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 _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

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

УТВЕРЖДАЮ
Руководитель ОП
проф., д.т.н. Парфенов В.Г. _____
«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Мухутдинов Д.В. **Группа** М4239 **Факультет** ИТиП

Руководитель Фильченков А.А., к.ф-м.н., кафедра КТ

1 Наименование темы: Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы: «31» мая 2019 г.

3 Техническое задание и исходные данные к работе

Требуется разработать децентрализованный алгоритм управления конвейерной системы для транспортировки багажа. Алгоритм должен позволять контроллерам конвейерной сети динамически изменять свое поведение в целях адаптации под изменившиеся условия работы, такие как поломка одного из конвейеров или изменение интенсивности потока багажа. Алгоритм должен обеспечивать своевременную доставку багажных единиц до пунктов назначения, в то же время минимизируя энергопотребление всей системы в целом.

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

Пояснительная записка должна содержать обзор существующих результатов в сфере управления конвейерными системами, а также в сферах, имеющих непосредственное отношение к предложенному алгоритму (таких как сетевая маршрутизация и обучение с подкреплением). Также записка должна содержать подробное изложение предложенного алгоритма и данные экспериментального сравнения его производительности с производительностью существующих методов управления конвейерной системой, проведенного с помощью виртуальной имитационной модели конвейерной сети.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

а) Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, 2012

б) Mnih et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

7 Дата выдачи задания «01» сентября 2017 г.

Руководитель ВКР _____

Задание принял к исполнению _____

«01» сентября 2017 г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

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

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент: Мухутдинов Дмитрий Вадимович

Наименование темы ВКР: Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением

Наименование организации, где выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработка децентрализованного алгоритма управления конвейерной системой, способного адаптироваться к гетерогенным изменениям в условиях работы.

2 Задачи, решаемые в ВКР:

а) разработка виртуальной имитационной модели конвейерной сети, позволяющей проводить сравнительный анализ алгоритмов управления;

б) реализация ряда известных алгоритмов управления конвейерной сетью (а также маршрутизации) в рамках разработанной имитационной модели;

в) разработка алгоритма, соответствующего поставленным требованиям;

г) экспериментальное сравнение разработанного алгоритма с существующими по качеству работы.

3 Число источников, использованных при составлении обзора: 44

4 Полное число источников, использованных в работе: 58

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
2	1	1	22	9	23

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	Раздел работы
Язык программирования Python 3.6	Гл. 2, 3
Библиотека для дискретно-событийного моделирования SimPy	Гл. 2, 3
Библиотека для операций с графами NetworkX	Гл. 2, 3
Математические пакеты NumPy и SciPy	Гл. 2, 3
Библиотеки для машинного обучения scikit-learn и PyTorch	Гл. 2, 3
Среда интерактивной разработки Jupyter Lab	Гл. 2, 3
Библиотеки Matplotlib, Seaborn и Pandas для обработки и визуализации данных	Гл. 2, 3

8 Краткая характеристика полученных результатов

Разработан алгоритм управления конвейерной системой на основе глубокого мультиагентного обучения с подкреплением. В ходе экспериментального исследования было установлено, что разработанный алгоритм превосходит существующие по качеству работы и способен адаптироваться к изменениям во внешней среде.

9 Гранты, полученные при выполнении работы

Государственное задание № 2.8866.2017/БЧ «Технология разработки программного обеспечения систем управления ответственными объектами на основе глубокого обучения и конечных автоматов»

10 Наличие публикаций и выступлений на конференциях по теме работы

1 *Д.В. М.* Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением // Сборник тезисов докладов конгресса молодых ученых. Электронное издание. — 2019.

2 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.

Студент Мухутдинов Д.В. _____

Руководитель Фильченков А.А. _____

«____» _____ 20___ г.

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 [14]. 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) [46]. Управление с прогнозируемыми моделями изначально разрабатывалось для задач химической промышленности и нефтепереработки [18], 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 [12, 35] и системами распределения багажа на основе рельсовых тележек (*англ.* destination coded vehicles, DCVs) [50, 57]. 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, задачу управления конвейерной системой можно декомпозировать на несколько подзадач, основными из которых являются задачи маршрутизации объектов и предотвращения их столкновения. Первая из этих подзадач 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».

Наиболее часто задача пакетной маршрутизации встречается в компьютерных сетях, и для них разработано большое количество алгоритмов. Преимуществами алгоритмов сетевой маршрутизации являются обусловленные характером задачи децентрализованность, низкая требовательность к вычислительным ресурсам и устойчивость к отказам маршрутизаторов и обрывам соединений. В работе [56] 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 обучения с подкреплением (*англ.* reinforcement learing, RL). Примерами алгоритмов на основе это-

го подхода является Q-routing [10] и его модификации [13, 32]. Q-routing производит обновление стратегии поведения после каждого *действия* — пересылки очередного пакета. Это, с одной стороны, обуславливает его способность к быстрой адаптации к изменениям нагрузки в сети. Но, с другой стороны, это же вынуждает маршрутизаторы пересылать большое количество служебных сообщений, что само по себе повышает нагрузку на сеть. Вследствие этого, применение Q-routing и других подобных алгоритмов редко бывает целесообразным в реальных компьютерных сетях.

В контексте конвейерных систем, 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.

Однако, кроме двух обозначенных подзадач, важной подзадачей управления конвейерной системой также является снижение ее энергопотребления. Промышленные конвейерные системы потребляют до 6% всей электроэнергии в мире. Тем не менее, на данный момент не существует децентрализованного алгоритма управления конвейерами, который бы учитывал в том числе и энергопотребление. Целью данной работы является разработка такого децентрализованного алгоритма.

В данной работе будет предложен децентрализованный алгоритм управления конвейерной системой, базирующийся на идеях Q-routing и глубокого обучения с подкреплением. 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 будет рассмотрена сама задача управления штучной конвейерной системой. Она будет разбита на подзадачи, такие как задача маршрутизации, задача избежания столкновений и задача оптимизации энергопотребления. Для каждой из этих задач будут рассмотрены известные подходы к их решению, включая подходы из смежных областей, таких как сетевая маршрутизация и глубокое обучение с подкреплением. Также будет рассмотрен и про-

анализирован предложенный ранее автором алгоритм маршрутизации DQN-routing, основанный на тех же идеях, будут изучены его ключевые недостатки.

В главе 2 будет описана рассматриваемая архитектура конвейерной системы и описан порядок базового взаимодействия агентов внутри нее. Задача управления конвейерной системой будет сформулирована как задача мультиагентного обучения с подкреплением, и будет предложен алгоритм управления конвейерной системой, опирающийся на эту формулировку.

В главе 3 будут приведены экспериментальные результаты работы алгоритма в моделях абстрактной компьютерной сети и конвейерной системы. Будут эмпирически обоснованы решения, принятые в ходе разработки алгоритма. Будет продемонстрирована применимость алгоритма к решению общей задачи маршрутизации, а также отсутствие недостатков, характерных для DQN-routing. Также будет показано, что алгоритм способен эффективно решать задачу управления конвейерной системой с учетом энергопотребления.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Задача управления конвейерной системой

Конвейерные системы имеют широкий спектр применения во многих различных областях человеческой деятельности. Соответственно, существует множество различных видов конвейеров и конвейерных систем. С инженерной точки зрения конвейерные системы можно различать по принципу действия и по конструктивным характеристикам — их можно подразделить на ленчатые, пластинчатые, тележные и прочие. Также конвееры различают по назначению, то есть по типу перемещаемых грузов и по сфере применения.

С точки зрения математического моделирования конвейерной системы наиболее осмысленно разделять конвейеры на два крупных множества – для перемещения штучных грузов и для перемещения насыпных (или кусковых) грузов. В первом случае перемещаемые по системе грузы можно моделировать как множество дискретных объектов, где каждый объект имеет свою определенную позицию, во втором случае так делать нельзя. Также эти типы конвейеров решают разные задачи, и соответсвенно, требуют различных подходов к оптимизации своей работы.

В данной работе будет рассмотрена задача управления конвейерной системой для перемещения штучных грузов. В качестве конкретной рассматриваемой задачи выбрана задача управления системой ленчатых (пластинчатых) конвейеров для транспортировки багажа в аэропорту. Для проведения экспериментов была разработана программная имитационная модель конвейерной системы (см. раздел 2.1)

При перемещении штучных грузов с помощью конвейерной системы очевидным критерием оптимизации является скорость доставки груза до точки назначения. Для систем с нелинейной топологией (какими, например, являются системы для транспортировки багажа), кроме того, необходимо решать задачу избежания столкновений между перемещаемыми объектами, и, более того, эта задача является приоритетной. И для всех без исключения конвейерных систем, включая и системы для перемещения насыпных грузов, актуальна задача оптимизации энергопотребления.

Далее мы рассмотрим эти три подзадачи и подходы к их решению по отдельности.

1.1.1. Маршрутизация

Задача оптимизации скорости доставки груза до точки назначения с помощью конвейерной системы по большей части сводится к задаче маршрутизации. В общем случае, задача маршрутизации — это задача поиска множества путей наименьшей стоимости $P = \{(p^i, p^i_1, ..., p^i_k, d^i)\}^N_{i=0}$ в графе $G = (V, E)$ при заданном множестве пар точек входа и точек назначения $Q = \{(p^i, d^i)\}^N_{i=0}$ и некоторых заданных ограничениях на пути. Как правило, за стоимость пути берется время его прохождения некоторым объектом, находящемся в заданной точке входа — например, время доставки пакета по адресу в компьютерных сетях.

Большинство исследований, посвященных маршрутизации, посвящены исследованию маршрутизации именно в компьютерных сетях, что неудивительно, учитывая ее колоссальную практическую значимость. Наиболее важные в рамках данной работы алгоритмы сетевой маршрутизации будут рассмотрены в разделе 1.2. Сейчас мы рассмотрим существующие подходы к маршрутизации в конвейерных системах, в особенности — в системах транспортировки багажа.

1.1.1.1. Маршрутизация для насыпных конвейеров

В конвейерных системах для перемещения насыпных грузов задача маршрутизации возникает, например, в тех случаях, когда нужно сортировать по разным точкам назначения несколько типов разных материалов, поступающих из одного источника. Такая задача часто возникает в рамках работы промышленных предприятий, например, металлургических заводов.

Существуют методы оптимального решения этой задачи, основанные на ее представлении в виде задачи линейного программирования [5]. Эти методы предполагают, что все запросы за некоторый временной интервал планирования T заранее известны, и подразумевают централизованный расчет стратегии работы для всей системы. К тому же, в силу существенно различных специфик работы между насыпными и штучными конвейерами, нельзя сходу применить эти методы к задаче маршрутизации в штучных конвейерных системах.

1.1.1.2. Маршрутизация для штучных конвейеров

Задача маршрутизации в конвейерных системах для перемещения штучных грузов очень похожа на задачу маршрутизации в компьютерных сетях,

если представить конвейерную сеть в виде направленного графа, где узлами графа являются точки назначения (входы, выходы и какие-либо промежуточные точки на пути грузов), а ребрами являются секции конвейеров между сочленениями. Так как методы маршрутизации в компьютерных сетях хорошо изучены и развиваются очень давно, имеет смысл применять их также и к конвейерным сетям.

Среди конвейерных систем, перемещающих штучные грузы, одними из наиболее распространенных являются системы для транспортировки багажа. Они предназначены для доставки багажа со стоек регистрации, где его оставляют пассажиры, к разгрузочным портам терминалов вылета, а также от терминалов прибытия к местам выдачи багажа.

В современных крупных аэропортах, где багаж необходимо перемещать на большие расстояния, системы для транспортировки багажа часто состоят не только из конвейеров, но также и из автоматических рельсовых тележек (*англ.* destination coded vehicles, DCVs) [19, 50]. В таких случаях конвейеры перемещают багаж только до тележек и обратно. Однако, чисто конвейерные системы транспортировки дешевле и проще в установке, поэтому они остаются востребованными, особенно для небольших аэропортов [53].

Рисунок 1 – Отклонитель отклоняет сумку на смежный конвейер

Ключевыми элементами конвейерной системы, помимо собственно конвейеров, являются сканеры багажной бирки (*англ.* scanner) и отклонители (*англ.* diverter) (рис. 1). Последние играют роль агентов, выполняющих маршрутизацию сумок в системе.

Отличительной чертой этих систем является невозможность применения к ним статических методов оптимального контроля наподобие метода маршрутизации для насыпных конвейеров, описанного в разделе 1.1.1.1. Это связано с тем, что все запросы маршрутизации за некоторый период планирования T не могут быть известны заранее, так как пассажиры регистрируют багаж на разные рейсы в случайном порядке, и информация о наличии конкретной сумки появляется в системе только после прохождения сумкой сканера [14, 22]. Таким образом, стратегия маршрутизации должна быть рассчитана для каждой новой сумки в отдельности.

В существующих конвейерных системах по большей части применяется статический подход к маршрутизации, когда для каждой новой сумки весь маршрут до конечной точки назначения рассчитывается в момент ее сканирования [31]. В некоторых системах применяется более динамический подход, аналогичный маршрутизации в конвейерных сетях, когда кратчайший маршрут до точки назначения рассчитывается при приближении сумки к очередному отклонителю (*англ.* diverter) [22]. Однако, в обоих случаях стоимость пути рассчитывается только с учетом длин конвейеров, например, с помощью алгоритма Дейкстры. Второй подход в этом случае имеет больше смысла, чем первый, только тогда, когда сумка может быть доставлена в любую из нескольких точек назначения (например, в случае, если сумка должна быть доставлена к одному из множества рентгеновских сканеров для проверки службой безопасности), и когда в ходе работы системы какие-либо из этих точек могут стать недоступны (например, когда выбранный ранее сканер занят другой сумкой).

В большинстве случаев, в системах для транспортировки багажа контроллеры отклонителей на сочленениях не производят расчет стратегии, а только посылают сообщения об обнаружении подъезжающей сумки центральному логическому узлу, который, в свою очередь, посылает обратно сообщение с действием, которое требуется совершить (сместить сумку/не смещать) [22, 31]. Это еще одно существенное отличие от маршрутизации в компьютерных сетях, так как централизованная маршрутизация в компьютерных сетях довольно редко имеет смысл (как правило, только в особых случаях, имеющих отношение к программно-определенным сетям (*англ.* software-defined networks, SDNs)) [36, 47].

Однако, централизованный подход к управлению всегда подразумевает наличие центральной точки отказа для всей системы — собственно логического контроллера. Очевидно, что это нежелательное свойство для киберфизических систем, поэтому децентрализованные подходы к управлению являются более предпочтительными. В следующем подразделе мы подробно рассмотрим существующий такой подход.

1.1.1.3. Децентрализованная маршрутизация в системе транспортировки багажа

Децентрализованный метод управления конвейерной системой для транспортировки багажа был предложен в работах [3, 56]. В этих работах описана архитектура и логика контроллеров отдельных элементов системы для транспортировки багажа (конвейеры, сканеры, отклонители и т. д.) а также протокол их взаимодействия на основе стандарта IEC 61499 [28]. Предложенная в них архитектура обладает следующими свойствами:

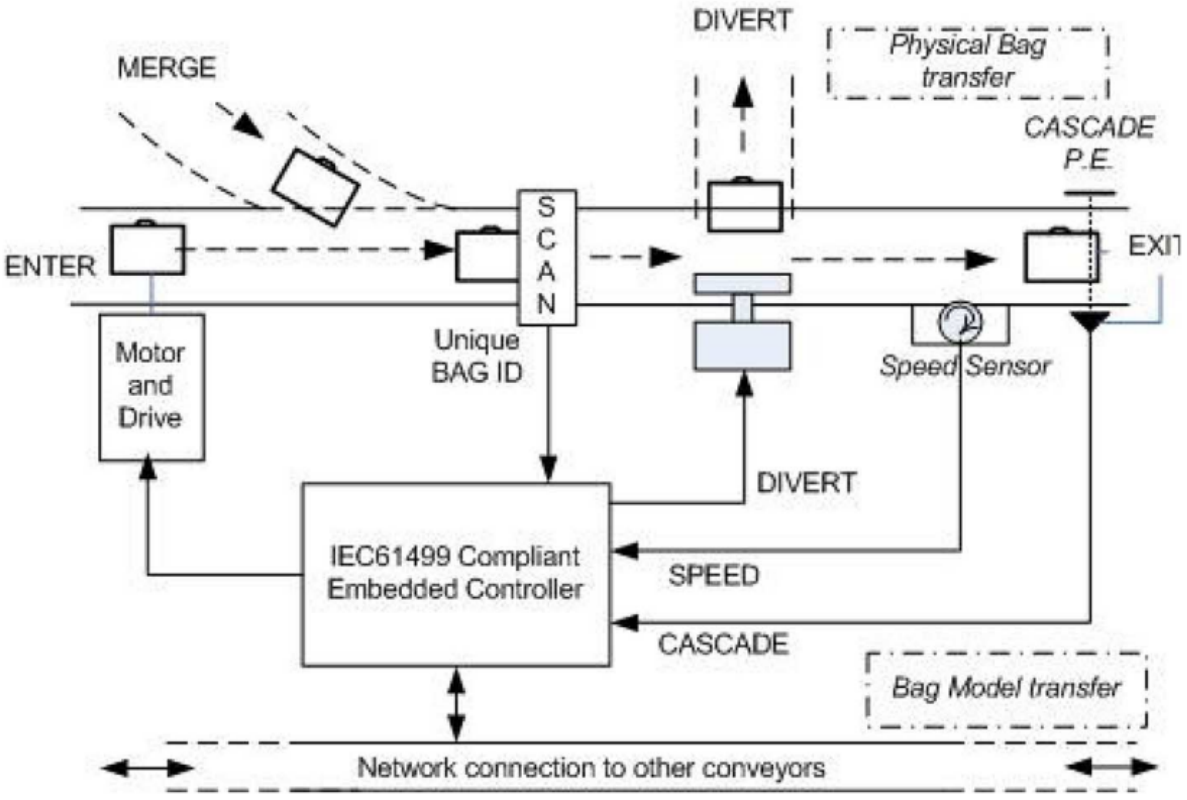


Рисунок 2 – Схема децентрализованной архитектуры конвейера

— Каждый отдельный конвейер имеет свой логический контроллер
— На конвейере в различных позициях могут находиться:

  - сочленения с входящими конвейерами (*англ.* mergers)
  - сочленения с исходящими конвейерами, оборудованные отклонителями(*англ.* diverters)
- Перед каждым сочленением, как входящим, так и исходящим, установлен детектор, определяющий позицию сумки.
- Сочлененные конвейеры соединены между собой коммуникационным интерфейсом и могут посылать друг другу сообщения.

На рисунке 2 представлена схема этой архитектуры.

Большая часть этих работ фокусируется на технических деталях реализации логики контроллеров в рамках упомянутого стандарта и изучением преимуществ его использования в сравнении с более старыми архитектурными стандартами для программных контроллеров в киберфизических системах (таких как IEC 61131-3 [27]).

С точки зрения непосредственно логики маршрутизации, в работах используется немного модифицированный распределенный алгоритм Беллмана-Форда [9], по сути, почти в точности повторяющий логику дистанционно-векторных алгоритмов сетевой маршрутизации (что будут рассмотрены более подробно в разделе 1.2.1). Работа [56] отдельно демонстрирует, что этот алгоритм способен адаптировать маршруты в случае поломок отдельных конвейеров, точно так же, как дистанционно-векторный алгоритм в компьютерных сетях способен адаптироваться к обрывам соединений. Это, вкупе с отсутствием единой точки отказа, выгодно выделяет предложенный децентрализованный подход на фоне общепринятых.

1.1.2. Избежание столкновений

Важнейшим ограничением на работу конвейерной системы любого предназначения является недопущение столкновения перемещаемых объектов. Это ограничение применимо и к конвейерным системам для насыпных грузов, в тех случаях, когда по системе перемещаются грузы разного типа, смешение которых недопустимо.

Для систем, в которых для маршрутизации применяется расчет оптимальной стратегии действий для заданного множества объектов в заданный промежуток времени, как в работе [5], ограничение на запрет столкновений объектов (или потоков материалов) можно выразить как одно из ограничений в задаче линейного/квадратичного программирования (или же как один из чле-

нов оптимизируемой функции с большим коэффициентом), чтобы выполненная глобальная стратегия, будучи правильно исполненной, гарантированно не привела к столкновениям.

Однако, как уже было сказано, такой подход неприменим для решения задачи управления системой транспортировки багажа. Вследствие этого в них, впрочем, как и в других транспортных системах, применяются другие подходы.

Подход, основанный на взаимной координации скоростей соединяющихся конвейеров, описан во многих источниках [3, 48]. Описания в них, в сущности, отличаются лишь глубиной и характером технических деталей. Суть подхода во всех случаях одна и заключается во взаимной синхронизации скоростей соединяющихся конвейеров при приближении объектов на них к сочленению.

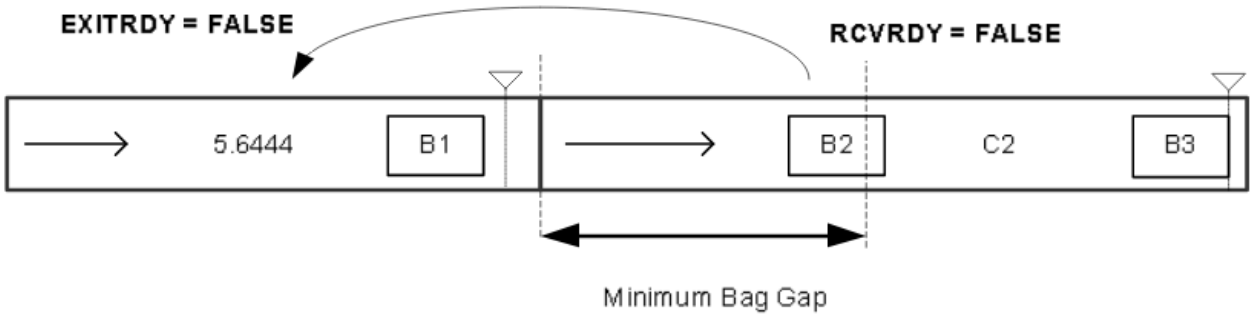


Рисунок 3 – Схема взаимодействия соединяющихся конвейеров при приближении сумки к сочленению

Для конкретики рассмотрим децентрализованный подход, описанный в [3]. Как было указано в предыдущем разделе, в нем подразумевается, что каждый конвейер имеет собственный контроллер. Каждый такой контроллер поддерживает модель положения сумок на соответствующем конвейере. Такая модель поддерживается путем получения сигналов от детекторов, стоящих в конце каждой секции конвейера, а также путем передачи сообщений о входящих/исходящих сумках между соседними конвейерами.

В задачу системы входит поддержание установленного минимального зазора D между сумками. Если начало конвейера $C2$ находится, согласно текущей модели положения сумок, на расстоянии меньшем D от некоторой сумки, то въезд другой сумки на начало конвейера запрещен. Конвейер сообщает об этом путем посылки специального сигнала RCVRDY = FALSE предыдуще-

му конвейеру $C1$, который, в свою очередь, выставляет соответствущий флаг EXITRDY = FALSE. Аналогичным образом работают сигналы MERGERDY и DIVERTRDY в случаях соединения конвейеров в сочленениях.

Если при приближении сумки в выходу или отклонителю соответствующий флаг EXITRDY/DIVERTRDY имеет значение FALSE, то конвейер останавливается, не допуская, чтобы сумка достигла конца текущей секции до тех пор, пока значение флага не станет TRUE. Кроме того, конвейер выставляет все свой флаг RCVRDY и все флаги MERGERDY, сообщая предыдущим конвейерам, что он останавливается и что на него не следует перемещать сумки. При приближении сумок к такому конвейеру предыдущие конвейеры будут также останавливаться, не допуская перемещение сумок на него до тех пор, пока конвейер не возобновит работу. Таким образом достигается, что между сумками, достигшими одного и того же сочленения, соблюдается разумное расстояние, что исключает их столкновение.

В данной работе для избежания столкновений сумок будет использован именно этот метод с небольшими модификациями в сочетании с разработанным алгоритмом маршрутизации.

1.1.3. Оптимизация энергопотребления

Оптимизация энергопотребления конвейерных систем является крайне важной в глобальном смысле задачей, так как на работу промышленных конвейерных систем уходит до 6% всей потребленной электроэнергии в мире [15].

Один из подходов к оптимизации энергопотребления — усовершенствование физического оборудования: разработка новых типов лент, смазок и натяжных колес конвейеров в целях снижения побочных затрат энергии на трение и шум [49]. Еще один подход, не отрицающий первого — снижение энергопотребления путем контроля поведения конвейера.

Простейшим подходом к энергосбережению в конвейерных системах является метод оптимального выключения (*англ.* optimal switching control) — выключение конвейера в то время, когда он не выполняет полезной работы. Этот подход, вообще говоря, применим не только к конвейерам, но и вообще к любому автоматизированному оборудованию. Так, в работе [40] предложен метод оптимизации энергопотребления разгрузочных конвейеров угольной шахты, основанный на методе оптимального контроля, который определяет оптимальные временные промежутки для включения и выключения конвейеров. Там же

приведена формулировка этой задачи в обобщенной форме как задачи минимизации функции

$$J = \int_{t_0}^{t_1} \sum_{i=1}^{n} P_i(t) u_i(t) p(t) \mathrm{d}t \,, \tag{1}$$

где $[t_0, t_1]$ — временной интервал планирования, n — количество конвейеров, $P_i(t)$ — энергопотребление i-го конвейера в момент времени t, $p(t)$ — функция, которая задает стоимость электроэнергии во время t (дана как условие задачи), и

$$u_i(t) = \begin{cases} 1, & \text{if switched on} \\ 0, & \text{if switched off} \end{cases} \tag{2}$$

— контролируемая величина, определяющая, включен или нет конвейер i в момент времени t. Минимизация функции J производится в соответствии с некоторыми ограничениями вида

$$g(u(t), t) \leqslant 0 \,, \tag{3}$$

которые выражают операционные требования к системе (конвейеры должны переместить груз, то есть они должны быть включены на момент поступления груза и в то время, когда он находится на конвейере).

В случае системы транспортировки багажа, как уже было сказано, расписание прибытия сумок не может быть известно заранее, что не позволяет использовать данный подход к оптимизации энергопотребления. Однако, никто не запрещает использовать простые эвристики, такие как остановка и выключение конвейера спустя некоторое время после того, как на нем не осталось сумок [42].

В случае, если включенный конвейер потребляет константное количество электроэнергии за единицу времени, метод оптимального переключения является наилучшим. Однако, на практике это не так. На текущую энергоэффективность конвейера в каждый конкретный момент времени влияет множество факторов: КПД двигателя, трение ленты, масса перемещаемого груза, скорость движения и так далее. Зависимость энергопотребления от этих параметров можно выразить с помощью математической модели.

В целях проектирования и производства эффективного конвейерного оборудования были разработаны стандартизированные модели расчета планируемого энергопотребления, такие как ISO 5048 [29], DIN 22101 [17] и JIS B 9905 [30]. Эти модели предназначены для конвейеров, перемещающих насыпные грузы, и учитывают большое количество точных параметров конструкции механизма, такие как масса вращающихся деталей, плотность конвейерной ленты и так далее.

Как правило, ввиду своей сложности, данные модели не используются для моделирования энергопотребления системы при решении задачи оптимизации. Для оптимизации энергопотребления в работе [58] была предложена приближенная аналитическая модель. Эта модель также предназначена для конвейеров, перемещающих насыпные грузы, и выражает текущее энергопотребление конвейера как:

$$P(V,T) = \frac{1}{\eta}\left(\theta_1 V T^2 + \theta_2 V + \theta_3 \frac{T^2}{V} + \theta_4 T + \frac{V^2 T}{3.6}\right), \tag{4}$$

где V — это текущая скорость конвейера, T — масса поступающего на вход конвейера за единицу времени вещества, η — КПД системы, а θ_1, θ_2, θ_3 и θ_4 — коэффициенты, рассчитываемые из постоянных характеристик конвейера, используемых в модели ISO 5048.

В работе [15] приводится простая обобщенная модель энергопотребления конвейера для транспортировки штучных грузов:

$$P(V) = \frac{M_r(m_L)V}{\eta}, \tag{5}$$

где V — это текущая скорость конвейера, M_r — это величина полного сопротивления движению, зависящая от суммарной массы находящихся на конвейере объектов m_L и константных характеристик, таких как длина конвейера, его ширина, и прочих.

Видно, что в этой модели энергопотребление линейно зависит от скорости конвейера (если не учитывать тот факт, что от скорости конвейера зависит скорость изменения m_L). В модели (4) тоже содержится линейный член от V, что побуждает прийти к выводу, что в правдоподобных моделях энергопотребления конвейера зависимость энергопотребления от скорости похожа на линейную. Таким образом, если конвейер способен произвольно изменять

свою скорость, в целях снижения энергопотребления также имеет смысл регулировать скорость конвейера, а не просто по возможности его выключать.

В разработанной в рамках данной работы имитационной модели конвейерной системы для подсчета “реальных” затрат на электроэнергию используется модифицированная под нужды штучных конвейеров модель (4).

1.2. Алгоритмы сетевой маршрутизации

Подавляющее большинство существующих алгоритмов маршрутизации разработаны именно для применения в компьютерных сетях. Среди большого множества известных алгоритмов маршрутизации в компьютерных сетях можно выделить два класса наиболее широко распространенных алгоритмов — дистанционно-векторные (*англ.* distance-vector) [39] и состояния каналов связи (*англ.* link-state) [38]. К каждому из этих классов относится множество различных алгоритмов, различающихся техническими деталями, но работающих по одному принципу. Далее мы рассмотрим только общие принципы работы алгоритмов из этих классов, опуская детали реализации отдельных их представителей.

1.2.1. Дистанционно-векторные алгоритмы

Листинг 1 – Процесс обновления вектора кратчайших расстояний. x – текущий узел, y — сосед, приславший собственный вектор расстояний D_y, w — вес ребра (x, y), V — множество всех узлов в сети.

```
1: procedure UpdateDV(x, y, w, D_y)
2:     for d ← V do
3:         (l_x, n_x) ← D_x(d)
4:         (l_y, n_y) ← D_y(d)
5:         if l_y + w < l_x or n_x = y then
6:             D_x(d) ← (l_y + w, n_y)
7:         end if
8:     end for
9: end procedure
```

В дистанционно-векторных алгоритмах каждый маршрутизатор поддерживает *вектор кратчайших расстояний* до всех остальных узлов в сети. Кроме собственно кратчайших расстояний, в этом векторе содержатся метки первых соседей текущего узла на соответствующих кратчайших путях. Расстояние до недостижимых узлов полагается равным бесконечности, соответствующий сосед для таких узлов не определен. Перенаправление пакета с узлом

назначения d, таким образом, осуществляется к соседу, хранящемуся для узла d в векторе кратчайших расстояний.

При обнаружении обрыва соединения (или, в общем случае, изменения веса смежного соединения) маршрутизатор, обновив соответствующим образом собственный вектор расстояний, рассылает его соседям, которые соответственно обновляют собственные вектора (листинг 1) и рассылают их далее.

Дистанционно-векторные алгоритмы просты в реализации и крайне нетребовательны к памяти и вычислительным ресурсам, но медленно распространяют информацию об изменениях по сети и сталкиваются с проблемами в при изменении топологии сети в определенных случаях (проблема count-to-infinity). Алгоритмы из рассмотренного в следующем подразделе класса избавлены от этих недостатков.

1.2.2. Алгоритмы состояния канала связи

В алгоритмах состояния канала связи каждый маршрутизатор поддерживает модель всей сети в виде графа. При обрыве или восстановлении смежного ребра, а также при изменении его веса маршрутизатор рассылает по сети обновленные состояния собственных смежных ребер. Логика распространения информации по сети приведена в листинге 2.

Таким образом, при обработке очередного пакета маршрутизатор посылает в него вдоль кратчайшего пути до узла назначения, который находится по текущей модели графа, например, с помощью алгоритма Дейкстры [16].

Алгоритмы из класса link-state обеспечивают более высокую скорость распространения информации по сети и, соответственно, более быструю адаптацию к обрывам и восстановлениям соединений. На текущий момент именно алгоритмы из этого класса (например, OSPF [41]) являются наиболее часто используемыми в реальных компьютерных сетях. Однако, большинство таких алгоритмов использует статические или редко обновляемые оценки весов соединений, что приводит к плохой производительности в условиях высоконагруженного трафика для некоторых топологий сетей.

1.2.3. Q-routing

Алгоритм Q-routing [10] является первым алгоритмом, в описании которого задача маршрутизации интерпретируется как задача обучения с подкреплением. Каждый маршрутизатор x поддерживает таблицу $Q_x(d, y)$, содержащую оценки стоимостей кратчайших путей до вершины d через каждого из

Листинг 2 – Логика link-state маршрутизатора. В каждой процедуре, x – метка текущего узла, $G_x = (V_x, E_x)$ – текущий граф сети, $W_x(u, v)$ – вес ребра (u, v).

```
1:  procedure InitLS(x, N_x)                    ▷ N_x – список соседей вершины x
2:      V_x ← {x}
3:      E_x ← ∅
4:      for (y, w) ← N_x do
5:          S_x(y) ← 0
6:      end for
7:      UpdateLS(x, x, 1, N_x)
8:  end procedure

9:  procedure AnnounceLS(x)
10:     N_x ← {(y, W_x(x, y))|(x, y) ∈ E_x}
11:     for (x, y) ← E_x do
12:         UpdateLS(y, x, S(x), N_x)
13:     end for
14:     S_x(x) ← S_x(x) + 1
15: end procedure

16: procedure UpdateLS(x, y, s_y, N_y)
17:     if s_y > S_x(y) then
18:         for (z, w) ← N_y do                 ▷ w – вес ребра до соседа
19:             V_x ← V_x ∪ {z}
20:             E_x ← E_x ∪ {(y, z)}
21:             W_x(y, z) ← w
22:         end for
23:         E_x ← E_x \ {(y, z)|z ∉ {z|(z, w) ∈ N_y}}
24:         S_x(y) ← s_y
25:         for (x, z) ← E_x do
26:             UpdateLS(z, y, s_y, N_y)
27:         end for
28:     end if
29: end procedure
```

соседей y. Пересылка пакета одному из соседей интерпретируется как *действие*, за которое маршрутизатор получает *вознаграждение* (равное времени, потраченное пакетом на преодоление исходящего ребра) и оценку стоимости дальнейшего пути от соседа (см. листинг 3).

Частое обновление оценок стоимостей путей в алгоритме Q-routing приводит к быстрой адаптации под изменяющиеся условия, такие как изменения

Листинг 3 – Логика алгоритма Q-routing. x – метка текущего узла, p – ID пакета.

```
1: function Route(x, y, p, d) ▷ d – узел назначения, y – узел, пославший пакет
2:     z ← argmin_{u∈neighbours of x} Q_x(d, u)
3:     t_Q ← Q_x(d, z)
4:     T_x(p) ← Time()                                  ▷ Текущее время
5:     ReceiveReward(y, x, p, T_x(p), t_Q)   ▷ Посылаем уведомление узлу y
6:     return z                                  ▷ Возвращаем узел назначения
7: end function

8: procedure ReceiveReward(x, y, p, t_y, t_Q)
9:     r_y ← t_y − T_x(p)
10:    ΔQ ← r_y + t_Q − Q_x(d, y)
11:    Q_x(d, y) ← ηΔQ                                  ▷ η – learning rate
12: end procedure
```

нагрузки или топологии графа. Однако, оно же обуславливает большое количество используемых алгоритмом служебных сообщений, которые в компьютерных сетях передаются по тем же каналам связи, что и служебные сообщения, что снижает среднюю пропускную способность этих каналов связи.

Однако при решении задачи маршрутизации физических объектов (таких как сумок в системе транспортировки багажа) время передачи служебного (электронного) сообщения пренебрежимо мало по сравнению со временем транспортировки физического объекта, и загруженность каналов связи не имеет ничего общего с загруженностью путей транспортировки объектов. Это побуждает к тому, чтобы рассмотреть возможности применения алгоритмов, подобных Q-routing, к таким задачам.

1.3. Применение обучения с подкреплением к поставленной задаче

В большинстве случаев при разработке процессов управления киберфизическими системами используются устоявшиеся подходы из областей теории операций и теории управления. Для управления простыми процессами используются PID-контроллеры[6], для сложных процессов разрабатываются математические модели, которые позволяют свести задачу управления к задаче оптимизации некоторой целевой функции, которую можно решать известными математическими методами.

Принцип *управления с прогнозируемыми моделями* (*англ.* model predictive control, MPC) как раз заключается в решении такой оптимизаци-

онной задачи на несколько шагов вперед, используя на каждом шагу после первого не реальные наблюдаемые параметры процесса, а предсказанные с помощью модели этого процесса. Этот принцип применяется как для задач управления промышленными конвейерными сетями [12, 35], так и для задач управления системами транспортировки багажа на основе DCV [50, 57] (но авторам не удалось обнаружить убедительных работ, применяющих MPC к задаче управления именно конвейерной системой транспортировки багажа).

Однако, этот метод имеет очевидные недостатки, такие как:

— необходимость разработки собственно модели процесса;
— вычислительная сложность, быстро растущая при повышении сложности модели и увеличении горизонта планирования.

Таким образом, его сложно применять в тех случаях, когда не получается придумать вычислительно эффективную модель процесса или когда количество параметров процесса слишком велико.

В отличие от MPC, методы *обучения с подкреплением* не требуют построения предсказательных моделей, а вместо этого корректируют стратегию поведения, основываясь на опыте взаимодействия с реальным процессом. В частности, подход глубокого обучения с подкреплением — с использованием нейросетей в качестве обучающихся агентов — демонстрирует ранее невиданные результаты в решении задач, ранее считавшихся нерешаемыми, таких как игра в Го [37]. Это мотивирует попытки применения методов глубокого обучения с подкреплением ко все более разнообразным задачам. В этой работе мультиагентное глубокое обучение с подкреплением будет применено к задаче управления конвейерной системой.

В следующих подразделах будут введены ключевые понятия из этой области и проведен краткий обзор существующих методов и применений глубокого обучения с подкреплением.

1.3.1. Термины и понятия

Определение 1. Обучение с подкреплением (*англ.* reinforcement learning, RL) — способ машинного обучения, при котором *агент* обучается, взаимодействуя с некоторой *средой* и получая за свои действия *вознаграждения*.

Формально, процесс взаимодействия агента и среды в обучении с подкреплением таков:

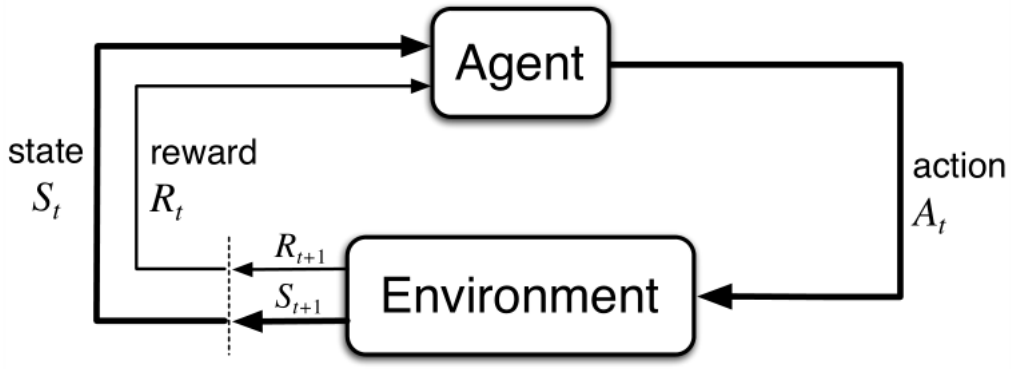


Рисунок 4 – Взаимодействие агента и среды

— Задаются начальные *стратегия агента* (*англ.* policy) $\pi_1(a|s)$ и *состояние среды* (*англ.* state) $s_1 \in \mathcal{S}$
— Для каждого момента времени $t = 1 \dots T$:
  – Агент выбирает *действие* $a_t \in \mathcal{A}$ согласно вероятностному распределению, заданному текущей стратегией: $a_t \; \pi_t(a|s_t)$;
  – Агент получает от среды вознаграждение $r_{t+1} \;\; p(r|a_t, s_t, r_t, a_{t-1}, s_{t-1}, \dots, a_1, s_1)$ и среда переходит в новое состояние $s_{t+1} \; p(s|a_t, s_t, r_t, a_{t-1}, s_{t-1}, \dots, a_1, s_1)$;
  – Агент корректирует стратегию $\pi_{t+1}(a|s)$.

Определение 2. Модель взаимодействия агента и среды называется **марковским процессом принятия решений** (*англ.* Markov decision process, MDP), если выполняется **марковское свойство**: вероятность получения вознаграждения r' и перехода в состояние s зависит только от текущего состояния и последнего действия:

$$P(r_{t+1} = r', s_{t+1} = s|a_t, s_t, r_t, \dots, a_1, s_1) = P(r_{t+1} = r', s_{t+1} = s|a_t, s_t) \quad (6)$$

В большинстве случаев задача обучения с подкреплением формулируется именно как марковский процесс принятия решений.

Определение 3. Стратегия агента $\pi(a|s)$ является **оптимальной**, если она максимизирует взвешенную сумму всех вознаграждений

$$\sum_{t=0}^{\infty} \gamma^t r_t \,, \quad (7)$$

где $0 \leqslant \gamma \leqslant 1$ — *коэффициент важности* для будущих выигрышей, для МППР без терминального состояния, или сумму

$$\sum_{t=0}^{N} r_t \tag{8}$$

для МППР с терминальным состоянием.

Во многих ситуациях, в число которых входят и случаи мультиагентных сред, агенту недоступно полное состояние среды s_t, а только какая-то его часть. Для таких случаев используется следующая модель:

Определение 4. Частично наблюдаемый марковский процесс принятия решений (*англ.* Partially observed Markov decision process, POMDP) — модель взаимодействия агента со средой, в котором вероятности вознаграждений и переходов между состояниями обладают марковским свойством, но стратегия агента $\pi(a|o)$ опирается на *наблюдение* $o_t \in \Omega$, а не на все состояние среды $s_t \in \mathcal{S}$. Наблюдение получается из состояния среды согласно заданному вероятностному распределению: $o_t\ p(o|s_t)$

Существуют различные методы нахождения оптимальной стратегии для заданного MDP или POMDP. В данной работе мы будем фокусироваться на методе *Q-обучения*.

Определение 5. Q-обучение (*англ.* Q-learning) [55] — метод нахождения оптимальной стратегии для MDP, заключающийся в построении функции ожидаемой полной стоимости $Q(s_{t'}, a_{t'}) \approx \sum\limits_{t=t'}^{\infty} \gamma^t r_t$. Текущее приближение $Q_{t+1}(s, a)$ обновляется при получении очередного вознаграждения r_{t+1} за действие a в состоянии s и переходе в состояние s' по формуле

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \left(r_{t+1} + \gamma \cdot \max_{a \in \mathcal{A}} - Q_t(s', a) \right) \tag{9}$$

Стратегия агента заключается в выборе действия с наибольшей ожидаемой полной стоимостью:

$$\pi_t(a|s) = \begin{cases} 1, & a = \text{argmax}_{a \in \mathcal{A}}\, Q_t(s, a) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

В случае MDP с конечными множествами действий и состояний Q-обучение гарантированно решает задачу обучения с подкреплением, т. е. $Q_t(s,a) \xrightarrow{t\to\infty} Q^*(s,a)$ и $Q^*(s,a)$ порождает оптимальную стратегию [55].

1.3.2. Обзор методов обучения нейросетей с подкреплением

Первым громким успехом в области глубокого обучения с подкреплением стал алгоритм Deep Q-Network (DQN) [26]. Он основан на методе Q-обучения, причем функция $Q(s,a)$ аппроксимируется нейронной сетью: на входной слой подается значение текущего состояния s, на выходном слое получаются значения $Q(s,a)$ для всех возможных $a \in \mathcal{A}$.

В оригинальной работе агент учился играть в видеоигры для платформы Atari 2600. Использовалась сверточная нейронная сеть, текущим состоянием s являлось изображение на экране и агент выбирал одно из возможных действий в игре.

Основной проблемой, с которой столкнулись авторы работы, стала неспособность Q-обучения в чистом виде сойтись к оптимальному решению — такая сходимость не гарантируется для Марковских процессов с бесконечным числом состояний. Для борьбы с этим эффектом в алгоритме DQN применено две эвристики: *повторение опыта* (*англ.* experience replay) и *целевая нейросеть* (*англ.* target network).

Повторение опыта заключается в сохранении всех встреченных эпизодов вида (s_t, a_t, r_t, s_{t+1}) в массиве и осуществлении обновления Q-функции после каждого действия не на основе этого действия и полученного за него вознаграждения, а на основе случайной выборки из k ранее встреченных эпизодов (k — параметр алгоритма). Таким образом, на каждом шаге обучения с некоторой вероятностью учитывается весь опыт игры, а не только опыт последнего события, что на практике приводит к стабилизации обучения.

Целевая нейросеть — это подход, заключающийся в использовании отдельной нейросети для вычисления опорных значений Q-функции при шаге Q-обучения. Веса из обучаемой нейросети копируются в целевую только раз в m шагов (m — параметр алгоритма). Этот подход призван стабилизировать обучение путем обеспечения одинаковых значений $Q(s_{t+1}, a)$ для одинаковых s_{t+1} и a на протяжении нескольких эпизодов.

В последующих работах алгоритм DQN был многократно модифицирован: были реализованы такие идеи, как *повторение опыта с приоритетом*

(*англ.* prioritized experience replay) [45], *двойное Q-обучение* (*англ.* double Q-learning) [23, 52] и Dueling DQN — оценка двух отдельных функций *ценности* (*англ.* value) $V(s)$ и *преимущества* (*англ.* advantage) $A(a)$ вместо оценки одной функции $Q(s, a)$.

Также были изучены перспективы применения рекуррентных нейросетей к задаче обучения с подкреплением: так, работа [24] продемонстрировала, что использование рекуррентного LSTM-слоя [25] улучшает качество обучения агента в частично наблюдаемых средах за счет частичного сохранения информации о прошлых наблюдениях. Одним из нашумевших примеров стало обучение рекуррентной нейронной сети игре в Doom [33].

В мультиагентном обучении с подкреплением нейросети были применены как в кооперативных, так и в некооперативных сеттингах. Одной из первых работ, посвященных обучению нейросетей с подкреплением в некооперативной среде, является [43]. В нем два агента обучались играть в игру Pong друг против друга. Примечательной работой, посвященной кооперативным играм, является [34]. В ней несколько агентов учились совместно решать головоломки, координируя свои действия с помощью сообщений, которым агенты тоже обучались.

1.3.3. DQN-routing

В бакалаврской выпускной квалификационной работе, выполненной автором данной работы ранее, а также в статье [42], был предложен алгоритм маршрутизации, объединениющий метода Q-routing с обучением нейросетей — *DQN-routing*. Алгоритм был представлен как универсальный алгоритм маршрутизации и протестирован в моделях компьютерной сети и конвейерной системы. Эта секция посвящена краткому обзору на него.

В листинге 4 приведена базовая логика алгоритма DQN-routing. Сразу же заметно, что структура алгоритма очень похожа на Q-routing. Однако, есть несколько ключевых отличий:

— Детали получения текущего состояния агента и порядок расчета вознаграждений абстрагированы;

— Агент использует стохастическую стратегию, полученную применением функции softmax к текущим значениям Q-функции для доступных действий;

Листинг 4 – Базовая логика алгоритма DQN-routing. x – метка текущего узла.

1: **function** Route(x, y, p) ▷ y – узел, пославший пакет, p – пакет
2: $s \leftarrow$ GetState(x, p)
3: $\pi(a|s) \leftarrow$ Softmax$(Q_x(s, a))$ ▷ Используем softmax-стратегию
4: $a \leftarrow \pi(a|s)$
5: $Q_e \leftarrow \sum_{a \in \mathcal{A}} Q_x(s, a)\pi(a|s)$
6: $D_p^x \leftarrow$ GetRewardData(s, p)
7: $S_x(p) \leftarrow (s, a, D_p^x)$
8: ReceiveReward(y, x, p, D_p^x, Q_e)
9: **return** a ▷ Возвращаем узел назначения
10: **end function**

11: **procedure** ReceiveReward(x, y, p, D_p^y, Q_e)
12: $(s, a, D_p^x) \leftarrow S_x(p)$
13: $r_a \leftarrow$ ComputeReward(D_p^x, D_p^y)
14: $\Delta Q \leftarrow r_a + Q_e - Q_x(s, a)$
15: $Q_x(s, a) \approx Q_x(s, a) + \Delta Q$ ▷ Шаг алгоритма оптимизации
16: **end procedure**

— Соответственно, оценка дальнейшей стоимости пути пакета вычисляется как матожидание Q-функции относительно полученной стохастической стратегии;
— Значение Q-функции обновляется не обычным сдвигом с некоторым learning rate, а с помощью алгоритма градиентной оптимизации (а именно, RMSProp [51])

Функция $Q_x(s, a)$ в алгоритме DQN-routing аппроксимируется feed-forward нейросетью с двумя скрытыми слоями по 64 нейрона. На вход нейросети подается текущее состояние агента s, на выходе нейросети получаются оценки $Q_x(s, a)$ для всех возможных действий a, где действие — это узел, к которому может быть перенаправлен пакет, причем если узел u не является соседом x, то $Q_x(d, u) = -\infty$ (рис. 5).

Алгоритм также предусматривает поддержку каждым агентом модели сети в виде графа с помощью протокола link-state (листинг 2). Построение текущего состояния агента приведено в листинге 5. Граф сети используется для включения в текущее состояние его матрицы смежности наряду с метками текущего узла, узла назначения и множеством текущих соседних узлов, закодированных унитарным кодом (*англ.* one-hot encoding).

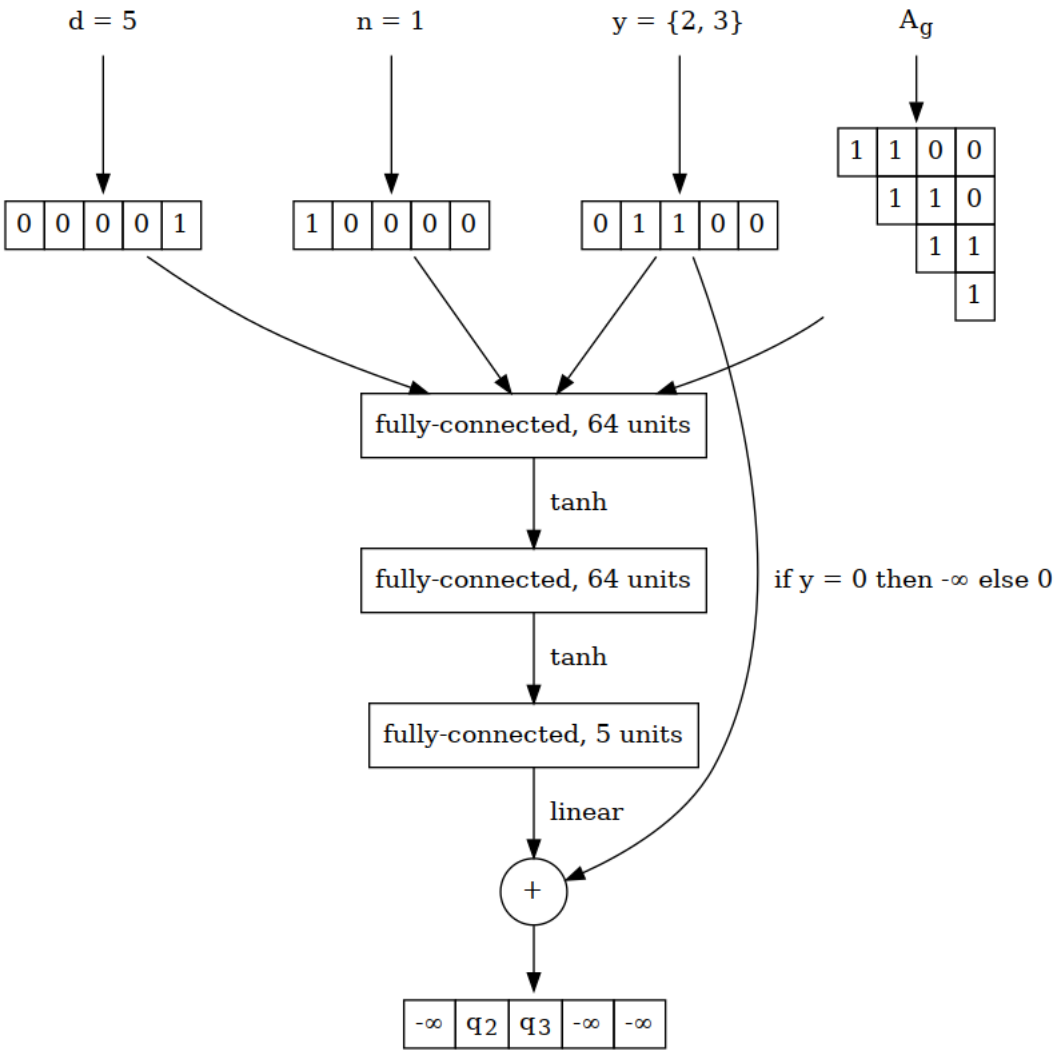


Рисунок 5 – DQN-routing: архитектура сети для графа из 5 узлов

Листинг 5 – Получение состояния агента в алгоритме DQN-routing

1: **function** GetState(x, p)
2: $d \leftarrow dst(p)$ ▷ Узел назначения пакета
3: $Y \leftarrow \{y|(x, y) \in E_x\}$ ▷ Множество соседей
4: $A_x \leftarrow$ AMatrix(G_x) ▷ Матрица смежности
5: $X \leftarrow$ AdditionalState(x, p)
6: **return** OneHot(x) $\odot$ OneHot(d) $\odot$ OneHot(Y) $\odot$ Flatten(A_x) $\odot$ X
7: **end function**

Реализация AdditionalState зависит от области применения алгоритма: в модели компьютерной сети эта функция возвращает пустой вектор, а в модели конвейерной системы она возвращает вектор статусов соседних конвейеров (1 – работает, 0 – не работает) в том же унитарном коде.

Реализация функций GetRewardData и ComputeReward также зависит от области применения:

— В модели компьютерной сети:
  – GetRewardData(s, p) = Time$()$;
  – ComputeReward$(t_x, t_y) = t_y - t_x$
— В модели конвейерной системы:
  – GetRewardData(s, p) = (Time$()$, EnergyGap(x, p));
  – ComputeReward$((t_x, _), (t_y, e_p)) = (t_y - t_x) + \alpha e_p$

То есть, в компьютерной сети вознаграждения аналогичны таковым в стандартном Q-routing, а в конвейерной системе при расчете вознаграждения учитывается вклад пересланного объекта в энергопотребление конвейера EnergyGap(x, p). Сам вклад рассчитывается как произведение постоянного энергопотребления конвейера k на неучтенное время работы: $e_p = p_k(t_y^k - t_e^k)$, причем после каждого такого расчета текущее время работы считается учтенным: $t_e^k \leftarrow t_y^k$.

В бакалаврской работе и [42] показано, что случайно проинициализированные нейросетевые агенты с данной архитектурой при работе онлайн не способны сойтись к оптимальной стратегии даже спустя долгое время, вследствие фундаментальной нестационарности мультиагентной системы. Для борьбы с этим предлагается использовать предварительное обучение с учителем, взяв за опорные значения функции $Q_s(d, x)$ длину кратчайшего пути между s и d, проходящего через x.

Также было показано, что применение experience replay ухудшает производительность предобученной модели в условиях изменяющегося трафика в сети.

Однако, DQN-routing не лишен недостатков. Наиболе существенным недостатком DQN-routing является то, что размер нейросети зависит от размера графа, в котором соответствующий агент работает как маршрутизатор. Причем, если размер выходного слоя нейросети зависит от размера графа линейно, то размер входного слоя зависит от него же квадратично, за счет получения на вход графа в виде матрицы смежности.

Это накладывает существенные ограничения на применимость алгоритма сразу в нескольких смыслах:

— Квадратичный рост входного слоя с ростом количества вершин в графе сильно ограничивает масштабируемость алгоритма, делая целесообразным его применение только в небольших графах до 100 вершин;

— Предобученная модель с заданной архитектурой может работать только в графах с количеством вершин n, где n — размер выходного слоя модели;
— Более того, необходимость предварительного обучения с учителем исключает свободный перенос модели на произвольный граф с n вершин; возможна адаптация только к топологиям, не слишком сильно отличающихся от встреченных во время предварительного обучения. Фактически, модель неустойчива даже к перенумерованию вершин.

Почти все эти недостатки являются прямым следствием использования унитарного кода при подаче номеров вершин на вход нейросети.

Помимо недостатков алгоритма самого по себе, также следует отметить, что для проведения экспериментов в бакалаврской работе и [42] использовалась крайне упрощенная модель конвейерной системы. В этой модели не отслеживались позиции отдельных объектов в системе, вместо этого их перемещение и уровни загрузки конвейеров моделировались с помощью абстрактных понятий *текущего выходного потока*, *текущего максимального потока* и *остаточного входного потока*, определенных для каждой из конвейерных секций. Столкновения объектов не моделировались вовсе, т. е. подзадача избежания столкновений в тех работах рассмотрена не была. Также, при моделировании энергопотребления конвейеров было принято, что конвейеры потребляют постоянное количество энергии за единицу времени, что, как было показано в разделе 1.1.3, достаточно далеко от реальности.

Так как бакалаврская работа, в которой был предложен алгоритм DQN-routing, была посвящена разработке обобщенного алгоритма маршрутизации, а не алгоритма управления конвейерной системой, использование подобной далекой от реальности модели конвейерной системы имело смысл — оно позволяло продемонстрировать, что алгоритм способен работать в системах, существенно отличающихся по поведению от компьютерных сетей. Однако, подобная модель не подходит для честной оценки работы алгоритма управления реальной конвейерной системой.

Таким образом, в настоящей работе можно выделить два ключевых направления:

— Разработка приближенной к реальности имитационной модели конвейерной системы

— Разработка и тестирование алгоритма управления конвейерной системой, использующего идеи DQN-routing, но избавленного от его ключевых недостатков.

Выводы по главе 1

В главе 1 была поставлена задача управления конвейерной системой. Задача была разбита на подзадачи: маршрутизацию, избежание столкновений и оптимизацию энергопотребления. Были рассмотрены существующие подходы к решению этих подзадач применительно к конвейерным системам разных типов, их преимущества, недостатки и границы применимости.

Также были рассмотрены существующие методы решения подзадачи маршрутизации применительно к компьютерным сетям, их преимущества и недостатки.

Задача управления конвейерной системой была 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. Архитектура системы и протокол взаимодействия между агентами

В данной работе мы предполагаем децентрализованную архитектуру системы, аналогичную той, что предложена в [3, 56] и частично описана в разделе 1.1.1.3. В этой архитектуре реальными агентами, принимающими решения, являются контроллеры отдельных конвейеров, управляющие собственной скоростью и поведением отклонителей в своих сочленениях.

Каждый контроллер соединен коммуникационным интерфейсом только с контроллерами смежных конвейеров. Каждый конвейер оснащен лазерными датчиками сумок, стоящими непосредственно перед каждым из сочленений конвейера и на его конце. Также конвейер оснащен датчиком скорости ленты и двигателем с переменной скоростью, что обеспечивает возможность точного контроля скорости.

В имитационной модели конвейерной системы, разработанной в рамках данной работы [4] на языке Python с использованием библиотеки дискретно-событийного моделирования Simpy, каждый конвейер моделируется как объект со следующими параметрами:

— длина L;
— максимальная скорость v_m;
— текущая скорость v;
— статус работы (включен/выключен) f_w;
— статус исправности (сломан/исправен) f_b;
— положения сочленений в форме расстояний от начала конвейера.

Заметим, что ширина самого конвейера не учитывается при моделировании — предполагается, что одна достаточна для проезда сумки, но что никакие сумки не могут иметь общие точки вдоль длины конвейера.

Ширина сочленений и сумок предполагается равной 1.5 метрам, то есть, прибывшая с бокового конвейера в смежное сочленение сумка оккупирует отрезок длиной в 1.5 метра на конвейере, на который она прибыла. Если же отклонитель производит отклонение в момент времени t, то считается, что сумка, пересекающаяся с шириной отклонителя не менее чем на 0.75 метров, переходит на смежный конвейер (т. е. центры сумки и отклонителя должны находиться друг от друга на расстоянии не более 0.5 м). Если же такой сумки не существует, то ничего не происходит.

Дискретизация непрерывного движения сумок на конвейере происходит следующим образом:

— Рассматриваются положения текущих сумок относительно *чекпоинтов*. Чекпоинтом является точка, при пересечении сумки с которой будет происходить некоторое событие, влекущее за собой взаимодействие с логикой контроллера (например, пересечение детектора) или с моделью другого конвейера (достижении сумкой конца конвейера).
— Определяется пара сумка-чекпоинт, для которой расстояние от переднего края сумки до чекпоинта по ходу движения конвейера минимально.
— Исходя из текущей скорости конвейера, рассчитывается время достижения сумкой чекпоинта и соответственно планируется обработка события достижения.
— При обработке любого события (не только запланированного в самой модели, но и, например, действия контроллера конвейера, такого как изменение скорости) производится расчет прошедшего времени с момента предыдущей обработки и позиции сумок сдвигаются на соответствующие расстояния по ходу движения конвейера.
— Если середина сумки достигла конца конвейера, или если сумка попала под отклонитель, сумка удаляется с текущего конвейера и помещается на следующий конвейер в соответствующей позиции.
— Если при помещении сумки на конвейер выяснилось, что она пересекается с уже находящейся на конвейере сумкой, то сумки помещаются впритык друг к другу, соприкасаясь в позиции, равной среднему позиций их центров, и регистрируется *столкновение*. Если же после этого сумки начинают задевать еще какие-то, находящиеся спереди или сзади, то задетые сумки сдвигаются соответственно вперед или назад так, чтобы находиться впритык к первым сумкам. Процесс продолжается каскадно, пока существуют вновь задетые сумки.

Из упрощений относительно реального мира следует отметить тот факт, что скорость конвейера изменяется мгновенно после получения сигнала об изменении скорости от контроллера. Также мы предполагаем, что сообщения между контроллерами передаются тоже мгновенно (что является более честным допущением, так как время передачи сигнала по проводу действительно пренебрежимо мало в масштабах времени конвейерных систем).

Теперь рассмотрим базовую логику контроллера конвейера. Как уже упоминалось, избежание столкновений является приоритетной подзадачей. Для его обеспечения контроллеры конвейера будут общаться по протоколу, описанному в разделе 1.1.1.3. Этот протокол является приоритетным: никакие решения об изменении скорости или запуске конвейера, сделанные другими компонентами логики контроллера, не имеют силы, если конфликтуют с требованиями протокола для избежания столкновений. За допустимое расстояние между сумками, поддерживаемое протоколом, возьмем 2 м.

Контроллер конвейера будет поддерживать модель позиций сумок точно так же, как и в [3]: конвейеры получают информацию о сумках от сканера на входе и передают ее соответствующему соседу при переходе этой сумке на соседа; при получении такого сообщения о входящей сумке конвейер добавляет ее в свою модель расположения сумок на себе, и при пересечении очередной сумкой лазерного детектора ее позиция в модели уточняется.

При пересечении сумкой лазерного детектора, стоящего перед отклонителем, контроллер определяет ID этой сумки и пункт ее назначения и определяет ее дальнейший маршрут. Если оказывается, что сумка должна переместиться на соседний конвейер, соответствующий отклонитель предпринимает движение отклонения.

Перейдем к логике маршрутизации. Каждое сочленение в системе будем представлять как узел графа, а конвейерные секции между ними — как ребра (рис. 6). С помощью такого представления мы можем свести задачу маршрутизации в конвейерной системе к общей задачи маршрутизации в направленном графе.

Как в случае предлагаемого алгоритма, так и в случае алгоритма Q-routing, будем считать, что в каждом сочленении, в том числе в сходящихся сочленениях, находится маршрутизирующий агент. Несмотря на то, что фиктивные маршрутизирующие агенты, находящиеся в сходящихся сочленениях, не имеют выбора, куда послать сумку, они нужны, чтобы в момент появления сумки в сочленении предсказывать стоимость ее дальнейшего пути и посылать соответствующее сообщение с вознаграждением агенту-отправителю сумки.

Предполагается, что каждому конвейеру известна собственная длина и длины собственных секций, соответствующие весам ребер в графовом представлении системы. Конвейер формирует и поддерживает в памяти эту модель

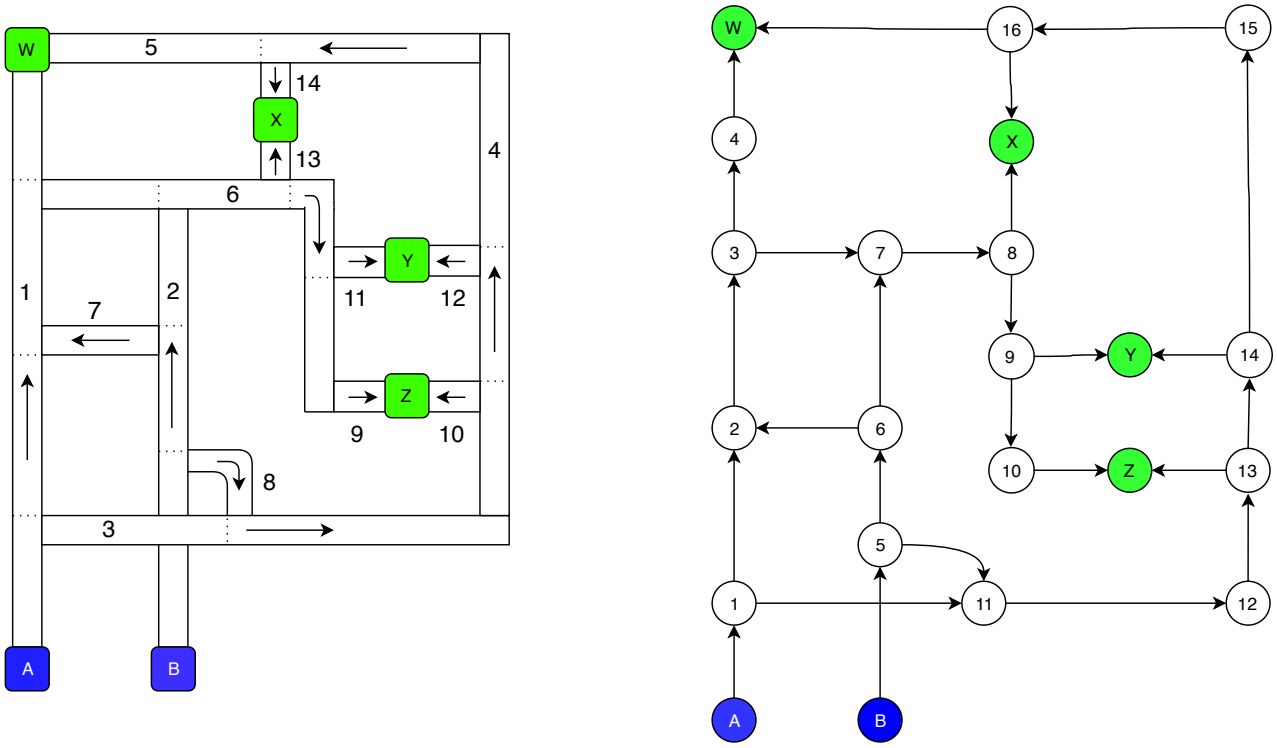


Рисунок 6 – Пример представления конфигурации конвейерной системы в виде направленного графа

с помощью link-state протокола. Так как эта модель является ориентированным графом, некоторые входные вершины могут быть недостижимыми из каких-либо вершин. Чтобы избежать направления сумки в то место, откуда она не сможет добраться до точки назначения, перед принятием решения о перенаправлении сумки вне зависимости от основной логики конвейера производится проверка на достижимость с помощью этой графовой модели, и рассматриваются только варианты перенаправления сумки в точки, из которых достижим ее узел назначения.

Также предполагается, что конвейер может автоматически определять, что соседний конвейер вышел из строя. В таком случае конвейер обновляет граф системы соответствующим образом и распространяет это обновление с помощью link-state протокола.

В отличие от работ [3, 56], в этой работе мы не уделяем внимания подробным техническим подробностям реализации описанной логики в рамках определенного стандарта. Вместо этого мы фокусируемся на высокоуровневой логике работы агентов и работаем с математическими моделями, не содержащими большого количества технических деталей.

2.2. Формулировка задачи в терминах обучения с подкреплением

В разделе 1.1 задача управления конвейерной системой была разбита на три подзадачи: маршрутизацию, избежание столкновений и энергосбережение. Там же было указано, что недопустимость столкновений является наиболее строгим ограничением в задаче, и был описан простой и надежный существующий децентрализованный метод, гарантирующий отсутствие столкновений [3]. В связи с этим будем в нашем методе пытаться решать только первую и третью подзадачи, и использовать его в комбинации с описанным методом избежания столкновений для решения всей задачи целиком.

Задача маршрутизации заключается в минимизации среднего времени сумки в пути до точки выгрузки:

$$J'_R = \frac{1}{N} \sum_{i=0}^{N} (t_d^i - t_s^i)\,, \tag{11}$$

где t_s^i — время появления i-ой сумки в системе (сканирования), и t_d^i — время ухода i-ой сумки в точку выгрузки, N — общее количество сумок. Если общее количество сумок N зафиксировано (а так как этот параметр не является контролируемым, можем считать его зафиксированным), то с тем же успехом можно минимизировать просто сумму времен сумок в пути:

$$J_R = \sum_{i=0}^{N} (t_d^i - t_s^i)\,, \tag{12}$$

Задача энергосбережения заключается в минимизации суммарного количества потребленной энергии:

$$J_E = \int_0^T \sum_{j=0}^{M} p_j(t)\ \mathrm{d}t\,, \tag{13}$$

где $p_j(t)$ — удельное энергопотребление конвейера j в момент времени t, M — количество конвейеров, T — полное время работы системы.

Соответственно, совместную задачу маршрутизации и энергосбережения можно представить как минимизацию функции

$$J = \theta_R J_R + \theta_E J_E\,, \tag{14}$$

где θ_R и θ_E — это веса приоритета, параметры задачи. Эквивалентно для удобства можно записать

$$J = J_R + \alpha J_E \,, \tag{15}$$

где $\alpha = \frac{\theta_E}{\theta_R}$ — приоритет оптимизации энергопотребления.

Чтобы свести эту задачу к задаче обучения с подкреплением, то есть к задаче поиска *оптимальной стратегии*, нужно выразить J в виде (7). Чтобы это сделать, нам нужно определиться с тем, что в системе является агентом, каково множество состояний агента $\mathcal{S}$, каково множество действий $\mathcal{A}$ и как определена функция вознаграждения $r : \mathcal{S} \times \mathcal{A}_s \to \mathbb{R}$.

Так как в наших целях разработка децентрализованного алгоритма, логично предположить, что нужно рассматривать мультиагентный сеттинг, где в качестве агента выступает контроллер отдельного конвейера, отвечающий за управление им и отклонителями на его сочленениях, по аналогии с архитектурой, описанной в разделе 1.1.1.3.

Если в этом случае рассматривать только функцию энергопотребления J_E, и предположить, что конвейер j может отслеживать суммарное количество собственной потребленной энергии на момент времени t:

$$P_j(t) = \int_0^t p_j(t) \, \mathrm{d}t \,, \tag{16}$$

то достаточно взять в качестве действий изменение скорости и включение/выключение питания, а в качестве состояний, скажем, текущую скорость и количество сумок на конвейере. Тогда, выбрав некоторый период дискретизации времени δt, мы можем определить функцию вознаграждения как $r(s_t, a_t) = -(P_j(t + \delta t) - P_j(t))$. Тогда, взяв параметр $\gamma = 1$, мы увидим, что формулы (13) и (7) эквивалентны.

Подвох заключается в том, что в таком случае контроллер конвейера быстро выяснит, что в любой ситуации ему не стоит двигаться, так как в разделе 1.1.3 было выяснено, что энергопотребление уменьшается с уменьшением скорости. Такое поведение конфликтует с тем, что нам необходимо все-таки доставлять сумки до точек назначения.

Можно предложить приоритизировать перемещение сумок над снижением энергопотребления в рамках одного конвейера: конвейер двигается с мак-

симально возможной скоростью всегда, когда на нем находятся сумки, и останавливается после того, как сумок на нем не осталось. То есть, мы оставляем конвейеру из множества действий только действия “on”/“off”, тем самым сводя задачу оптимизации J_E к задаче оптимального выключения.

Решение этой задачи в рамках одного конвейера элементарно и заключается в выключении конвейера спустя определенный промежуток времени d_t после того, как его покинула последняя сумка. Задержка d_t нужна потому что, во-первых, конвейер не стартует и не запускается мгновенно, а имеет некоторые периоды разгона и торможения, а во-вторых, потому что на разгон и торможение в реальности уходит больше энергии, чем на равномерное движение за то же время. Поэтому останавливаться моментально после того, как последняя сумка покинула конвейер, неоптимально.

Если мы ограничим поведение агента таким образом, то, на первый взгляд, нам останется решить только задачу маршрутизации — отдельно минимизировать J_R. Разобъем время сумки i в пути от сканера к выгрузке $t_d^i - t_s^i$ на времена пути по отдельным секциям конвейеров: $t_d^i - t_s^i = \sum_{j=0}^{n-1}(t_{j+1}^i - t_j^i)$, где t_j^i — это время достижения сумкой i j-го сочленения на пути, $t_0^i = t_s^i$, $t_n^i = t_d^i$. Тогда, рассматривая в качестве действий отклонение или пропускание сумки на сочленении с отклонителем, можно рассмотреть в качестве вознаграждения время в пути, которое сумка проведет до следующего отклонителя. Тогда, на первый взгляд, выражение для J_R тоже совпадет с формулой (7) и задачу можно будет считать решенной. Однако, это не совсем так: это приведет к тому, что каждый отдельный конвейер будет вести себя жадно — всегда отклонять сумки вдоль самого быстрого пути до следующего сочленения, потому что это будет максимизировать сумму вознаграждений, которые получает именно он. Такая стратегия далеко не всегда будет приводить к тому, что сумки будут отправляться по скорейшему пути до точки назначения.

Если же рассмотреть в качестве формального агента в системе не конвейер, а *сумку*, и рассмотреть ее текущую позицию в системе как часть ее наблюдаемого состояния, то для такого агента минимизация суммы выигрышей действительно будет приводить к движению по кратчайшему пути. Значит, для этого агента можно (потенциально) найти оптимальную стратегию маршрутизации с помощью Q-обучения.

Конечно, сумка не может физически реализовывать логику маршрутизации. Вместо этого это делает контроллер конвеера, на котором она сейчас находится. Заметим также, что путь сумки в системе конечен, а значит, конечно и количество действий сумки-агента. Это значит, что параметр γ в (9) можно взять равным 1. Тогда эта формула примет вид:

$$Q_x(o_t, a_t) = Q_x(o_t, a_t) + \alpha \left(r_t + \gamma \cdot \max_{a \in \mathcal{A}_{o_{t+1}}} Q_y(o_{t+1}, a) \right) \tag{17}$$

Здесь x и y обозначают конвейеры, между которыми переместилась сумка (x может совпадать с y), а r_t — это вознаграждение, полученное ей за действие перенаправления (на самом деле, контроллером конвейера, который совершил действие перенаправления, а именно, x).

Если взять в качестве r_t просто время в пути между сочленениями с минусом, то максимизация суммы r_t будет эквивалентна минимизации J_R. Если к тому же в качестве наблюдения сумки взять только идентификатор текущего сочленения, то получится в точности алгоритм Q-routing.

Мы уже определились, что если ограничить доступные действия конвейерной ленты как “on”/“off”, то для оптимизации энергопотребления достаточно простой стратегии остановки с задержкой после ухода сумок с конвейера. Значит ли это, что использование алгоритма Q-routing в неизменном виде приведет к оптимальной оптимизации энергопотребления в конвейерной системе? Нет, это не так: прошлые рассуждения работали при условии, что агентом в системе считается конвейер, а не сумка. Требуется модифицировать вознаграждения, чтобы учитывать затраты на энергопотребление при маршрутизации. Если этого не сделать, то оценки стоимостей путей сумок не будут никак учитывать энергопотребление, что сделает невозможным, например, перенаправление сумок по более длинному пути вдоль уже работающих конвейеров ради экономии энергии.

Будем рассчитывать полную стоимость пути сумки i как

$$C_i = (t_d^i - t_s^i) + \alpha e^i \,, \tag{18}$$

где t_s^i — время появления сумки в источнике, t_d^i — время достижения сумкой точки выгрузки, α — это коэффициент важности энергосбережения (параметр), а e^i — это энергетические затраты на перемещение чемодана, по-

рядок расчета которых нам предстоит определить. Суммарная стоимость всех путей, таким образом, будет равна

$$J_c = \sum_{i=0}^{N} C_i = \sum_{i=0}^{N} (t_d^i - t_s^i) + \alpha \sum_{i=0}^{N} e^i = J_R + \alpha \sum_{i=0}^{N} e^i \tag{19}$$

Заметим, что J_c отличается от J (уравнение (15)) только тем, что вместо J_E стоит сумма энергозатрат на перемещение всех сумок $\sum\limits_{i=0}^{N} e^i$. То есть, если мы определим энергозатраты на перемещение сумки e^i таким образом, что сумма всех таких энергозатрат будет отличаться от J_E на константу, то минимизация J_c будет эквивалентна минимизации J.

Пусть e^i считается как сумма энергозатрат на перемещение сумки вдоль каждой конвейерной секции на пути сумки i:

$$e^i = \sum_{j \in path(i)} e_j^i \tag{20}$$

Как определить e_j^i? Сделаем так, чтобы сумма всех энергозатрат $\sum\limits_{i=0}^{N} e_j^i$ для всех секций j, принадлежащих конвейеру k, отличалась от суммарных энергозатрат на работу этого конвейера на константу:

$$\int_0^T p_k(t)\ \mathrm{d}t = \sum_{j \in sections(k)} \sum_{i=0}^{N} e_j^i + C \tag{21}$$

Если мы сумеем этого добиться, то тогда J_c и J будут эквивалентны. Определим e_j^i так:

$$e_j^i = \int_{t_k^e}^{t_j^i} p_k(t)\ \mathrm{d}t = P_k(t_j^i) - P_k(t_k^e)\,, \tag{22}$$

где t_j^i — это время покидания сумкой i секции j, а t_k^e — это *время последней регистрации* потраченной конвейером k электроэнергии.

Поясним смысл величины t_k^e. Будем считать, что расчет e_j^i производится в тот же момент, что и расчет временных затрат на прохождение сумкой i секции j — то есть, в момент покидания сумкой этой секции. Если в момент t_j^i покидания сумкой любой из секций конвейера мы будем регистрировать текущее суммарное энергопотребление конвейера $P_k(t_j^i)$, то e_j^i можно считать как

$$e_j^i = P_k(t_j^i) - P_k^r\,, \tag{23}$$

где P_k^r — это предыдущее зарегистрированное суммарное энергопотребление (т. е. значение P_k^r обновляется каждый раз, когда сумка покидает одну из секций конвейера k).

Таким образом, энергозатраты на перемещение сумки i вдоль секции j имеют смысл “количество потраченной энергии, которое было учтено вследствие проезда сумки по секции”. Энергозатраты на перемещение всех сумок по конвейеру тогда составят

$$P'_k = \sum_{i=1}^{c_k}(P_k(t_i) - P(t_{i-1})) = P_k(t_{c_k}) - P_k(t_0) = P_k(t_{c_k})\,, \tag{24}$$

где c_k — количество всех событий пересечения сумкой какого-либо сочленения конвейера k, t_i — время i-го такого события, $t_0 = 0$, $P_k(0) = 0$.

То есть, сумма определенных нами энергозатрат на прохождение сумок вдоль одного конвейера оказывается равной суммарным энергозатратам этого конвейера на момент времени t_{c_k} ухода последней сумки с этого конвейера. Однако, конвейер останавливается только через некоторое время d_t после ухода с него последней сумки, так что эта величина не равна полному количеству потребленной конвейером энергии за все время. Однако, так как d_t — это константная величина, и так как после t_{c_k} на конвейере по определению не осталось сумок и он может двигаться с максимально возможной скоростью, дополнительные энергозатраты на этот финальный этап работы конвейера можно считать константными. Действительно, как было рассмотрено в разделе 1.1.3, на энергопотребление конвейера из переменных параметров влияют только свойства перемещаемого груза и скорость конвейерной ленты. После t_{c_k} свойства перемещаемого груза постоянны (груза на ленте нет), и скорость может быть постоянной (максимальной). Таким образом, энергию, потребленную конвейером после t_{c_k} можно считать константой, что приводит к тому, что тождество 21 выполняется.

Итак, мы определили стоимости прохождения сумки вдоль секций конвейера таким образом, чтобы минимизация суммарной стоимости всех путей сумок была эквивалентна минимизации истинной целевой функции J. Тогда определим вознаграждение за перенаправление сумки i на секцию j как

$$r_j^i = -(t_j^i + \alpha e_j^i)\,, \tag{25}$$

где t_j^i — время прохождения сумкой секции. При таких вознаграждениях максимизация суммы вознаграждений (7) для агента-сумки эквивалентно минимизации стоимости пути. Таким образом, используя принцип Q-обучения для обучения агентов, мы будем минимизировать стоимость пути для каждой сумки в отдельности. Это не строго эквивалентно минимизации суммарной стоимости путей всех сумок J_c (что, как было показано, эквивалентно минимизации J), так как пути сумок пересекаются и действия одного агента влияют на вознаграждения другого, но в большинстве случаев, в особенности в отсутствие заторов на конвейерах, решениями этих двух задач являются одни и те же (или очень похожие) стратегии.

2.3. Алгоритм DQN-LE-routing

Так как в разделе 2.2 в процессе сведения задачи управления конвейерной системой к задаче обучения с подкреплением получилось, по сути, что все свелось к задаче маршрутизации, мы будем разрабатывать алгоритм в два этапа:

а) Предложение нейросетевого алгоритма маршрутизации в абстрактной компьютерной сети, аналогичного Q-routing;
б) Проведение первичного экспериментального исследования алгоритма в модели абстрактной компьютерной сети;
в) Модификация предложенного алгоритма для работы в конвейерной системе.

Такой подход имеет следующие преимущества:

а) Маршрутизация в абстрактной компьютерной сети — это почти подзадача маршрутизации в конвейерной сети в чистом виде, проще сначала решить подзадачу;
б) Симуляция компьютерной сети существенно менее вычислительно затратна, что позволяет проще проводить эксперименты на начальном этапе исследования;
в) Демонстрируется обобщаемость предложенного метода на более широкий класс задач.

В этом разделе в соответствии с этим подходом будет предложен алгоритм маршрутизации *DQN-LE-routing*, являющийся модификацией DQN-routing, избавленной от большинства его ключевых недостатков.

Как было указано в разделе 1.3.3, корнем большинства проблем с алгоритмом DQN-routing является использование унитарного кода для кодирования подаваемых на вход нейросети вершин в графе. Номера вершин нельзя подавать на вход в чистом виде, так как это неупорядоченные категориальные данные — близость номеров 9 и 10 не имеет ничего общего с близостью вершин 9 и 10 в случайном графе, но нейронная сеть устроена так, что близость значений на выходном слое коррелирует с близостью значений на входном.

Унитарный код является универсальным методом для кодирования любых категориальных данных, предотвращающим любые нежелательные корреляции. Однако, не все корреляции являются нежелательными — в случае вершин в графе, более близкие по реберному расстоянию вершины имеет смысл кодировать более близкими значениями.

Методы отображения вершин графа в векторные пространства с более низкими размерностями, сохраняющие после отображения некоторую заданную метрику *схожести* вершин, называются *графовыми эмбеддингами* (*англ.* graph embeddings). Существует множество алгоритмов для получения эмбеддингов из вершин графа [11, 20], различающихся по свойствам и вычислительной сложности. Большинство современных методов получения графовых эмбеддингов можно подразделить по принципу работы на три семейства:

— Методы матричной факторизаци: Laplacian Eigenmaps [8], HOPE [7];
— Методы случайных путей (*англ.* random walks): DeepWalk [44], node2vec [21];
— Методы глубокого обучения: SDNE [54].

Методы глубокого обучения и методы случайных путей, как правило, применяются для построения эмбеддингов в очень больших графах, таких, как социальные сети. Методы глубокого обучения также могут учитывать гетерогенную информацию об узлах и ребрах графа и, как правило, работают достаточно универсально с различными метриками стоимости узлов. Однако, методы из этих классов требуют существенно большего количества вычислительных ресурсов, чем методы матричной факторизации, в случае относительно небольших графов (в пределах нескольких тысяч вершин). Также, и методы

случайных путей и методы глубокого обучения являются недетерминированными.

Так как для рассматриваемых в данной работе задач маршрутизации актуальны именно относительно небольшие графы, и детерминированное получение одних и тех же эмбеддингов для одного и того же графа является весьма желательным свойством, мы будем исследовать только методы матричной факторизации.

Метрики похожести вершин, которые графовые эмбеддинги должны сохранять, подразделяются на *локальные*, которые обозначают как похожие только вершины, близкие реберно, и *нелокальные*, которые могут обозначать похожими вершины, расположенные в графе далеко друг от друга.

В ходе экпериментального сравнения нескольких методов матричной факторизации, использующих различные метрики похожести (раздел 3.1), было установлено, что для решения задачи маршрутизации лучше всего подходит метод Laplacian Eigenmaps [8], использующий локальную метрику схожести, основанную на реберном расстоянии между вершинами. Метод заключается в следующем: при данной взвешенной матрице смежности графа W и заданной размерности эмбеддинга k решается обобщенная задача нахождения собственных векторов

$$Ly = \lambda Dy\,, \tag{26}$$

где D - это диагональная матрица взвешенных степеней вершин графа, $D_{ii} - \sum_j W_{ji}$, L – это матрица Кирхгофа для графа, $L = D - W$.

Если полученные собственные вектора y_i отсортировать по возрастанию соответствующих собственных чисел λ_i, то тогда результатом работы алгоритма являются k собственных векторов $y_1, ..., y_{k+1}$. Эмбеддинг для i-го узла в сети можно получить как $(y_1(i), ..., y_{k+1}(i))$.

Замечание: метод предполагает, что чем больше вес W_{ij} ребра (i, j), тем ближе должны оказаться получившиеся эмбеддинги для узлов i и j. Однако, если веса имеют смысл “расстояний” между вершинами (например, длин конвейерных секций между сочленениями), то желательно обратное поведение. Поэтому на самом деле матрицу W мы будем получать, инвертируя веса в изначальной матрице смежности графа A:

$$W_{ij} = \begin{cases} 0, & \text{if} A_{ij} = 0 \\ \exp(-A_{ij}^2), & \text{otherwise} \end{cases} \tag{27}$$

Обозначим процедуру получения эмбеддингов из графа G как $LE(G)$, результатом процедуры будет матрица из собственных векторов $Y = LE(G)$ размерности $k \times n$, такая, что i-ый столбец матрицы Y_i является эмбеддингом для i-го узла.

Преимущества данного метода:

— Простота реализации

— Скорость работы $O(|E|k^2)$ [20] (при использовнии эффективной реализации разреженных матриц), что хорошо для разреженных графов — а графы, представляющие конвейерные системы, достаточно разрежены;

— Учет локальной схожести вершин — близкие в графе вершины будут иметь близкие эмбеддинги, в то же время стоимости путей до близких вершин должны быть близки.

— Линейность — при одновременном сдвиге на некоторый вектор значения эмбеддингов не теряют смысла.

Данный метод работает только для ненаправленных графов, поэтому при маршрутизации в направленных графах (как в конвейерных системах) будем строить эмбеддинги по ним как по ненаправленным.

Кроме того, нам хочется, чтобы полученные эмбеддинги различались в графах, отличающихся только весами ребер. Смысл этого таков: если мы увеличим вдвое длины всех конвейеров в системе (и, соответственно, увеличим вдвое веса ребер в соответствующем топологическом графе), то и средняя стоимость перемещения сумки вдоль секции в этой системе возрастет примерно вдвое. Но при этом эмбеддинги, посчитанные от двух таких графов, получатся почти одинаковыми, что приведет к тому, что нейросеть будет выдавать по ним почти одинаковые предсказания стоимостей путей, хотя они должны быть разными.

Поэтому перед подсчетом $LE(G)$ будем находить средний вес ребра w' в G и делить веса ребер на него. После получения матрицы Y будем домножать ее на w', получая, таким образом, что в графе, средняя длина ребер в котором в β раз больше, чем в данном, среднее расстояние между эмбеддингами будет также в β раз больше.

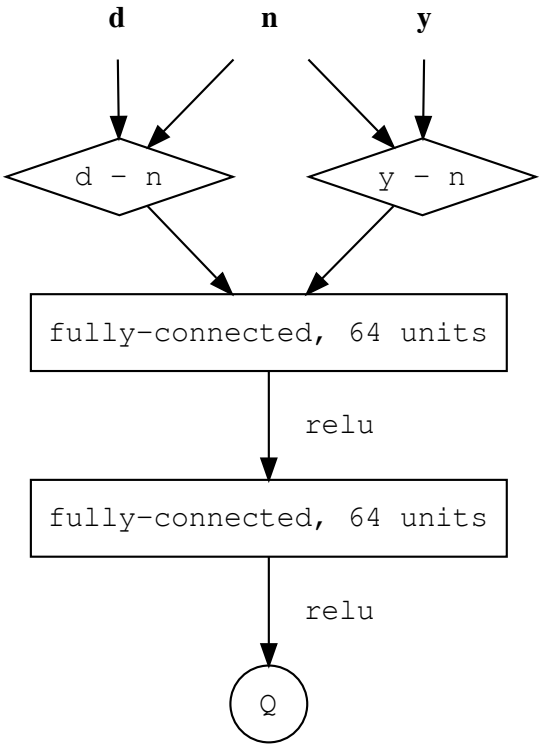


Рисунок 7 – DQN-LE-routing: архитектура сети

Кодируя номера узлов эмбеддингами размерности k, мы только частично решаем проблему зависимости размеров нейросети от количества узлов в графе. Выходной слой все еще имеет размер n, и на вход все еще подается матрица смежности.

Первую проблему решим просто: откажемся от предсказания значений функции $Q(s, a)$ сразу для нескольких действий (соседних узлов) a, и вместо этого будем подавать конкретного соседа в виде эмбеддинга на входной слой.

Вторую проблему решим, просто отказавшись от подачи матрицы смежности на вход. Она была необходима для того, чтобы адаптироваться к изменениям в топологии сети. Однако, эмбеддинги уже содержат в себе информацию о топологии, поэтому при ее изменении будем просто перестраивать $LE(G)$ на обновленном графе G.

Так как эмбеддинги линейны, каждый отдельный агент может считать, что он находится в “центре мира” — начале координат, а эмбеддинги остальных узлов сдвинуты соответствующим образом. Таким образом, на вход нейросети достаточно подать только два сдвинутых эмбеддинга, соответ-

ствующих узлу назначения и соседу, для которого делается предсказание Q-функции.

Итого,

$$Q_n(d, y) = f_\theta((LE(G)_d - LE(G)_n) \odot (LE(G)_y - LE(G)_n)), \qquad (28)$$

где f_θ — это feed-forward нейросеть (рис. 7)

Эксперименты в модели абстрактной компьютерной сети (раздел 3.2) показывают, что полученная модификация алгоритма, которую мы назовем DQN-LE-routing, способна адаптироваться к изменениям топологии и нагрузки не хуже оригинального DQN-routing, но при этом обученная модель способна работать также и в ранее не встреченных графах, в том числе графах большей размерности.

При работе в конвейерной сети кроме задачи быстрой доставки объектов также стоит задача экономии энергии. В разделе 2.2 было установлено, что во избежание “конфликта интересов” между энергосбережением и снижением времени доставки мы не будем пытаться снижать энергопотребление путем замедления конвейеров, а будем только останавливать их после того, как на них не осталось сумок. В таких ограничениях был предложен способ включения энергозатрат в стоимость пути сумки и соответствующий ему порядок расчета вознаграждений для агентов.

Однако, вклад энергозатрат в полученное вознаграждение зависит далеко не только от взаимного положения текущего узла, его соседей и узла назначения сумки в графе топологии системы. Например, если конвейер был выключен до перенаправления на него сумки, вновь учтенные энергозатраты на перемещение первой сумки, что вызвала его включение, будут существенно выше, чем учтенные энергозатраты на перемещение очередной сумки, если конвейер уже работает и сумки на нем уже есть. Кроме того, как было рассмотрено в разделе 1.1.3, реальное энергопотребление конвейера всегда зависит от его скорости и массы перемещаемого груза.

Исходя из этого, будем подавать на вход нейросетевому агенту дополнительную информацию, призванную помочь в оценке реальной стоимости пути. Пусть рассматривается соседний узел y, который относится к конвейеру k, тогда, кроме сдвинутого эмбеддинга для y подадим на вход:

— Информацию о том, работает ли конвейер k: один бит информации, 0 или 1;
— Текущую скорость конвейера k: одно вещественное число;
— Текущее количество сумок на конвейере k.

Выводы по главе 2

В данной главе была описана модификация разработанного ранее алгоритма DQN-routing — DQN-LE-routing, избавленная от ключевых недостатков первого. Также был предложен метод расчета вознаграждений для агентов в конвейерной сети, позволяющий оптимизировать ее энергопотребление, и было обосновано, как подача дополнительных данных на вход нейросетевого агента поможет ему лучше предсказывать финальные затраты на энергопотребление.

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 [2]), 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.

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

3.1. Выбор используемых графовых эмбеддингов

При выборе типа графовых эмбеддингов для кодирования информации об узлах графа перед подачей на входной слой нейросети были рассмотрены два метода получения эмбеддингов, основанных на матричной факторизации — Laplacian Eigenmaps (LE) и HOPE.

Первый работает только на неориентированных графах и только с локальной метрикой похожести, основанной на длинах ребер между смежными вершинами; второй поддерживает ориентированные графы и несколько метрик похожести:

а) Common Neighbors (CN) — число общих соседей между двумя вершинами, локальная метрика;

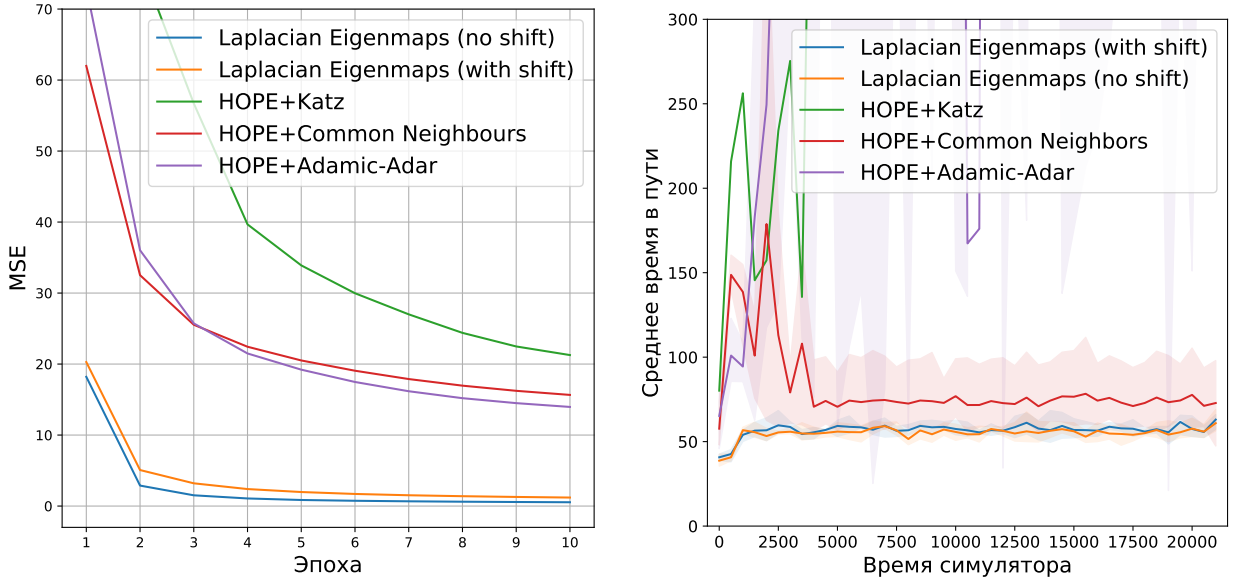
б) Adamic-Adar (AA) — то же, что и CN, но каждый общий сосед имеет вес, равный его инвертированной степени, то есть, вершины с большим количеством соседей вносят меньше вклада в вес;

в) Katz Index — нелокальная метрика, равная взвешенной сумме стоимостей всех путей между двумя вершинами; чем больше количество ребер в пути, тем меньше его вес.

Также, для LE рассматриваются два варианта подачи базового текущего состояния на вход нейросети:

а) Без сдвига (no shift) — на вход нейросети подаются три эмбеддинга, соответствующие узлам d (узел назначения), n (текущий узел) и y (рассматриваемый сосед;

б) Со сдвигом (with shift) — эмбеддинги для d и y сдвигаются на эмбеддинг для n и на вход нейросети подаются *два* эмбеддинга: $d - n$ и $y - n$.



(a) Скорость предобучения

(b) Работа в симуляторе

Рисунок 8 – Сравнение качества работы при использовании различных эмбеддингов

Варианты нейросетевых агентов с использованием Laplacian Eigenmaps и HOPE с перечисленными метриками похожести были сравнены по скорости предобучения на датасете, подготовленном для предобучения агентов при экспериментах в модели абстрактной компьютерной сети (раздел 3.2).

На графике 8a видно, что использование LE приводит к быстрому снижению средней ошибки почти до нуля, причем при осуществлении сдвига обучение происходит незначительно медленнее, в то время как при использовании HOPE с любыми метриками похожести ошибка остается достаточно высокой и после десяти эпох обучения. Причем использование нелокальной метрики (Katz index) приводит к наихудшим результатам.

График 8b демонстрирует работу алгоритма при использовании различных эмбеддингов в модели компьютерной сети при равномерной умеренной нагрузке. Здесь видно, что при использовании HOPE с метриками Katz и Adamic-Adar наблюдается хаотичное поведение, а при использовании метрики Common Neighbors наблюдается схождение к стабильному, но явно неоптимальному поведению. При использовании же LE в обоих вариантах наблюдается стабильное высокопроизводительное поведение.

Из данных экспериментов можно сделать вывод, что HOPE-эмбеддинги не подходят для использования в алгоритме ни в одном из вариантов, а Laplacian Eigenmaps работают одинаково хорошо в обоих рассмотренных вариантах применения. Но при использовании сдвига входной слой нейросетевого агента становится меньше на треть, что уменьшает количество оптимизируемых параметров и вычислительные затраты при работе алгоритма. Поэтому в дальнейших экспериментах будут применяться именно LE-эмбеддинги со сдвигом.

3.2. Эксперименты в модели абстрактной компьютерной сети

При проведении экспериментов в модели абстрактной компьютерной сети алгоритм DQN-LE-routing сравнивался со следующими бейзлайнами:

— **Shortest paths (SP)**: алгоритм кратчайших путей с постоянными весами ребер использующий протокол link-state;

— **Q-routing**: алгоритм на основе табличного обучения с подкреплением, описанный в разделе 1.2.3, на идеях которого базируются алгоритмы DQN-routing и DQN-LE-routing;

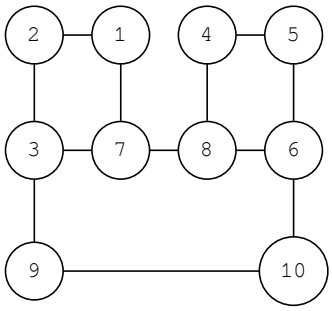


Рисунок 9 – Базовый граф для тестов в модели компьютерной сети

— **DQN-routing**: алгоритм, ранее предложенный автором данной работы, рассмотренный в разделе 1.3.3.

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. 9). 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 [51].

3.2.1. Адаптация к изменению нагрузки на сеть

В этом сценарии пакеты перемещаются между двумя половинами графа 9 (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.

При резком повышении нагрузки роутеров статический link-state алгоритм не способен работать эффективно, как видно на графике 10, так как очереди маршрутизаторов, через которых проходит большинство кратчайших путей между половинами сети (узлы 7 и 8 на рис. 9), начинают переполняться, и пакеты начинают застревать в этих очередях, вызывая стремительный рост

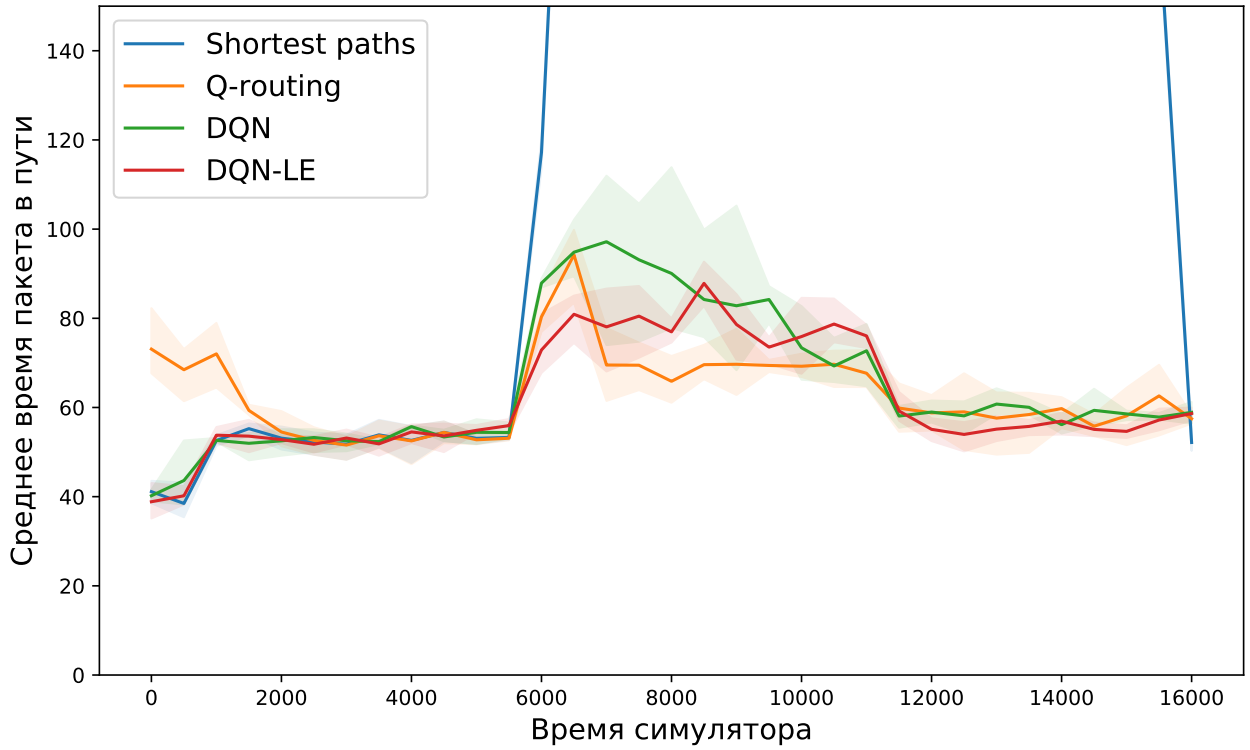


Рисунок 10 – Резкое понижение и последующее снижение нагрузки

среднего времени на доставку. Все прочие алгоритмы оказываются способны быстро обнаружить затор и направить трафик по обходному пути.

3.2.2. Адаптация к обрывам и восстановлениям соединений

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 9.

Как видно на графике 11, 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. При этом DQN-LE-routing обеспечивает такое поведение без использования подачи матрицы смежности графа на вход нейросетевого агента, только за счет перестроения эмбеддингов при изменении топологии сети.

3.2.3. Перенос опыта на новые топологии графов

Необходимость предобучения для работы в графе заданной топологии является существенным недостатком алгоритма DQN-routing, сильно ограни-

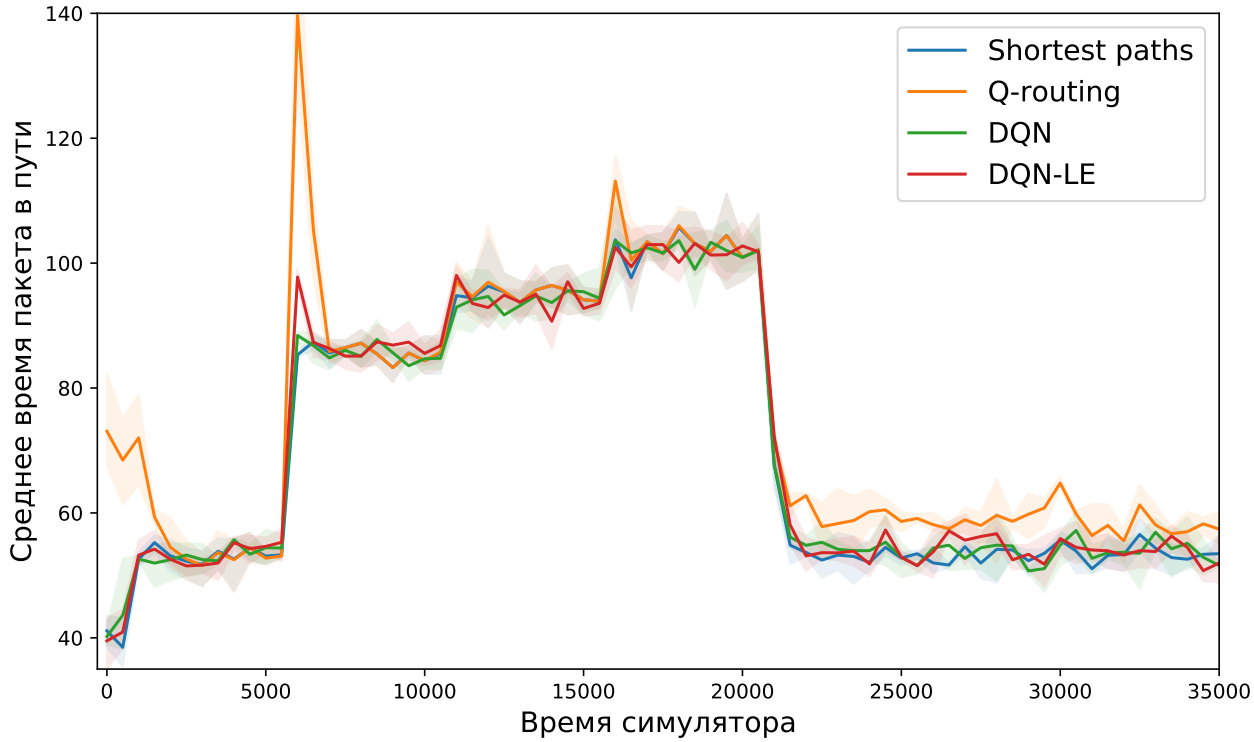


Рисунок 11 – Обрыв и последующее восстановление трех соединений

чивающим его практическую ценность. Алгоритм DQN-LE-routing тоже требует предобучения, но благодаря использованию графовых эмбеддингов обобщающая способность единожды обученной модели гораздо выше.

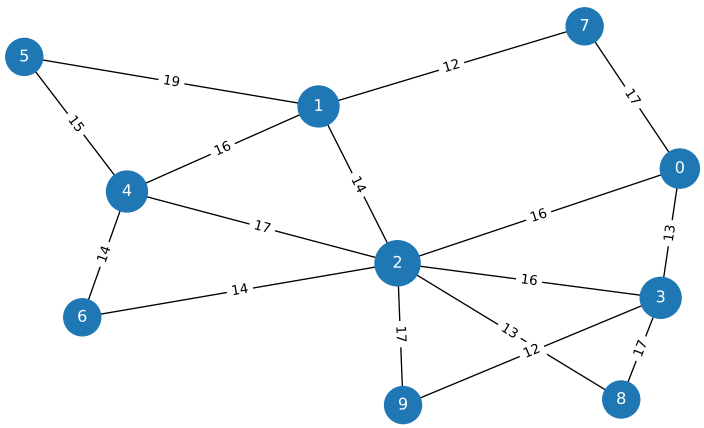


Рисунок 12 – Случайный граф из 10 вершин

Нейросетевые агенты для алгоритмов DQN- и DQN-LE-routing, обученные на данных с базового графа из 10 вершин (рис. 9), 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.

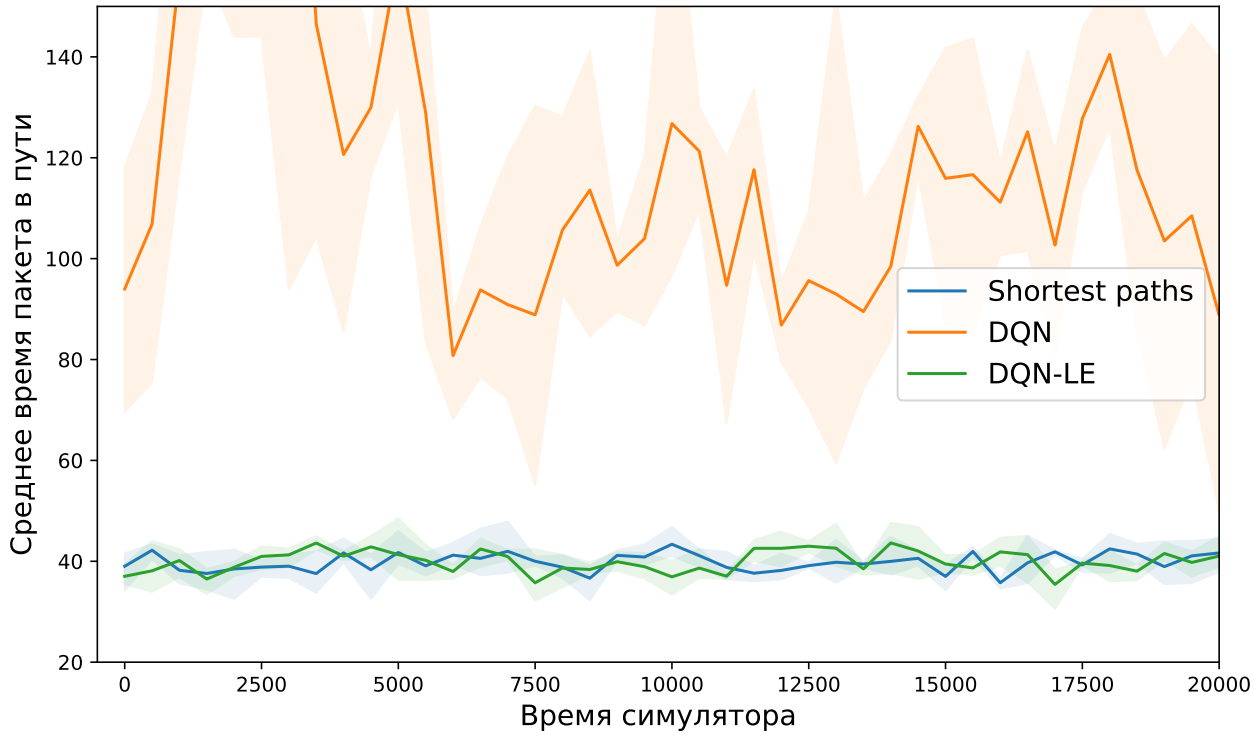


Рисунок 13 – Перенос опыта на новую топологию графа

График 13 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 15a и 15b изображена работа алгоритма DQN-LE-routing, предобученного на базовом графе из 10 вершин, внутри случайного графа из 40 вершин (рис. 14), в условиях низкой и высокой нагрузки. В условиях низкой нагрузки DQN-LE-routing начинает показывать приемлемое качество работы

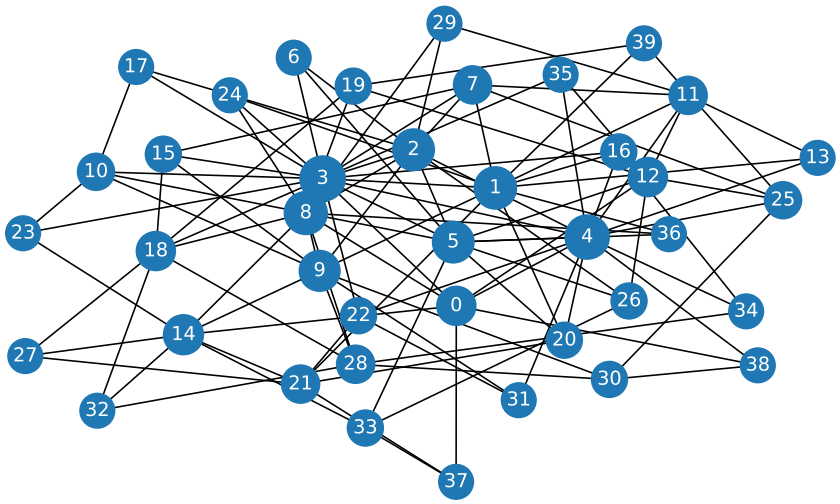
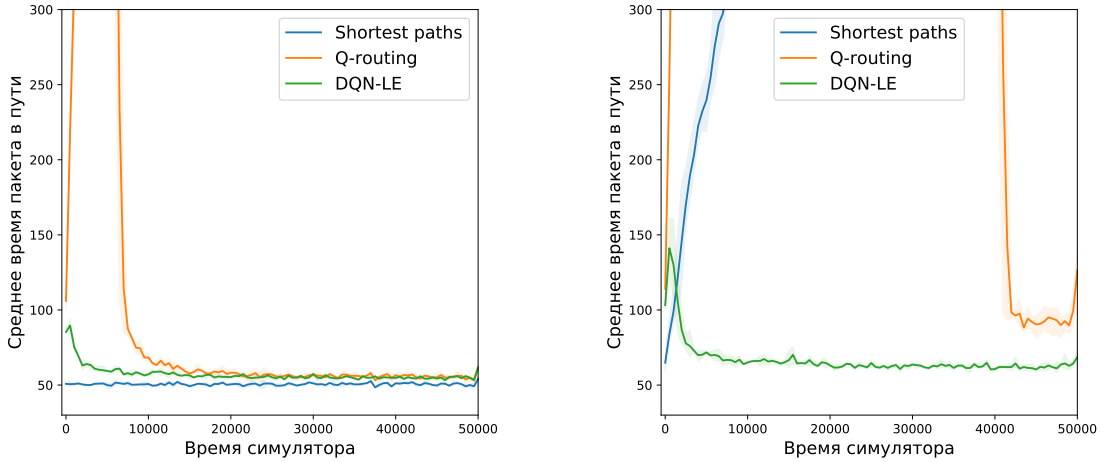


Рисунок 14 – Случайный граф из 40 вершин



(a) Низкая нагрузка (b) Высокая нагрузка

Рисунок 15 – 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.

3.3. Эксперименты в модели конвейерной системы

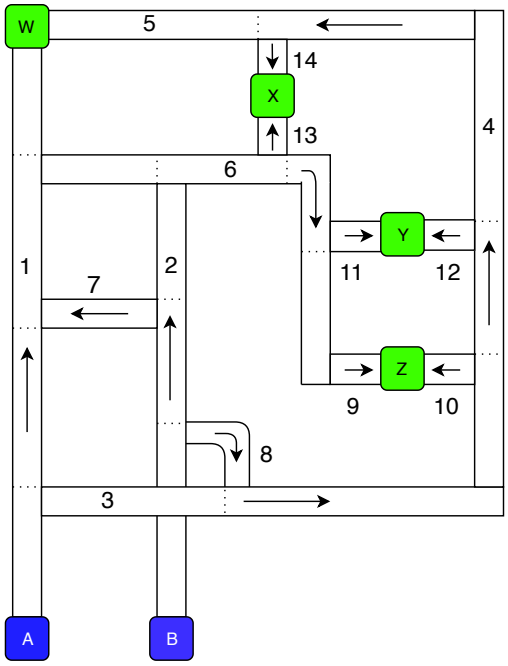


Рисунок 16 – Базовая модель конвейерной системы

При проведении экспериментов в модели конвейерной системы алгоритм DQN-LE-routing сравнивался со следующими бейзлайнами:

— **BSR**: централизованный статический алгоритм, рассчитывающий кратчайшие пути алгоритмом Дейстктры, основываясь на длинах конвейерных секций [31];

— **Vyatkin-Black**: децентрализованный алгоритм управления, описанный в разделе 1.1.1.3;

— **Q-routing**: алгоритм, аналогичный Vyatkin-Black, но использующий Q-routing для маршрутизации сумок. Вознаграждения для Q-routing рассчитываются так же, как и для DQN-LE-routing.

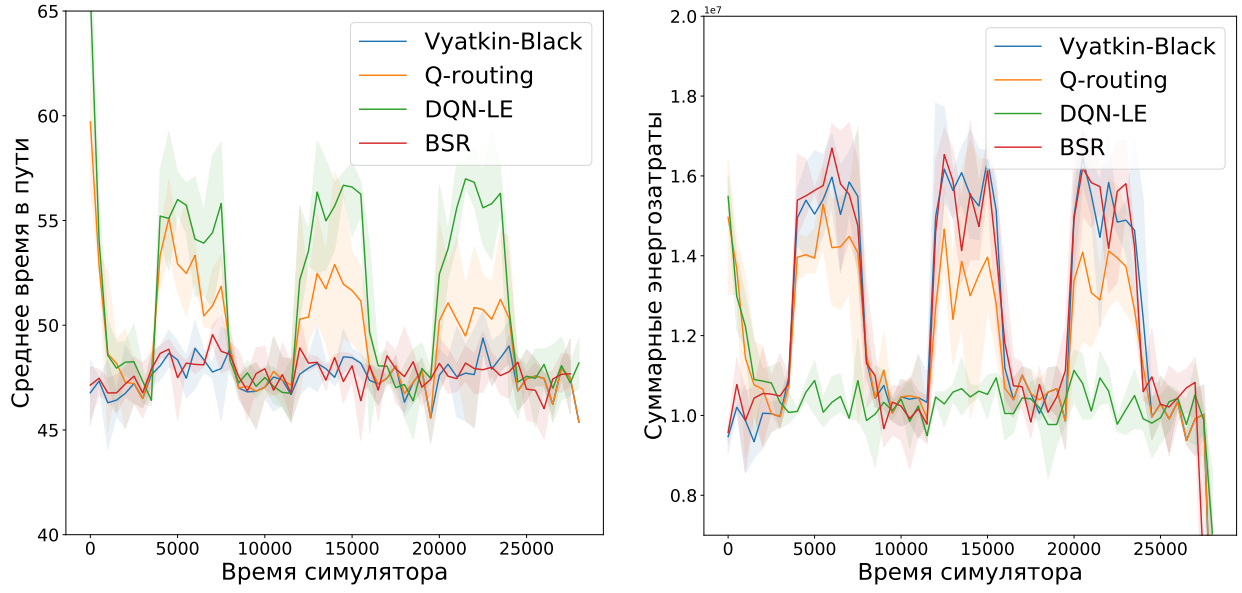
При работе каждого из алгоритмов используется остановка конвейера спустя некоторое время d_t после ухода с него последней сумки, несмотря на то, что алгоритмы Vyatkin-Black и BSR не описывают такого поведения и не решают задачу экономии электроэнергии в целом. Это сделано в целях большей честности сравнения и демонстрации преимуществ DQN-LE-routing перед простым добавлением эвристики с остановкой к существующему решению. Из тех же соображений честности алгоритм BSR дополнен логикой избежания столкновений, несмотря на то, что в оригинальном BSR проблема избежания столкновений в целом не рассматривается. Логика аналогична той, что

описана в Vyatkin-Black и используется в том числе для Q-routing и DQN-LE-routing, но централизована.

Нейросетевые агенты для DQN-LE-routing предварительно обучались на предпосчитанных значениях кратчайших путей от случайных сочленений до выходных вершин, полученных по базовой модели конвейерной сети (рис. 16). Модель имеет 14 конвейеров (всего 26 отдельных секций), 16 сочленений, две входные вершины и четыре выходные вершины. Каждая из секций конвейеров с 1 по 8 включительно имеет длину 10 метров. Конвейеры с 9 по 14 каждый имеют единственную секцию длиной в 3 метра. Максимальная скорость каждого конвейера равна 1 м/с.

3.3.1. Неравномерный поток до выходных вершин

Этот эксперимент проводился в описанной выше базовой модели конвейерной сети (рис. 16). В ней из каждой входной вершины до каждой из выходных вершин существует как минимум два пути — один пролегает через конвейеры 1, 2 и 3, а второй — через конвейеры 3, 4 и 5.



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

Рисунок 17 – Неравномерный поток до выходных вершин

При этом кратчайшие по расстоянию пути от входов до выходов Y и Z пролегают через конвейеры 3 и 4. Однако, с точки зрения экономии энергопотребления задействовать эти конвейеры может быть невыгодно. Так как конвейеры 1 и 2, будучи конвейерами, принимающими сумки с входных вершин,

и так включены всегда, с точки зрения экономии энергии может быть выгоднее отправлять сумки до Y и Z через конвейер 6, не включая конвейеры 8, 3 и 4.

В данном сценарии сумки, поступающие на входы, направляются в разные периоды времени либо в один из выходов W и X, либо в любой из выходов.

На графике 17 можно видеть, что в таких условиях алгоритмы Vyatkin-Black и BSR, учитывающие только расстояния до выходных вершин при маршрутизации, обеспечивают стабильно высокую скорость доставки багажа, но энергопотребление подскакивает всякий раз, когда сумки начинают идти к выходам Y и Z. Это происходит потому что включаются конвейеры 8, 3 и 4. Алгоритм DQN-routing же, напротив, обеспечивает стабильно низкое энергопотребление, но терпит повышение среднего времени доставки чемоданов. Оно происходит потому, что сумки начинают направляться к выходам Y и Z длинным путем через конвейер 6. Алгоритм Q-routing же наблюдает скачки обоих величин, скорее всего, из за периодической разбалансировки оценок Q-функции, приводящего к случайным посылкам сумок вдоль конвейеров 3 и 4.

3.3.2. Адаптация к поломкам конвейеров

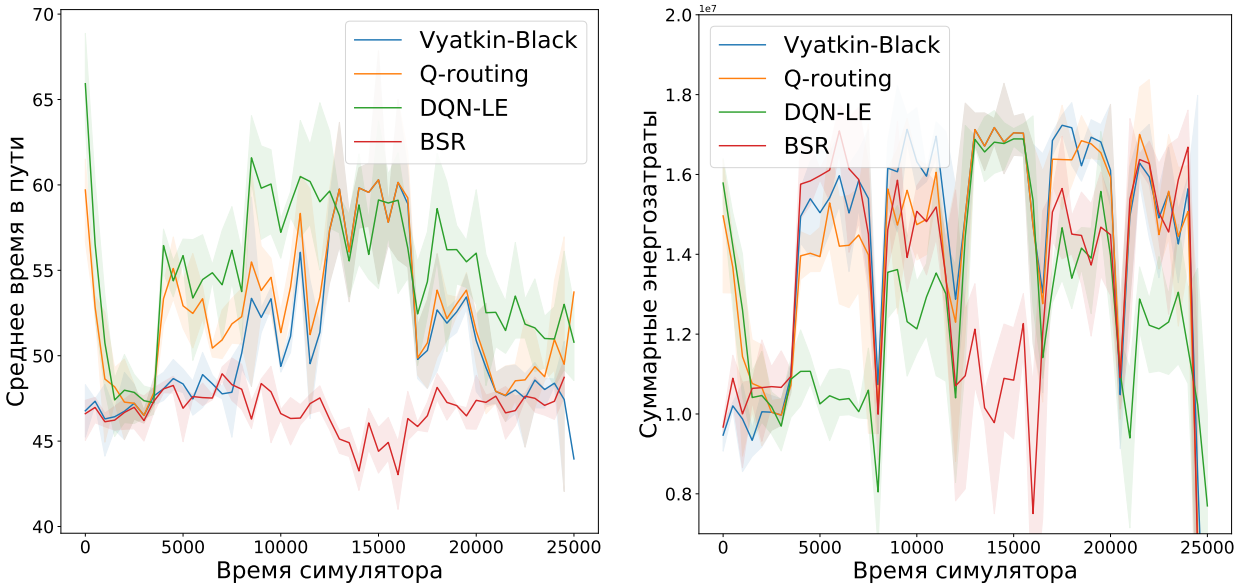
Этот эксперимент также проводился в базовой модели конвейерной сети. В нем сумки все время равномерно идут от всех выходов ко всем входам. Но в определенный момент времени ломается конвейер 7, еще через некоторое время — конвейер 6. Через некоторое время конвейер 7 восстанавливается, и еще через какое-то время восстанавливается конвейер 6.

Таблица 1 – Количество зарегистрированных столкновений сумок

BSR	461
Vyatkin-Black	0
Q-routing	0
DQN-LE	0

Статический алгоритм роутинга BSR не реагирует на поломки конвейеров, поэтому не изменяет свое поведение, когда они случаются (рис. 18). Однако же, как показано в таблице 1, это просто приводит к тому, что сумки направляются на сломанные конвейеры и застревают там, сталкиваясь с другими неподвижными сумками.

Динамические же алгоритмы управления учитывают поломки и перенаправляют новые сумки по обходному пути. Это приводит к отсутствию столк-



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

Рисунок 18 – Адаптация к поломкам конвейеров

новений сумок. Алгоритм же DQN-LE-routing, как видно на графике 18, и в таких условиях обеспечивает наиболее низкое энергопотребление.

Выводы по главе 3

Экспериментальное исследование показало, что алгоритм DQN-LE-routing успешно справляется с задачей маршрутизации. Была продемонстрирована способность алгоритма к адаптации к изменениям среды, таким как изменение интенсивности потока маршрутизируемых объектов и изменение топологии графа.

Также было показано, что DQN-LE-routing, в отличие от предложенного автором ранее алгоритма DQN-routing, способен адаптироваться к не встреченным ранее топологиям графов без дополнительного обучения с учителем. Кроме того, эксперименты в модели конвейерной сети показали, что алгоритм способен эффективно снижать энергопотребление системы, в отличие от существующих опубликованных алгоритмов, применяющихся в управлении системами для транспортировки багажа.

ЗАКЛЮЧЕНИЕ

В данной работе был представлен новый метод управления конвейерной системой, основанный на маршрутизации с помощью самообучающихся нейросетевых агентов — DQN-LE-routing. Для анализа работы алгоритма и его сравнения с существующими аналогами была разработана имитационная модель штучной конвейерной сети.

В ходе экспериментального исследования было установлено, что, в отличие от существующих аналогов, алгоритм способен успешно решать задачу совместной оптимизации энергопотребления и времени доставки объектов. Также было показано, что DQN-LE-routing превосходит по качеству работы предложенный ранее автором алгоритм DQN-routing и обладает большим потенциалом применения в реальных системах.

Кроме того, предложенный алгоритм, будучи model-free алгоритмом обучения с подкреплениям, имеет потенциал применения также и к другим задачам, связанным с маршрутизацией: как минимум — к другим типам автоматических систем для перемещения штучных грузов (например, рельсовым тележкам), и как максимум — к задачам дорожной или сетевой маршрутизации.

По тематике работы в журнале Future Generation Computer Systems была опубликована статья “Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system” [42]. Также, доклад “Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением” по результатам работы был представлен на VIII Конгрессе молодых ученых [1] и получил награду в номинации “Лучший научно-исследовательский доклад”.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Д.В. М.* Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением // Сборник тезисов докладов конгресса молодых ученых. Электронное издание. — 2019.

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

3 *Black G., Vyatkin V.* Intelligent component-based automation of baggage handling systems with IEC 61499 // IEEE Transactions on Automation Science and Engineering. — 2009. — Т. 7, № 2. — С. 337–351.

4 *Мухутдинов Д.* Библиотека DQNRoute. — 2019. — `https://github.com/flyingleafe/dqnroute`.

5 *Ago M., Nishi T., Konishi M.* Simultaneous optimization of storage allocation and routing problems for belt-conveyor transportation // Journal of Advanced Mechanical Design, Systems, and Manufacturing. — 2007. — Vol. 1, no. 2. — P. 250–261.

6 *Åström K. J., Hägglund T., Astrom K. J.* Advanced PID control. Vol. 461. — ISA-The Instrumentation, Systems, Automation Society Research Triangle …, 2006.

7 Asymmetric transitivity preserving graph embedding / M. Ou [et al.] // Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. — ACM. 2016. — P. 1105–1114.

8 *Belkin M., Niyogi P.* Laplacian eigenmaps and spectral techniques for embedding and clustering // Advances in neural information processing systems. — 2002. — P. 585–591.

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

10 *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.

11 *Cai H., Zheng V. W., Chang K. C.-C.* A comprehensive survey of graph embedding: Problems, techniques, and applications // IEEE Transactions on Knowledge and Data Engineering. — 2018. — Vol. 30, no. 9. — P. 1616–1637.

12 *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.

13 *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.

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

15 Design and implementation of intelligent energy efficient conveyor system model based on variable speed drive control and physical modeling / I. A. Halepoto [et al.] // International Journal of Control and Automation. — 2016. — Vol. 9, no. 6. — P. 379–388.

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

17 Continuous conveyors–belt conveyors for loose bulk materials – basis forcalculation and dimensioning : Standard / DIN. — 2002.

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

19 *Fay A., Fischer I.* Decentralized control strategies for transportation systems // 2005 International Conference on Control and Automation. Vol. 2. — IEEE. 2005. — P. 898–903.

20 *Goyal P., Ferrara E.* Graph embedding techniques, applications, and performance: A survey // Knowledge-Based Systems. — 2018. — Vol. 151. — P. 78–94.

21 *Grover A., Leskovec J.* node2vec: Scalable feature learning for networks // Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. — ACM. 2016. — P. 855–864.

22 *Hallenborg K.* Decentralized scheduling of baggage handling using multi-agent technologies // Multiprocessor Scheduling, Theory and Applications. — IntechOpen, 2007.

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

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

25 *Hochreiter S., Schmidhuber J.* Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.

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

27 IEC 61131-3: Programmable controllers - Part 3: Programming languages / I. E. Commission [et al.] // Int. Stand. First Ed. Geneva. — 2013. — Vol. 3.

28 IEC 61499-1: Function Blocks - Part 1: Architecture / I. E. Commission [et al.] // Int. Stand. First Ed. Geneva. — 2015. — Vol. 2.

29 Continuous mechanical handling equipment – belt conveyors withcarrying idlers – calculation of operating power and tensile forces : Standard / ISO. — 1989.

30 Rubber belt conveyors with carrying idlers – calculation of operatingpower and tensile forces : Standard / JIS. — 1992.

31 *Johnstone M., Creighton D., Nahavandi S.* Status-based routing in baggage handling systems: Searching verses learning // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). — 2009. — Vol. 40, no. 2. — P. 189–200.

32 *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.

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

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

35 *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.

36 Machine learning for networking: Workflow, advances and opportunities / M. Wang [et al.] // IEEE Network. — 2018. — Vol. 32, no. 2. — P. 92–99.

37 Mastering the game of Go with deep neural networks and tree search / D. Silver [et al.] // nature. — 2016. — Vol. 529, no. 7587. — P. 484.

38 *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.

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

40 *Middelberg A., Zhang J., Xia X.* An optimal control model for load shifting–with application in the energy management of a colliery // Applied Energy. — 2009. — Vol. 86, no. 7/8. — P. 1266–1273.

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

42 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.

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

44 *Perozzi B., Al-Rfou R., Skiena S.* Deepwalk: Online learning of social representations // Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. — ACM. 2014. — P. 701–710.

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

46 *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.

47 Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning / B. Mao [et al.] // IEEE Transactions on Computers. — 2017. — Vol. 66, no. 11. — P. 1946–1960.

48 *Schäfer G.* Material-flow control for collision avoidance in a conveyor system. — 11 22/2011. — US Patent 8,061,506.

49 *Staniak K., França Jr J.* Energy saving in long distance conveyors-Novel idler technology // Mine planning and equipment selection. Rotterdam, Netherlands: Balkema. — 1996. — P. 479–86.

50 *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.

51 *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.

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

53 *Vickers K., Chinn R.* Passenger terminal baggage handling systems // IEE Colloquium on Systems Engineering of Aerospace Projects (Digest No. 1998/249). — IET. 1998. — P. 6–1.

54 *Wang D., Cui P., Zhu W.* Structural deep network embedding // Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. — ACM. 2016. — P. 1225–1234.

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

56 *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.

57 *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.

58 *Zhang S.*, *Xia X.* Modeling and energy efficiency optimization of belt conveyors // Applied Energy. — 2011. — Vol. 88, no. 9. — P. 3061–3071.