый маршрутизатор s в сети хранит таблицу, в которой для каждого другого узла сети d хранится следующая информация:
  - Предполагаемое кратчайшее расстояние от s до d
  - Сосед n, которому нужно отправить пакет, чтобы пакет прошел по кратчайшему пути до узла d.
- Периодически каждый маршрутизатор рассылает свою версию таблицы кратчайших расстояний всем своим соседям
- При получении вектора кратчайших расстояний от соседа маршрутизатор s сравнивает его поэлементно с текущей версией. Если оказывается, что наименьшая стоимость пути от соседа n до узла d, сложенная с оценкой стоимости ребра (s, d) меньше, чем наименьшая стоимость пути от s до d в текущей таблице, то значение в текущей таблице обновляется, и наилучшим соседом для отправки пакета в узел d становится сосед n.

Как можно видеть, дистанционно-векторный алгоритм является, в сущности, распределенной версией алгоритма Беллмана-Форда поиска кратчайшего пути в графе [4].

Различные реализации дистанционно-векторного метода различаются, в частности, оценками стоимости соединений в сети. Так, например, протокол RIP [16] просто оценивает стоимость каждого соединения в 1, а IGRP [5] оценивает стоимость соединений исходя из оценок задержки и пропускной способности.

Преимуществами дистанционно-векторных алгоритмов являются простота реализации и низкие требования к памяти и вычислительной мощности. Недостатками же являются низкая скорость распространения информации по сети и сложности с приспособлением под изменяющуюся топологию (проблема count-to-infinity). Этих проблем удается избежать при применении другого распространенного подхода — алгоритмов на основе состояния канала связи.

1.2.2. Алгоритмы состояния канала связи

В отличие от дистанционно-векторных алгоритмов, в алгоритмах состояния канала связи (link-state) каждый узел сети хранит у себя модель всей сети в виде графа. Рассмотрим шаги алгоритма подробнее:

— Каждый маршрутизатор периодически проверяет состояние соединений до соседей
— При обнаружении обрыва какого-либо соединения алгоритм удаляет это соединение из собственного графа и рассылает соседям новую версию состояния соединений до них
— Соседи обновляют собственные версии графов в соответствии с полученной информацией и пересылают сообщение дальше
— Чтобы избежать зацикливания сообщений об обновлении состояния, каждое сообщение снабжается *номером версии*. Маршрутизатор n игнорирует сообщение от маршрутизатора n, если номер версии этого сообщения меньше или равен предыдущему.

Имея информацию обо всей сети в целом, маршрутизатор может рассчитать кратчайшие пути до всех остальных узлов. Обычно для этого используется алгоритм Дейкстры [11].

Link-state алгоритмы обладают способностью адаптироваться под изменения топологии сети гораздо быстрее, чем distance-vector алгоритмы за счет

несколько более сложной реализации и чуть больших затрат по памяти и вычислительной мощности. Это обуславливает то, что на данный момент именно link-state протоколы, такие как OSPF [25], доминируют в сетевой маршрутизации. Однако даже в решении задачи сетевой маршрутизации link-state алгоритмы в чистом виде не лучшим образом адаптируются к повышению нагрузки в сети. Рассматриваемые в дальнейшем другие алгоритмы, основанные на принципе обучения с подкреплением, справляются с задачей адаптации к изменчивой нагрузке лучше.

1.2.3. Q-routing

Среди других подходов особый интерес представляют подходы на основе обучения с подкреплением. Первым алгоритмом маршрутизации, основанным на этой идее, стал алгоритм Q-routing [6]. Принцип его работы таков:

— Каждый маршрутизатор x хранит $Q_x(d, y)$ — оценку минимального времени в пути до узла d, если следующим узлом на пути является сосед y. Очевидно, что $\forall y : Q_x(x, y) = 0$

— Пакет, который необходимо доставить в узел d, отправляется соседу $y = \text{argmin}_{(x,y)\in E} Q_x(d, y)$

— При получении пакета узел y отправляет узлу x время получения t_r и собственную оценку оставшегося времени в пути $t = \min\limits_{(y,z)\in E} Q_y(d, z)$

— Зная время отправления пакета t_s и получив t_r и t, узел x обновляет собственную оценку по формуле: $Q_x(d, y) = \alpha((t_r - t_s) + t - Q_x(d, y)) + Q_x(d, y)$, где α — это learning rate, параметр алгоритма.

Было показано, что этот алгоритм способен хорошо адаптироваться к изменениям в топологии сети и интенсивности трафика. Такие его модификации, как dual Q-routing [19] и predictive Q-routing [8] демонстрируют еще более высокое качество маршрутизации. Однако по сравнению с distance-vector или link-state методами данные алгоритмы используют гораздо больше служебных сообщений (служебный пакет на каждую пересылку целевого пакета), что ограничивает их применение в реальных высоконагруженных компьютерных сетях.

Однако в задачах маршрутизации вне контекста компьютерных сетей это перестает быть проблемой, так как целевые «пакеты» (чемоданы на конвейере, автомобили на автостраде, etc.) и служебные сообщения в таких задачах

являются объектами разной природы и передаются по разным каналам, причем служебные сообщения по сравнению с целевыми «пакетами» доставляются мгновенно. Эти обстоятельства делают применение алгоритмов обучения с подкреплением в таких задачах более привлекательным.

1.2.4. Применение машинного обучения в сетевой маршрутизации

TBD

1.2.4.1. Программно-определяемые сети (SDN)

TBD

1.2.5. Другие подходы

Для полноты обзора приведем еще несколько примеров.

Идея использования нейросетей для решения задачи маршрутизации не нова. В работах [2, 3] для решения задачи поиска кратчайшего пути в графе используются нейронные сети Хопфилда. Однако, эти исследования преследовали цель ускорения вычисления кратчайшего пути за счет аппаратной реализации нейросети, что кардинально отличается от цели текущей работы.

Еще одним интересным подходом является AntNet [10]. Это алгоритм, построенный на идее исследования состояния сети с помощью специальных пакетов-«агентов». Алгоритм показал хорошие результаты в ходе исследований, но не получил широкого применения, вероятно, в силу уже массового к тому времени распространения link-state и distance-vector протоколов.

1.3. Применение обучения с подкреплением к поставленной задаче

1.3.1. Термины и понятия

Определение 2. Обучение с подкреплением (*англ.* reinforcement learning, RL) — вид машинного обучения, в котором *агент* (agent) каждый момент времени t взаимодействует со *средой* (environment), находящейся в *состоянии* (state) $s_t \in \mathcal{S}$ путем выбора *действия* (action) $a \in \mathcal{A}_{s_t}$ и получения *вознаграждения* (reward) $r_{t+1} \in \mathbb{R}$ с переходом в новое *состояние* s_{t+1}.

Определение 3. Марковский процесс принятия решений (*англ.* Markov decision process, MDP) — это кортеж $(\mathcal{S}, \mathcal{A}_s, P, R, \gamma)$, где

— $\mathcal{S}$ — конечное множество состояний

— $\mathcal{A}_s$ — конечное множество действий, доступных из состояния s

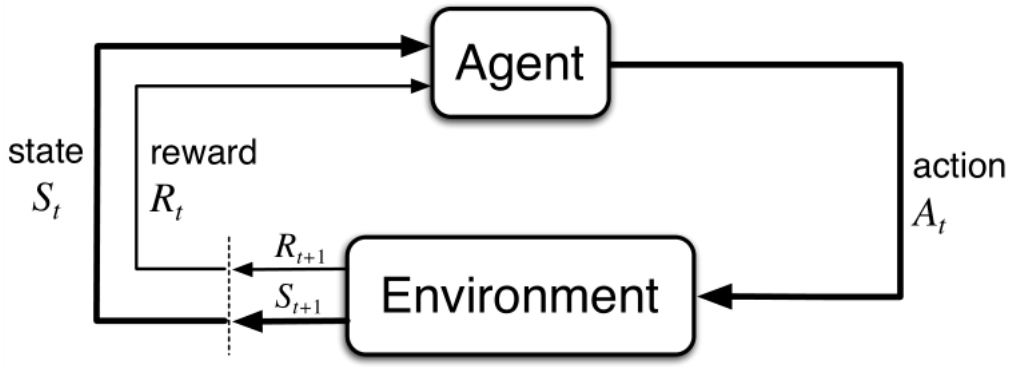


Рисунок 1 – Схема взаимодействия агента и среды в обучении с подкреплением

— $P(s'|s,a)$ — вероятность того, что действие a в состоянии s приведет к переходу в состояние s' в следующий момент времени.
— $R : \mathcal{S} \times \mathcal{A}_s \to \mathbb{R}$ — вознаграждение за действия a в состоянии s
— $\gamma \in [0,1]$ — *скидочный коэффициент* (discount factor), управляющий соотношением между важностью текущих вознаграждений и будущих вознаграждений.

Определение 4. Оптимальной стратегией для данного марковского процесса принятия решений называется такая функция выбора действий $\pi : \mathcal{S} \to \mathcal{A}_s$, что взвешенная сумма вознаграждений $\sum\limits_{t=0}^{\infty} \gamma^t R_{\pi(s_t)}(s_t, s_{t+1})$ максимальна.

Определение 5. Частично наблюдаемый марковский процесс принятия решений (Partially observed Markov decision process, POMDP) — это кортеж $(\mathcal{S}, \mathcal{A}_s, P, R, \Omega, O, \gamma)$, где
— $\mathcal{S}$ — конечное множество состояний
— $\mathcal{A}_s$ — конечное множество действий, доступных из состояния s
— $P(s'|s,a)$ — вероятность перехода из s в s' при выполнении действия a
— $R : \mathcal{S} \times \mathcal{A}_s \to \mathbb{R}$ — вознаграждение за действие a в состоянии s.
— Ω — множество *наблюдений*
— $O(o|s',a)$ — вероятность получения наблюдения o при переходе в истинное состояние s' в результате действия a.

Определение оптимальной стратегии для частично наблюдаемого марковского процесса аналогично таковому для обычного.

Для марковских процессов в условиях известности всех компонентов, включая P, R и O существуют детерминированные методы нахождения опти-

мальной стратегии. Однако найти оптимальную стратегию можно и в условиях неизвестности P, R и O.

Определение 6. Q-обучение (*англ.* Q-learning) [34] — это метод нахождения оптимальной стратегии для марковского процесса принятия решений, не требующий информации о функции R и распределении P. Метод заключается в оценке *функции полезности* (action-value function) $Q(s,a)$. Функция полезности изменяется при каждом предпринятом действии a с переходом из состояния s в s' по следующей формуле:

$$Q(s,a) = Q(s,a) + \alpha \left(r + \gamma \cdot \max_{a \in \mathcal{A}_{s'}} -Q(s',a) \right) \tag{1}$$

Известно, что для любого конечного марковского процесса принятия решений Q-обучение находит оптимальную стратегию, т. е. $Q(s,a) \xrightarrow{t \to \infty} Q^*(s,a)$, и $\pi(s) = \text{argmax}_{a \in \mathcal{A}_s} Q^*(s,a)$ — оптимальная стратегия.

1.3.2. Формулировка задачи в терминах обучения с подкреплением

Попытаемся сформулировать глобальную задачу управления конвейерной системой в терминах обучения с подкреплением. Пусть множество состояний $\mathcal{S}$ — это декартово произведение состояний всех конвейеров (конвейерных секций). Пусть также в состояние каждого конвейера (секции) входит информация о всех объектах (сумках, проходящих через этот конвейер (секцию). Пусть множество действий, доступных из состояния $s \in \mathcal{S}$ $\mathcal{A}_s$ — это множетво пар вида (x,y), где $x \in V$ — это узел сочленения конвейеров, к которому направляется сумка, а $y \in \{0,1\}$ — действия «пропустить сумку» и «столкнуть сумку на соседний конвейер».

У такой постановки задачи есть следующие проблемы:

а) Некоторые действия должны выполняться одновременно, а не одно за другим. Это можно формально обойти, сказав, что одновременно выполняющиеся действия $a_k, ..., a_{k+n}$ выполняются в последовательные моменты времени $t_k, ..., t_{k+n}$, либо изменив структуру множества действий $\mathcal{A}_s$: вместо пар (x,y) рассматривать списки таких пар.

б) В любом случае невозможно узнать вознаграждение за действие a_t сразу после перехода из состояния s_t в s_{t+1} — вознаграждение за посылку

сумки p по конвейеру e к точке сочленения v вычисляется как

$$-\sum_{t=t_k}^{t_{k+n}} R_e(e, s_e^t, p)$$

, где $t_k ... t_{k+n}$ — моменты времени, в которые сумка ехала по конвейеру e, а $t_{k+n+1} ... t_{k+m}$ до точки сочленения v. (Суммы взяты с минусом, чтобы максимизация суммарного вознаграждения минимизировала суммарную стоимость путей сумок). Это можно формально обойти, добавив в множество действий $\mathcal{A}_s$ действия вида a_e^p — «продолжить движение сумки p по конвейеру e». Однако такую формальную модель будет крайне сложно как-то применить на практике — непонятно, как в какой-либо реальной задаче получить по отдельности стоимости вида $R_e(e, s_e^t, p)$ — стоимости «пребывания сумки p на конвейере e, который находится в состоянии s».

Как можно видеть, несмотря на то, что формально процесс маршрутизации в конвейерной сети можно смоделировать как марковский решающий процесс, оптимизация такой модели невозможна с практической точки зрения как минимум потому, что в реальных задачах полную и точную информацию о состоянии всей сети в любой момент времени получить нельзя.

Гораздо более логичным кажется рассмотреть задачу с точки зрения отдельного конвейера (а точнее, перенаправляющего манипулятора (*англ.* diverter) в точке сочленения, так как именно он осуществляет перенаправления сумок) как обучающегося агента. Используя подход *независимого Q-обучения* (*англ.* independent Q-learning, IQL) [30], скажем, что манипулятор считает всю остальную часть сети как среду, с которой он взаимодействует. Безусловно, манипулятору недоступна исчерпывающая информация о состоянии всей сети, а только небольшая ее часть — свое собственное состояние, и, возможно, какая-то еще информация (о конвейере, на котором он находится, о соседних манипуляторах, возможно, что-то еще). Таким образом, кажется, что работу такого агента можно смоделировать как частично наблюдаемый марковский процесс. Однако и здесь мы встречаемся с проблемой не мгновенного получения вознаграждения — к тому времени, как агент узнает от среды (а именно — от соседа, которому он послал сумку) финальную стоимость прохождения сумки до соседа, он успеет сменить множество состояний. Более того, в такой

постановке задачи состояние s_{t+1} почти никак не зависит от действия a_t. К тому же такая постановка мотивирует манипулятор все время посылать сумки по направлению наименьшей стоимости, максимизируя таким образом свой собственный выигрыш, что явно не оптимизирует суммарную стоимость пути сумки.

Однако все встает на свои места, если мы сделаем формальный трюк и в качестве агента, взаимодействующего со средой, рассмотрим *сумку*, двигающуюся по конвейерам и стремящуюся минимизировать суммарную стоимость своего пути. Сумка рассматривает всю сеть как среду, а наблюдаемое состояние точки сочленения, у которой она находится, включая идентификатор этой точки, как собственное *наблюдение* $o_t \in \Omega$. С точки зрения сумки, во время ее движения по конвейеру между сочленениями ничего не происходит — при поступлении в точку сочленения сумка выбирает действие $a_t \in \mathcal{A}_o$ — «ехать дальше» или «перенаправиться», «мгновенно» перемещается в соответствующую точку сочленения дальше по пути с получением вознаграждения r_t, равного суммарной стоимости этого перемещения, взятого с отрицательным знаком, и получает новое наблюдение $o_{t+1} \in \Omega$ — состояние новой точки.

Как можно видеть, в такой формулировке задача хорошо моделируется как частично наблюдаемый марковский процесс. Для нахождения оптимальной стратегии в частично наблюдаемом марковском процессе также можно использовать принцип Q-обучения. Несмотря на то, что в общем случае в POMDP Q-обучение не сходится к оптимальной стратегии, на практике это дает хорошие результаты.

Запишем формулу Q-обучения для поставленной задачи:

$$Q(o_t, a_t) = Q(o_t, a_t) + \alpha \left(r_t + \gamma \cdot \max_{a \in \mathcal{A}_{o_{t+1}}} Q(o_{t+1}, a) \right) \tag{2}$$

Заметим, что путь пакета конечен, и нас интересует оптимизация его полной стоимости. Поэтому в данной задаче будем считать, что $\gamma = 1$.

Если наблюдение состоит только из идентификатора текущего узла, а вознаграждение является просто временем, потраченным сумкой на перемещение, то эта формула вырождается в формулу, используемую алгоритмом Q-routing (1.2.3). Безусловно, на практике, как и в оригинальном алгоритме Q-routing, сама сумка не является агентом, вычисляющим собственную стра-

тегию. Вместо этого на место каждой очередной сумки себя ставит контроллер манипулятора.

Как уже было замечено, в общем случае множество наблюдений Ω может быть очень большим или даже бесконечным, что обуславливает необходимость аппроксимации функции $Q(o, a)$. Для этого можно использовать нейросети. В следующей главе будут рассмотрены примеры применения нейросетей в задачах обучения с подкреплением

1.4. Обзор методов обучения нейросетей с подкреплением

Активные исследования в области обучения с подкреплением с использованием нейросетей начались с публикации командой DeepMind алгоритма DQN [17]. Алгоритм показал способность эффективно обучаться игре в классические видеоигры на эмуляторе Atari 2600.

Алгоритм DQN базируется на методе Q-обучения. В качестве состояния нейросеть получает на вход текущее изображение игрового экрана. Набор действий соответствует возможному набору игровых действий.

Ключевой проблемой при использовании Q-обучения с нейросетями является то, что метод Q-обучения в применении к марковским процессам с бесконечным числом состояний, вообще говоря, не сходится к оптимальной стратегии. Алгоритм DQN борется с этим обстоятельством с помощью *experience replay* — буфера из всех встреченных четверок (s, a, r, s'), где s — начальное состояние, a — действие, предпринятое в этом состоянии, r — полученное вознаграждение, s' — следующее состояние. Этот буфер работает как «память» нейросети: в каждый момент времени из буфера выбирается случайное подмножество встреченных ситуаций, и нейросеть заново обучается на них, «вспоминая» игровой опыт. Кроме этого, для стабилизации процесса обучения в алгоритме DQN использовалась *целевая нейросеть* (target network) — дополнительная нейросеть, предоставляющая опорные оценки для основной сети и копирующая ее состояние раз в k шагов.

В дальнейшем было изобретено множество модификаций оригинального алгоритма DQN. Из них можно выделить модификацию с использованием приоритизированного experience replay [28]. Идея приоритизированного experience replay заключается в том, чтобы с большей вероятностью доставать из буфера эпизоды, сильнее всего повлиявшие на процесс обучения — т. е. те, где нейросеть ошибалась в своих оценках Q-функции сильнее всего. Также

были разработаны такие модификации алгоритма, как Double DQN [33], основанный на идее двойного Q-обучения [14] и Dueling DQN [12], основанный на идее оценки Q-функции как отдельных величин — ценности состояния $V(s)$ и преимущества действия $A(a)$.

Также в работе [15] было показано, что добавление LSTM-слоя [**hochreiter1997long**] в архитектуру нейросети позволяет добиться лучших результатов при оптимизации частично наблюдаемых марковских процессов благодаря сохранению информации о предыдущих наблюдениях в скрытом состоянии рекуррентного слоя. В дальнейшем глубокие нейросети с рекуррентной архитектурой показывали впечатляющие результаты в решении сложных задач, таких как игра Doom [20].

Одним из первых исследований по применению глубоких нейросетей в мультиагентной среде является исследование [27], в котором две нейросети играли в игру Pong друг с другом. Но уже в работе [21] показано, что рекуррентные нейросети могут научиться пересылать друг другу сообщения для решения совместных задач (в частности, в этом исследовании нейросети учились разрабатывать совместную стратегию поведения для решения головоломки об узнике и лампочке). На основе этой работы было проведено исследование [18], в котором одна нейросеть обучалась «задавать вопросы» другой, чтобы по полученным ответам угадать, какое из изображений «загадала» другая нейросеть.

Выводы по главе 1

В главе 1 были рассмотрены существующие подходы к управлению конвейерными системами и схожими системами, существующие алгоритмы маршрутизации, их преимущества и недостатки. Задача управления конвейерной системой была сведена к задаче маршрутизации и сформулирована в терминах обучения с подкреплением; был намечен подход к ее решению. Также был проведен обзор современных методов глубокого обучения с подкреплением, в том числе в мультиагентной среде.

ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННОГО АЛГОРИТМА

2.1. DQN-routing

В бакалаврской выпускной квалификационной работе, выполненной автором данной работы ранее, а также в статье [26], был предложен алгоритм *DQN-routing*, идея которого заключается в объединении метода Q-routing (1.2.3) с обучением нейросетей. Эта секция посвящена краткому обзору на него.

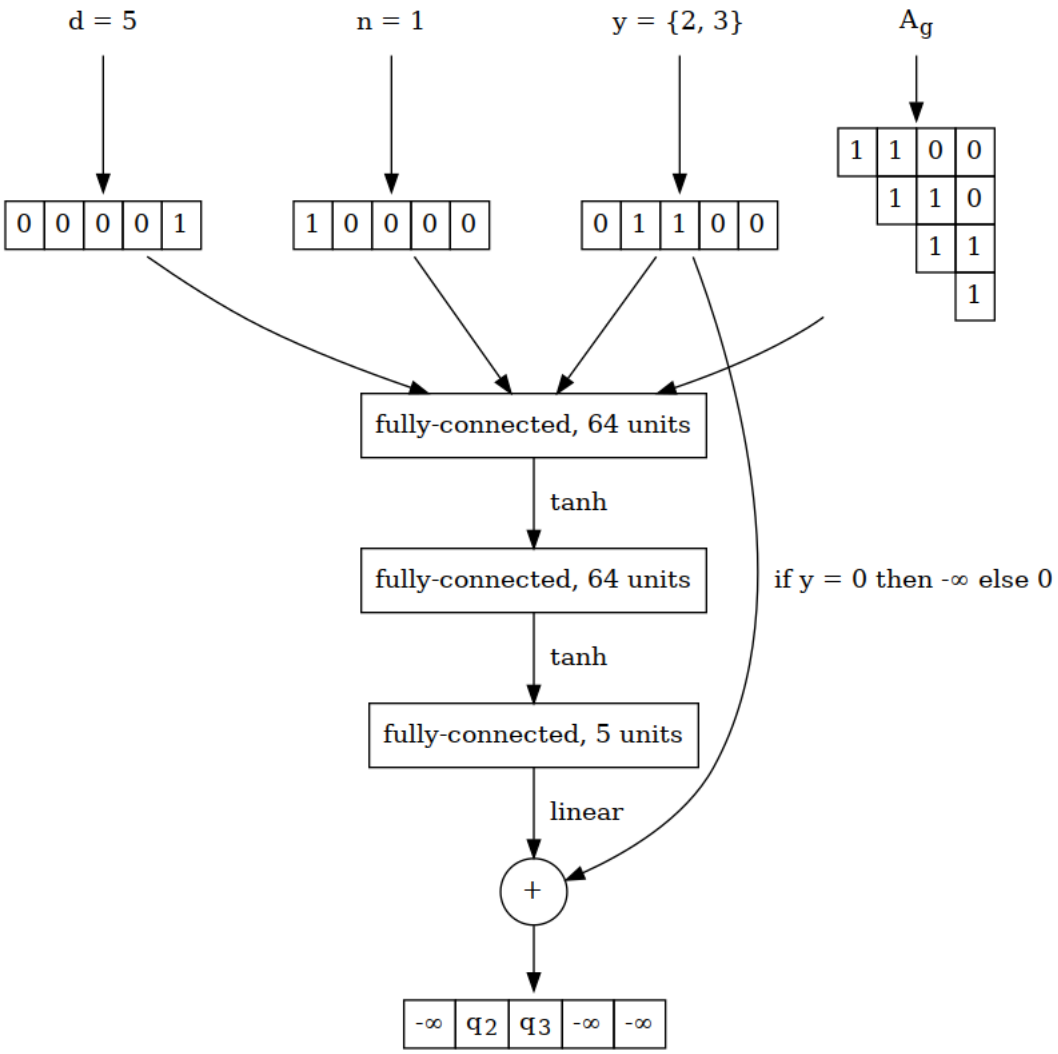


Рисунок 2 – DQN-routing: архитектура сети для графа из 5 узлов

Базовой частью любого текущего состояния s (или, точнее *наблюдения* $o \in \Omega$) будем считать кортеж $(n, d, y_1 ... y_m)$, где n — это идентификатор (номер) текущего узла, d — номер узла назначения текущего пакета, а $y_1 ... y_m$ — номера соседей текущего узла. Добавление номера текущего узла позволит одной и той же нейросети работать в качестве роутера в любом из узлов графа.

Также на вход нейросети подается матрица смежности графа, разложенная в вектор. Информацию о текущей топологии сети агент получает при помощи алгоритма link-state.

Чтобы избежать взаимозависимости оценок Q-функции для узлов с близкими номерами, будем использовать *унитарный код*, т. е. для сети с семью узлами номер 3 будет кодироваться как 0010000, номер 5 — как 0000100, и т. д. Для кодирования множества соседей будем использовать тот же принцип, т. е. множество соседей 1, 3, 4 будет закодировано как 1011000. Таким образом, размер базового входного состояния для нейросети составит $3n$, где n — количество узлов в сети.

Выходной слой нейронной сети состоит из n нейронов с линейными функциями активации, где i-ый нейрон соответствует i-ому узлу в сети. К выходам узлов, не являющихся соседями текущего, отдельно прибавляется $-\infty$ (на практике — большое отрицательное число, например -1000000). Таким образом, на выходе нейросети образуются оценки функции $Q(s, a)$ для всех действий a, где для действий, недоступных в текущем состоянии (узлов, не являющихся соседями), $Q(s, a) = -\infty$.

2.2. DQN-LE-routing

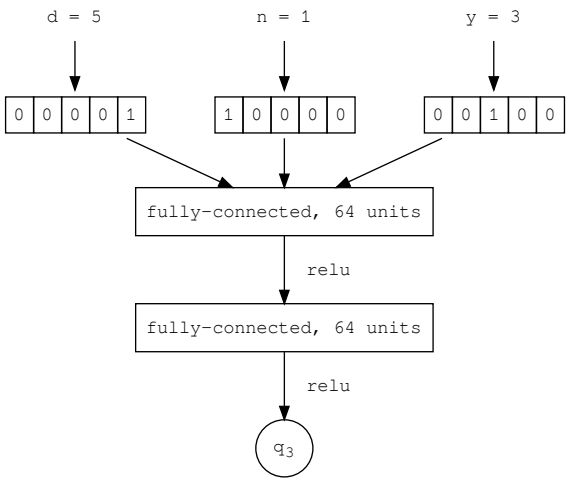


Рисунок 3 – DQN-LE-routing: архитектура сети

TBD

Выводы по главе 2

TBD

ГЛАВА 3. ЭКСПЕРИМЕНТЫ

Экспериментальное сравнение разработанных алгоритмов проводилось в двух имитационных моделях: модели абстрактной компьютерной сети и модели конвейерной системы. Эксперименты в модели абстрактной компьютерной сети использовались для того, чтобы проверять работоспособность базовой структуры рассматриваемого алгоритма в относительно простом сеттинге. Кроме того, симуляция компьютерной сети является менее вычислительно интенсивной задачей, чем симуляция конвейерной системы. Также, существуют хорошо изученные способы генерации случайных графов, обладающих топологическими свойствами, аналогичными таковым у реальных компьютерных сетей (например, модель Барабаши-Альберт [1]), в то время как случайная генерация правдоподобных моделей конвейерной сети является нетривиальной нерешенной задачей. Все это делает модель компьютерной сети хорошей площадкой для первичного анализа разработанного алгоритма.

Стоит, однако, подчеркнуть, что рассматривается модель *абстрактной* компьютерной сети. Отличие рассматриваемой модели от модели реальной компьютерной сети заключается в том, что служебные сообщения передаются между узлами мгновенно, в отличие от целевых пакетов, которые испытывают задержку при проходе по соединениям и в очередях обработки на узлах. Это сделано для того, чтобы приблизить модель абстрактной компьютерной сети к модели реальной конвейерной сети.

При проведении экспериментов в обоих моделях разработанные алгоритмы сравнивались со следующими бейзлайнами:

- **Shortest paths (SP)**: алгоритм кратчайших путей, использующий протокол link-state;
- **Q-routing**, описанный в разделе 1.2.3 и являющийся идейным предком предложенных алгоритмов DQN-routing и DQN-LE-routing.
- TBD: добавить бейзлайны

Каждый тестовый сценарий для каждого типа алгоритма был запущен трижды с различными числами инициализации генератора случайных чисел (были использованы числа 42, 43 и 44). Целевые значения усреднялись на временных отрезках в 500 единиц времени модели. На представленных графиках линиями представлены значения, усредненные по трем запускам сценария; со-

ответствующими линиям полупрозрачными полосами изображены разбросы между минимальным и максимальным значениями по трём запускам.

3.1. Эксперименты в модели абстрактной компьютерной сети

Нейросетевые агенты для алгоритмов DQN-routing и DQN-LE-routing перед работой проходили предварительное обучение с учителем на эпизодах работы алгоритма кратчайших путей внутри графа из 10 вершин (Рис. 4). Каждое соединение в графе имеет задержку 10 мс и пропускную способность 1024 байт/мс. Каждый пакет имеет размер 1024 байт, каждый роутер обрабатывает один пакет за 5 мс. Алгоритм кратчайших путей в качестве веса ребра использует его задержку.

Всего было сгенерировано 230000 эпизодов, включая эпизоды работы на версиях данного графа с некоторыми отсутствующими ребрами. Для DQN-LE агента использовались графовые эмбеддинги размерности 8. Оба нейросетевых агента обучались на данных в течение 10 эпох с помощью алгоритма RMSProp [32].

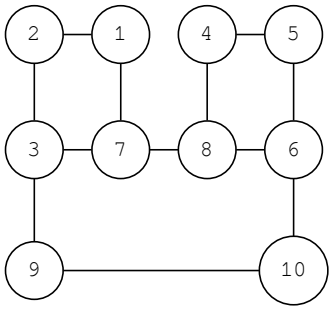


Рисунок 4 – Базовый граф для тестов в модели компьютерной сети

3.1.1. Адаптация к изменениям интенсивности трафика

В этом сценарии пакеты перемещаются между двумя половинами графа 4 (между множествами вершин $\{1, 2, 3, 7\}$ и $\{4, 5, 6, 8\}$). Сначала пакеты посылаются раз в 10 мс, затем нагрузка резко возрастает — пакеты посылаются с периодом 3.5 мс —, и в конце падает вновь.

В условиях повышения нагрузки на сеть и перегрузки «популярных» роутеров, через которые проходит большинство кратчайших путей (узлы 7 и 8 в

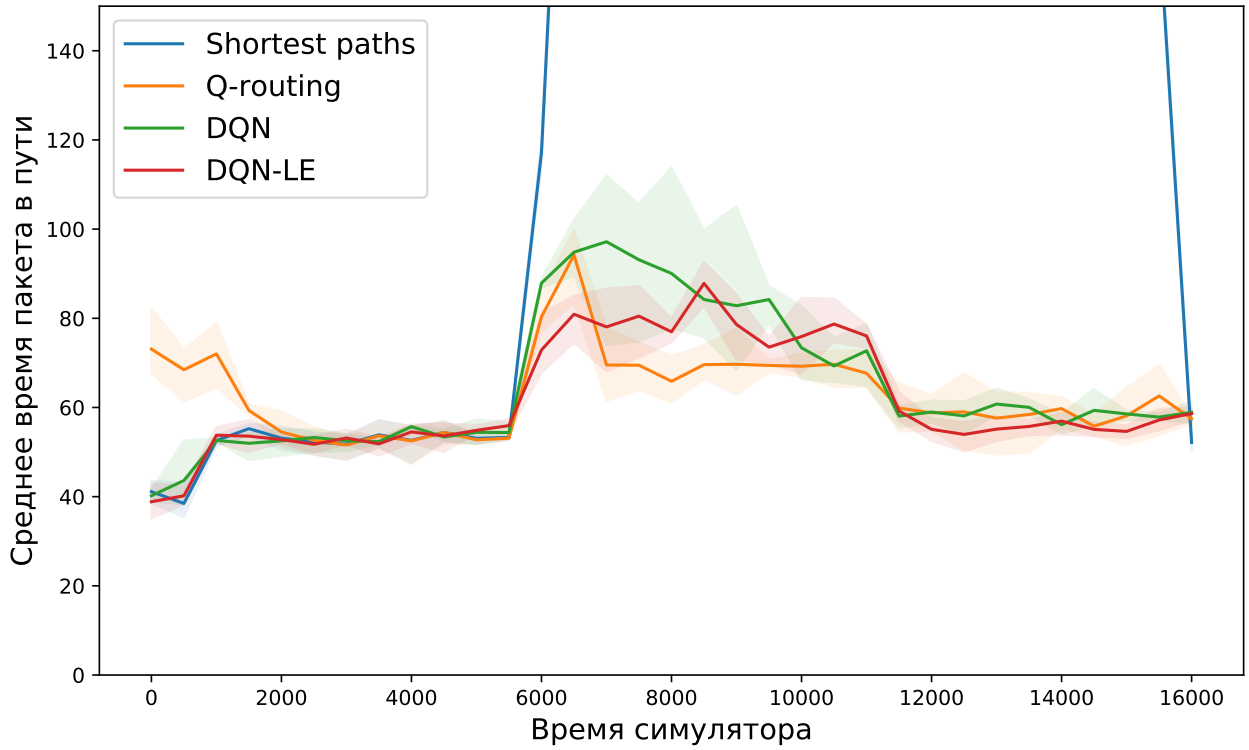


Рисунок 5 – Резкое понижение и последующее снижение нагрузки

графе 4) алгоритм shortest paths не способен работать эффективно, как видно на графике 5 — очереди этих роутеров начинают стремительно расти, а с ними и среднее время доставки пакетов. Все прочие алгоритмы оказываются способны быстро обнаружить затор и повести трафик обходным путём.

3.1.2. Изменение топологии сети

В этом сценарии при небольшой нагрузке в сети производился последовательный обрыв соединений $(7, 8)$, $(1, 2)$ и $(5, 6)$ и последующее их восстановление в том же порядке. Пакеты посылались, как и в предыдущем сценарии, между двумя половинами графа 4.

Как видно на графике 6, алгоритмы DQN- и DQN-LE-routing в этом сценарии работают почти в точности так же, как и оптимальный в этой ситуации алгоритм кратчайших путей, в то время как Q-routing не способен вернуться к оптимальному поведению после восстановления всех соединений.

3.1.3. Перенос опыта на новые топологии графов

Необходимость предобучения для работы в графе заданной топологии является существенным недостатком алгоритма DQN-routing, сильно ограничивающим его практическую ценность. Алгоритм DQN-LE-routing тоже тре-

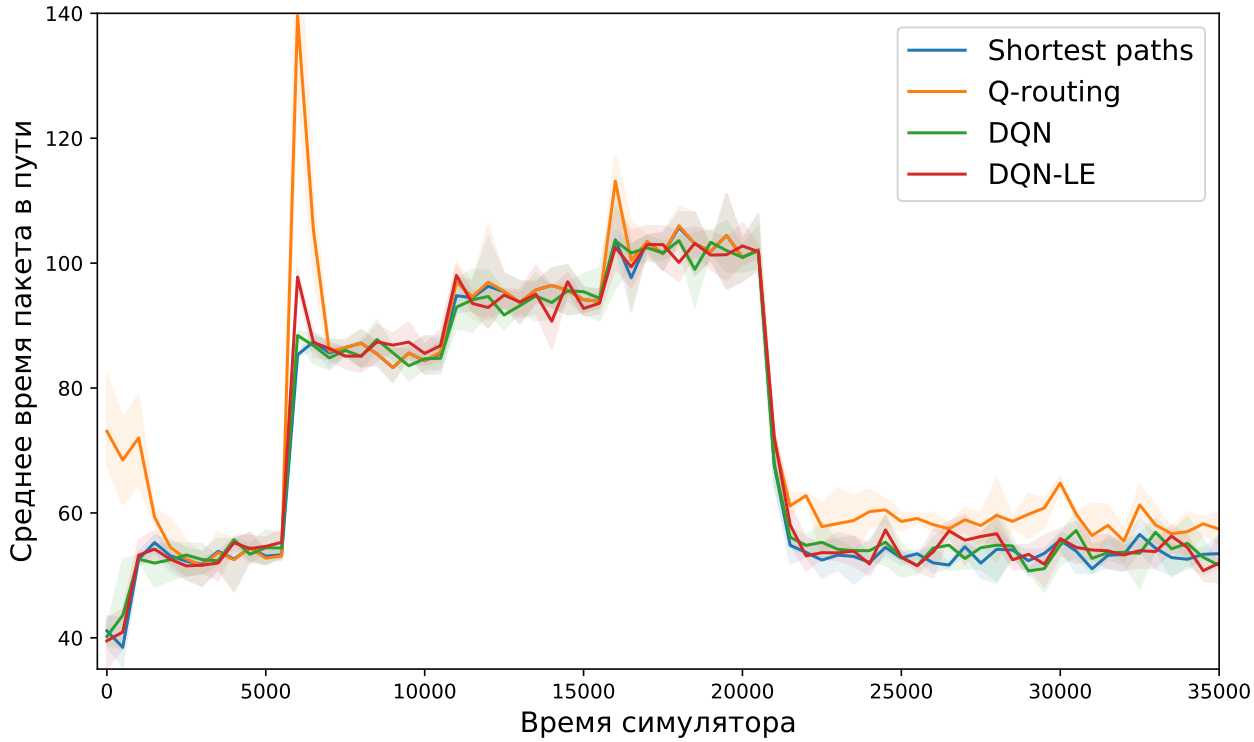


Рисунок 6 – Обрыв и последующее восстановление трех соединений

бует предобучения, но благодаря использованию графовых эмбеддингов обобщающая способность единожды обученной модели гораздо выше.

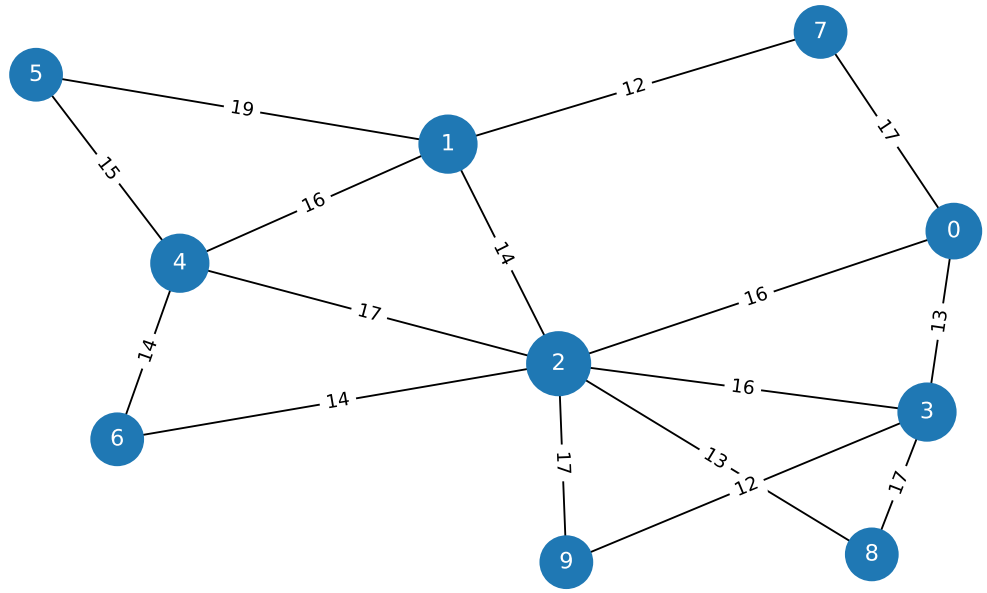


Рисунок 7 – Случайный граф из 10 вершин

Нейросетевые агенты для алгоритмов DQN- и DQN-LE-routing, обученные на данных с базового графа из 10 вершин (рис. 4), были использованы для работы в случайном графе из 10 вершин, сгенерированном по модели

Барабаши-Альберт с параметром $m = 2$ и случайными значениями задержки соединений от 10 до 20 мс. Прочие параметры сгенерированного графа идентичны таковым в базовом графе.

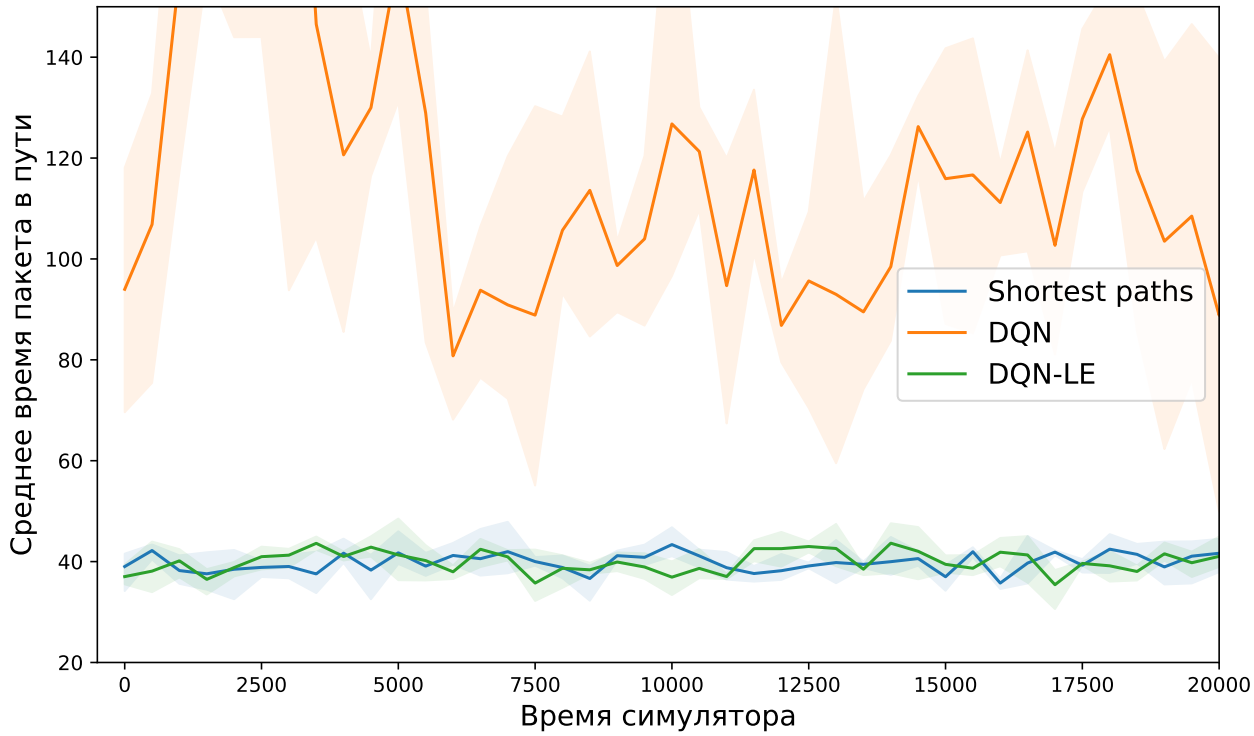


Рисунок 8 – Перенос опыта на новую топологию графа

График 8 демонстрирует показатели работы алгоритмов в новом графе при низкой нагрузке. Работа алгоритма кратчайших путей (оптимального при низкой нагрузке) также приведена для сравнения. Видно, что DQN-routing ведет себя в новом графе хаотично и даже не способен сойтись к какой-либо стабильной стратегии. В отличие от него, DQN-LE-routing работает оптимально (аналогично алгоритму кратчайших путей) с самого начала, что позволяет сделать вывод, что опыт нейросети, полученый при обучении, очень хорошо обобщается на графы того же размера.

Нейросетевой агент алгоритма DQN-routing, обученный на графе с N вершинами, принципиально не может работать на графах большего размера, однако DQN-LE-routing свободен от этого ограничения, так как векторные представления заданной размерности d можно получить для вершин любого графа. На графиках 10a и 10b изображена работа алгоритма DQN-LE-routing, предобученного на базовом графе из 10 вершин, внутри случайного графа из 40 вершин (рис. 9), в условиях низкой и высокой нагрузки. В условиях низкой нагрузки DQN-LE-routing начинает показывать приемлемое качество работы

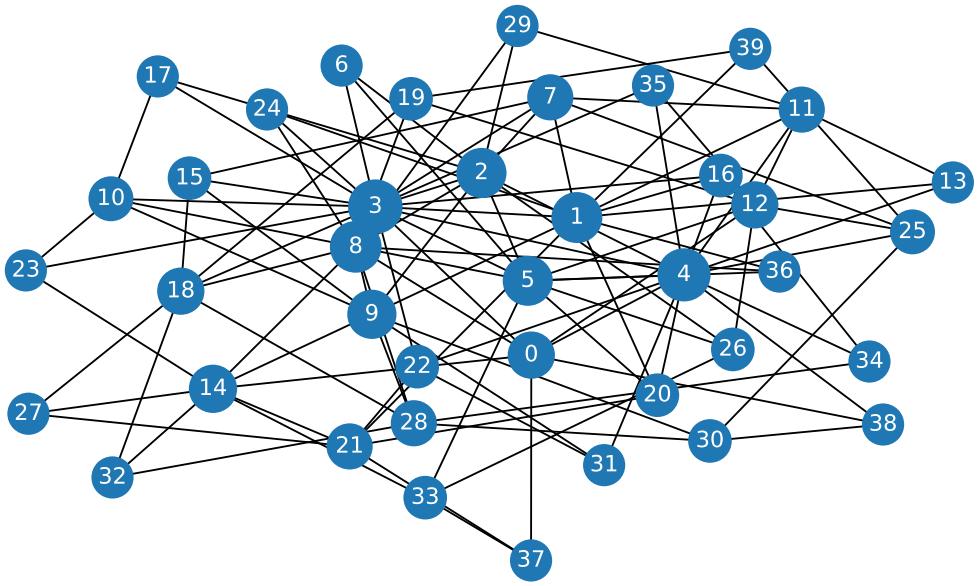
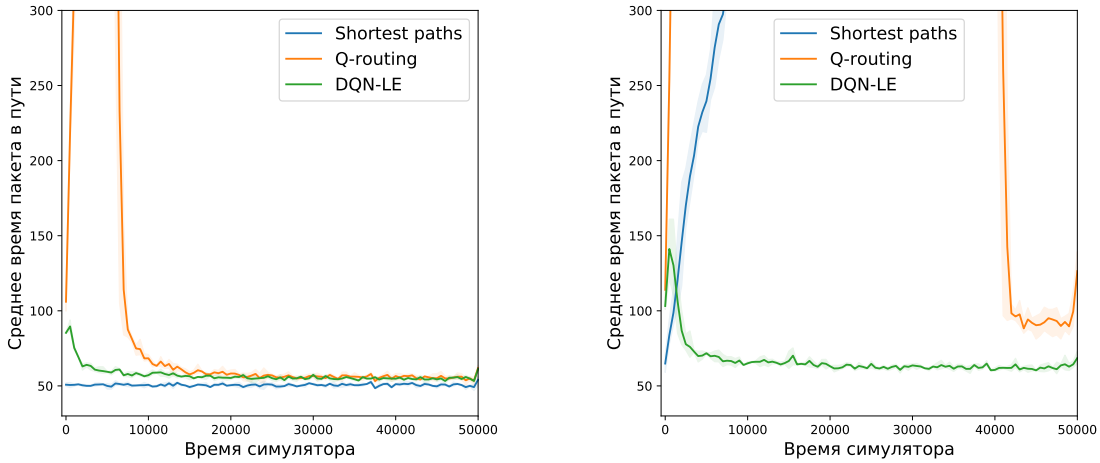


Рисунок 9 – Случайный граф из 40 вершин



(a) Низкая нагрузка (b) Высокая нагрузка

Рисунок 10 – Перенос опыта на новую топологию большего размера

существенно быстрее, чем Q-routing, хоть и не сходится к оптимальной стратегии аналогичной алгоритму кратчайших путей — но к ней не может сойтись и Q-routing. В условиях высокой нагрузки DQN-LE-routing ведет себя существенно лучше обоих бейзлайнов — shortest path переполняет очереди ключевых узлов и расходится, Q-routing долгое время ведет себя хаотично, прежде чем сойтись к очень неоптимальной стратегии, в то время как DQN-LE-routing работает как часы.

Из результатов экспериментов можно сделать вывод, что предобученный DQN-LE-routing обладает хорошей обобщающей способностью и способен успешно работать на большом множестве различных графов.

3.2. Эксперименты в модели конвейерной системы

В отличие от модели компьютерной сети, в модели конвейерной системы алгоритм будет пытаться оптимизировать не только среднее время доставки пакета (чемодана), но и *энергопотребление* конвейеров. Оптимизация энергопотребления заключается в том, чтобы использовать для доставки чемоданов до точек назначения как можно меньше конвейеров.

В алгоритме DQN-routing на вход нейросети будет дополнительно подаваться вектор w размера n (где n — количество конвейерных секций в системе), в котором на i-ой позиции стоит 0, если конвейер, на котором находится секция i, в данный момент находится в состоянии ожидания, и 1 — если он работает.

В алгоритме DQN-LE-routing при расчете Q-функции для соседней секции x на вход нейросети будет дополнительно подаваться число 1, если конвейер, на котором находится секция x, работает, и 0 — если не работает.

Эксперименты проводились на модели конвейерной сети из 14 конвейеров с 27 секциями в сумме, двумя входными вершинами и четырьмя выходными вершинами (рис. 11). Секции с первой по 20 имеют длину 10 метров, с 21 по 27 — 2 метра. Энергопотребление каждого конвейера равно 1 кВт, максимальная скорость — 1 м/с.

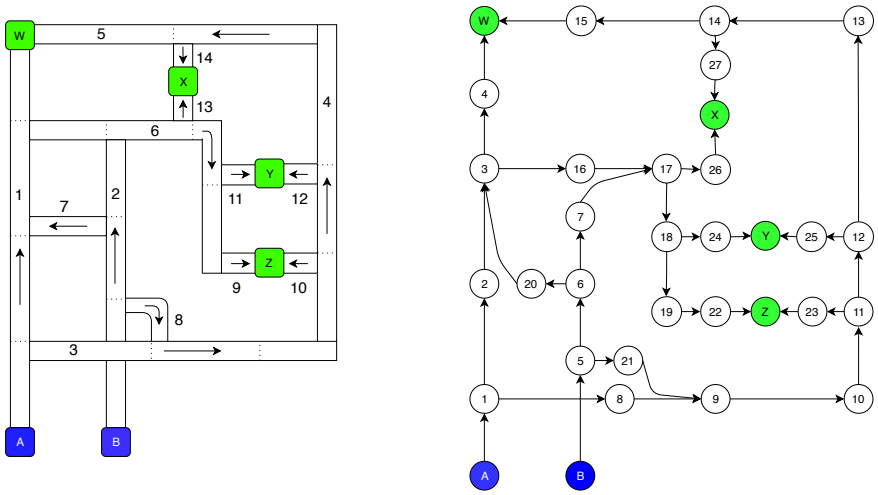
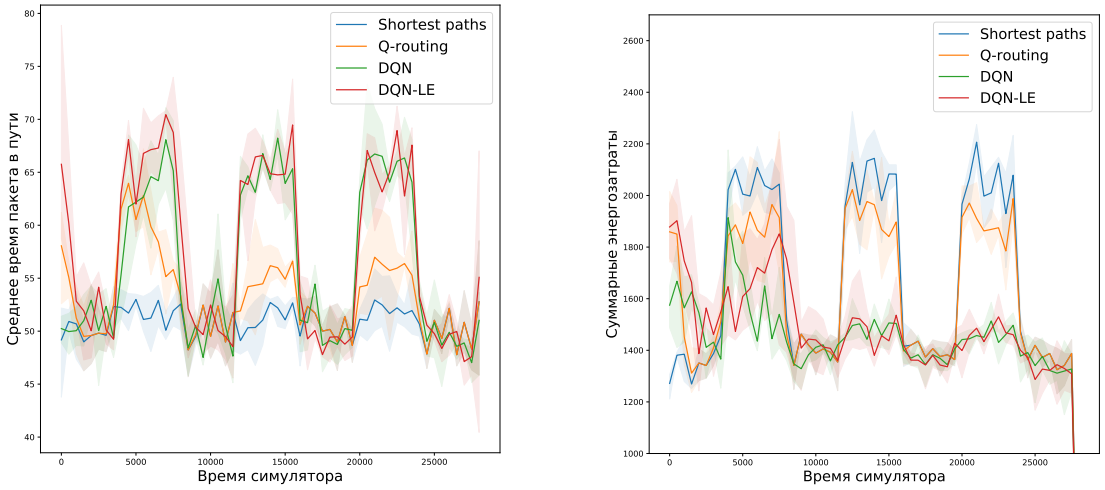


Рисунок 11 – Модель конвейерной системы для тестов

3.2.1. Неравномерный поток до выходных вершин

Как можно видеть на иллюстрации 11, кратчайший путь от входов до выходов W и X пролегает через конвейеры 1, 2, 7 и 6, а до выходов Y и Z – через конвейеры 1, 2, 3 и 4. Однако до выходов Y и Z также можно попасть через конвейер 6, не задействуя конвейеры 3 и 4. Если для нас важно оптимизировать энергозатраты, то в условиях небольшого входящего потока чемоданов такая стратегия является более предпочтительной.



(a) Время сумок в пути (b) Энергопотребление системы

Рисунок 12 – Неравномерный поток до выходных вершин

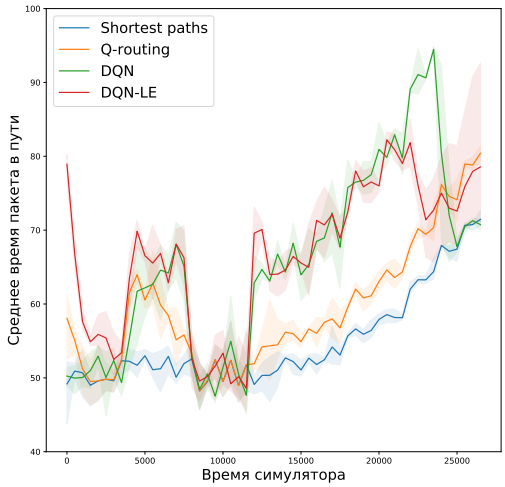
В данном сценарии поток чемоданов попеременно идет либо только в выходы W и X, либо во все четыре выхода сразу. Параметр важности экономии энергопотребления α равен 1.

На иллюстрации 12 видно, что алгоритмы DQN- и DQN-LE-routing после непродолжительного периода адаптации начинает работать заметно лучше алгоритмов shortest paths и Q-routing с точки зрения экономии энергии, жертвуя средним временем доставки чемодана.

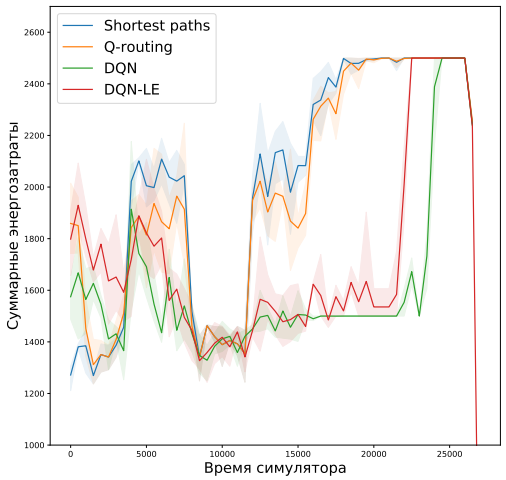
3.2.2. Плавное повышение нагрузки

Первая половина этого сценария повторяет предыдущий — поток попеременно идет то во все выходы, то только в выходы W и X. Во второй половине сценария частота появления чемоданов на входах возрастает в два раза и продолжает плавно расти.

В этом сценарии алгоритмы DQN- и DQN-LE-routing используют разные параметры важности экономии энергопотребления, обоначенные соотв. α



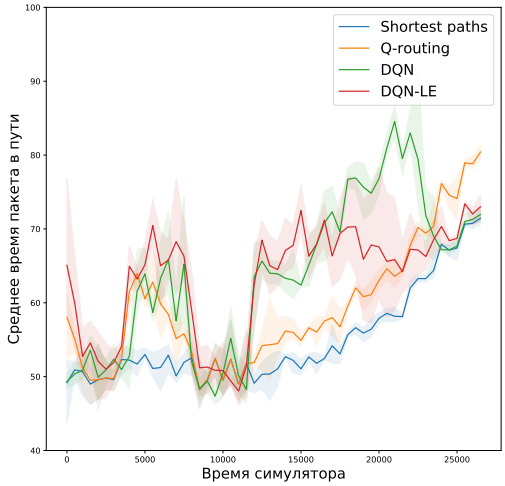
(a) Время сумок в пути



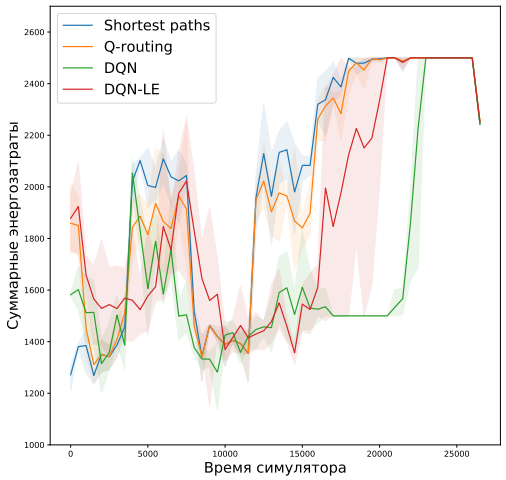
(b) Энергопотребление системы

Рисунок 13 – Плавное повышение нагрузки: $\alpha = 1$, $\alpha_{LE} = 0.4$

и α_{LE}, так как на практике алгоритм DQN-LE-routing оказался более чувствительным к величине этого параметра.



(a) Время сумок в пути



(b) Энергопотребление системы

Рисунок 14 – Плавное повышение нагрузки: $\alpha = 0.8$, $\alpha_{LE} = 0.2$

Графики и показывают, что при постепенном увеличении интенсивности трафика (и, как следствие, загруженности конвейеров) алгоритмы DQN- и DQN-routing в какой-то момент начинают использовать конвейеры 3 и 4, чтобы снизить среднюю скорость доставки чемоданов, причем это происходит тем раньше, чем ниже параметр α (α_{LE}).

Выводы по главе 3

TBD

ЗАКЛЮЧЕНИЕ

TBD

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Albert R., Barabási A.-L.* Statistical mechanics of complex networks // Reviews of modern physics. — 2002. — Т. 74, № 1. — С. 47.

2 *Ali M. K. M., Kamoun F.* Neural networks for shortest path computation and routing in computer networks // IEEE Transactions on Neural Networks. — 1993. — Vol. 4, no. 6. — P. 941–953.

3 *Araujo F., Ribeiro B., Rodrigues L.* A neural network for shortest path computation // IEEE Transactions on Neural Networks. — 2001. — Vol. 12, no. 5. — P. 1067–1073.

4 *Bellman R.* On a routing problem // Quarterly of Applied Mathematics. — 1958. — No. 16. — P. 87–90.

5 *Bosack L.* Method and apparatus for routing communications among computer networks. — 2 11/1992. — URL: `https://www.google.com/patents/US5088032` ; US Patent 5,088,032.

6 *Boyan J. A., Littman M. L.* Packet routing in dynamically changing networks: a reinforcement learning approach // Advances in Neural Information Processing Systems. — 1994. — No. 6. — P. 671–678.

7 *Cataldo A., Scattolini R.* Dynamic pallet routing in a manufacturing transport line with model predictive control // IEEE Transactions on control systems technology. — 2016. — Vol. 24, no. 5. — P. 1812–1819.

8 *Choi S. P. . M., Yeung D.-Y.* Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control // Advances in Neural Information Processing Systems. — 1996. — No. 8. — P. 945–951.

9 *De Neufville R.* The baggage system at Denver: prospects and lessons // Journal of Air Transport Management. — 1994. — Vol. 1, no. 4. — P. 229–236.

10 *Di Caro G., Dorigo M.* AntNet: Distributed stigmergetic control for communications networks // Journal of Artificial Intelligence Research. — 1998. — Vol. 9. — P. 317–365.

11 *Dijkstra E. W.* A note on two problems in connexion with graphs // Numerische Mathematik. — 1959. — No. 1. — P. 269–271.

12 Dueling network architectures for deep reinforcement learning / Z. Wang [et al.] // arXiv preprint arXiv:1511.06581. — 2015.

13 *Eaton J. W., Rawlings J. B.* Model-predictive control of chemical processes // Chemical Engineering Science. — 1992. — Vol. 47, no. 4. — P. 705–720.

14 *Hasselt H. V.* Double Q-learning // Advances in Neural Information Processing Systems. — 2010. — P. 2613–2621.

15 *Hausknecht M., Stone P.* Deep recurrent q-learning for partially observable mdps // arXiv preprint arXiv:1507.06527. — 2015.

16 *Hendrick C.* Routing Information Protocol : RFC. — 06/1988. — No. 1058.

17 Human-level control through deep reinforcement learning / V. Mnih [et al.] // Nature. — 2015. — No. 518. — P. 529–533.

18 *Jorge E., Kågebäck M., Gustavsson E.* Learning to Play Guess Who? and Inventing a Grounded Language as a Consequence // arXiv preprint arXiv:1611.03218. — 2016.

19 *Kumar S., Miikkulainen R.* Dual reinforcement Q-routing: An on-line adaptive routing algorithm // Artificial Neural Networks in Engineering. — 1997. — No. 7. — P. 231–238.

20 *Lample G., Chaplot D. S.* Playing FPS games with deep reinforcement learning // arXiv preprint arXiv:1609.05521. — 2016.

21 Learning to communicate with deep multi-agent reinforcement learning / J. Foerster [et al.] // Advances in Neural Information Processing Systems. — 2016. — P. 2137–2145.

22 *Luo J., Huang W., Zhang S.* Energy cost optimal operation of belt conveyors using model predictive control methodology // Journal of Cleaner Production. — 2015. — Vol. 105. — P. 196–205.

23 *McQuillan J. M., Richer I., Rosen E. C.* The New Routing Algorithm for the ARPANet // IEEE Trans. on Comm. — 1980. — Vol. 28, no. 5. — P. 711–719.

24 *McQuillan J. M., Walden D. C.* The ARPA network design decisions // Computer Networks. — 1977. — Vol. 1, no. 5. — P. 243–289.

25 *Moy J.* OSPF Version 2 : RFC. — 04/1998. — No. 1058.

26 Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system / D. Mukhutdinov [et al.] // Future Generation Computer Systems. — 2019. — Vol. 94. — P. 587–600.

27 Multiagent cooperation and competition with deep reinforcement learning / A. Tampuu [et al.] // PloS one. — 2017. — Vol. 12, no. 4. — e0172395.

28 Prioritized experience replay / T. Schaul [et al.] // arXiv preprint arXiv:1511.05952. — 2015.

29 *Qin S. J., Badgwell T. A.* A survey of industrial model predictive control technology // Control engineering practice. — 2003. — Vol. 11, no. 7. — P. 733–764.

30 *Tan M.* Multi-agent reinforcement learning: Independent vs. cooperative agents // Proceedings of the tenth international conference on machine learning. — 1993. — P. 330–337.

31 *Tarau A. N., De Schutter B., Hellendoorn H.* Model-based control for route choice in automated baggage handling systems // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). — 2010. — Vol. 40, no. 3. — P. 341–351.

32 *Tieleman T., Hinton G.* Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude // COURSERA: Neural networks for machine learning. — 2012. — Vol. 4, no. 2.

33 *Van Hasselt H., Guez A., Silver D.* Deep Reinforcement Learning with Double Q-Learning. // AAAI. — 2016. — P. 2094–2100.

34 *Watking C.* Learning from Delayed Rewards : PhD thesis / Watking C. — Cambridge : King's College, 1989.

35 *Yan J., Vyatkin V.* Distributed Software Architecture Enabling Peer to Peer Communicating Controllers // IEEE Transactions on Industrial Informatics. — 2013. — Vol. 9, no. 4. — P. 2200–2209.

36 *Zeinaly Y., De Schutter B., Hellendoorn H.* An integrated model predictive scheme for baggage-handling systems: Routing, line balancing, and empty-cart management // IEEE Transactions on control Systems technology. — 2015. — Vol. 23, no. 4. — P. 1536–1545.

ПРИЛОЖЕНИЕ А. ПРОМЕЖУТОЧНЫЕ РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

TBD

ПРИЛОЖЕНИЕ Б. ОПИСАНИЕ РАЗРАБОТАННЫХ ИМИТАЦИОННЫХ МОДЕЛЕЙ

TBD