НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2022 Том 22 № 6 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2022 Vol. 22 No 6 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2022-22-6-1187-1196
УДК 004.8+ 65.011.56

Совместное обучение агентов и векторных представлений графов в задаче управления конвейерными лентами

Константин Евгеньевич Рыбкин[1], Андрей Александрович Фильченков[2]✉, Артур Александрович Азаров[3], Алексей Сергеевич Забашта[4], Анатолий Абрамович Шалыто[5]

[1,2,3,4,5] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[3] Северо-Западный институт управления – филиал РАНХиГС, Санкт-Петербург, 199178, Российская Федерация

[1] matveich1903@gmail.com, https://orcid.org/0000-0003-0856-4276
[2] afilchenkov@itmo.ru✉, https://orcid.org/0000-0002-1133-8432
[3] artur-azarov@yandex.ru, https://orcid.org/0000-0003-3240-597X
[4] azabashta@itmo.ru, https://orcid.org/0000-0002-7494-4309
[5] shalyto@mail.ifmo.ru, https://orcid.org/0000-0002-2723-2077

Аннотация

Предмет исследования. Рассмотрена задача маршрутизации системы конвейерных лент на основе мультиагентного подхода. В большинстве данных конвейерных систем багажных лент в аэропортах используются алгоритмы маршрутизации, основанные на ручном моделировании поведения конвейеров. Такой подход плохо масштабируем. Новые исследования в области машинного обучения предлагают решать задачу маршрутизации с помощью обучения с подкреплением. **Метод.** Сформулирован подход к совместному обучению агентов и векторных представлений графа. В рамках подхода предложен алгоритм *QSDNE*, использующий агентов *DQN* и векторные представления *SDNE*. **Основные результаты.** Выполнен сравнительный анализ разработанного алгоритма с алгоритмами мультиагентной маршрутизации без совместного обучения. На основании результатов работы алгоритма *QSDNE* сделан вывод о его эффективности для оптимизации времени доставки и энергопотребления в конвейерных системах. Алгоритм позволил сократить среднее время доставки на 6 % по сравнению с лучшим аналогом. **Практическая значимость.** Разработанный подход может быть использован для решения задач маршрутизации со сложными функциями оценки пути и динамически меняющимися топологиями графов, а предложенный алгоритм — для управления конвейерными лентами в аэропортах и в цеховых производствах.

Ключевые слова

мультиагентное обучение, обучение с подкреплением, адаптивная маршрутизация, конвейерные ленты, графовое представление

Благодарности

Исследование выполнено за счет гранта Российского научного фонда (проект № 20-19-00700).

Ссылка для цитирования: Рыбкин К.Е., Фильченков А.А., Азаров А.А., Забашта А.С., Шалыто А.А. Совместное обучение агентов и векторных представлений графов в задаче управления конвейерными лентами // Научно-технический вестник информационных технологий, механики и оптики. 2022. Т. 22, № 6. С. 1187–1196. doi: 10.17586/2226-1494-2022-22-6-1187-1196

Joint learning of agents and graph embeddings in a conveyor belt control problem

Konstantin E. Rybkin[1], Andrey A. Filchenkov[2]✉, Artur A. Azarov[3], Alexey S. Zabashta[4], Anatoly A. Shalyto[5]

[1,2,3,4,5] ITMO University, Saint Petersburg, 197101, Russian Federation
[3] North-West Institute of Management — branch of the Russian Presidential Academy of National Economy and Public Administration, Saint Petersburg, 199178, Russian Federation

[1] matveich1903@gmail.com, https://orcid.org/0000-0003-0856-4276
[2] afilchenkov@itmo.ru✉, https://orcid.org/0000-0002-1133-8432
[3] artur-azarov@yandex.ru, https://orcid.org/0000-0003-3240-597X
[4] azabashta@itmo.ru, https://orcid.org/0000-0002-7494-4309
[5] shalyto@mail.ifmo.ru, https://orcid.org/0000-0002-2723-2077

Abstract

We focus on the problem of routing a conveyor belts system based on a multi-agent approach. Most of these airport baggage belt conveyor systems use routing algorithms based on manual simulation of conveyor behavior. This approach does not scale well, and new research in machine learning proposes to solve the routing problem using reinforcement learning. To solve this problem, we propose an approach to joint learning of agents and vector representations of a graph. Within this approach, we develop a *QSDNE* algorithm, which uses *DQN* agents and *SDNE* embeddings. A comparative analysis was carried out with multi-agent routing algorithms without joint learning. The results of the *QSDNE* algorithm showed its effectiveness in optimizing the delivery time and energy consumption in conveyor systems as it helped to reduce mean delivery time by 6 %. The proposed approach can be used to solve routing problems with complex path estimation functions and dynamically changing graph topologies, and the proposed algorithm can be used to control conveyor belts at airports and in manufacturing workshops.

Keywords

multi-agent learning, reinforcement learning, adaptive routing, conveyor belt, graph representation

Acknowledgements

The study was supported by a grant from the Russian Science Foundation (project no. 20-19-00700).

For citation: Rybkin K.E., Filchenkov A.A., Azarov A.A., Zabashta A.S., Shalyto A.A. Joint learning of agents and graph embeddings in a conveyor belt control problem. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2022, vol. 22, no. 6, pp. 1187–1196 (in Russian). doi: 10.17586/2226-1494-2022-22-6-1187-1196

Введение

В 2019 году мировой пассажиропоток воздушного транспорта составил 4,5 млрд пассажиров в год [1]. Несмотря на существенное снижение этого показателя, вызванного эпидемией коронавируса, индустрия гражданской авиации быстро развивается. Например, число перевезенных пассажиров в марте 2022 года на внутренних рейсах может даже превысить аналогичный показатель в марте 2019 года [2]. 46 % всех потерь багажа происходит из-за ошибок при его транспортировке, что приводит к многомиллионным убыткам авиакомпаний ежегодно [3].

На сегодняшний день в большинстве таких конвейерных систем багажных лент в аэропортах используются алгоритмы маршрутизации, основанные на ручном моделировании поведения конвейеров на отдельных участках сети [4]. Такой подход плохо масштабируем, так как для каждой конкретной системы приходится заново реализовывать алгоритм. Новые исследования в области машинного обучения предлагают решать задачу маршрутизации с помощью обучения с подкреплением. Основным подходом является мультиагентный [5–7], при котором агенты, помещенные в вершины или транспортируемые объекты, принимают решение о том, куда далее следует перемещать объект. Такие агенты заранее обучаются стратегиям принятия решений в рамках обучения с подкреплением.

Принципиальный вопрос дизайна подобного рода решений — на каком уровне хранится информация о графе. Мультиагентность предполагает, что агенты имеют лишь ограниченное представление о графе. В алгоритме *DQN-LE* [8] предложено использовать векторные представления вершин графа, и, таким образом, агенты обучаются принимать решения на основе выработанного векторного представления. Следовательно, агенты работают с ограниченным представлением графа, которое строится вне зависимости от решаемой задачи.

В настоящей работе рассмотрен вопрос о том, можно ли совместить в рамках машинного обучения и мультиагентного подхода обучение стратегиям принятия решений о транспортировке и представлений графов.

Цель работы — разработка алгоритма обучения векторного представления вершин совместно с мультиагентной системой для решения задачи маршрутизации на конвейерных лентах.

Задача маршрутизации и ее решения

Задача транспортировки грузов. Задачу транспортировки грузов (задачу маршрутизации) можно описать следующим образом: необходимо составить такой оптимальный маршрут грузов из точек их отправления в точки приема таким образом, чтобы затраченное на транспортировку время и/или стоимость были минимальными.

Для решения данной задачи предложено множество подходов, например, симплекс-метод [9] или метод последовательного заполнения матрицы плана (метод северо-западного угла) [10]. Однако наиболее эффективные методы решения на сегодняшний день **основаны на теории графов**.

Существующие алгоритмы маршрутизации, использующие графовое представление, можно разделить на два класса: для статической и динамической маршрутизации. Так как рассматриваемая авторами настоящей работы система конвейеров динамическая (конвейеры могут выходить из строя и останавливаться), то основное внимание уделим **динамическим алгоритмам на графах**.

Можно выделить три основных категории таких алгоритмов. Дистанционно-векторные алгоритмы (*distance-vector*) [11], например алгоритмы в протоколах IGRP (Interior Gateway Routing Protocol) [12] и RIP (Routing Information Protocol) [13], которые основаны на алгоритме Беллмана–Форда [14]. Каждый узел графа хранит вектор расстояний до каждого соседнего узла. Для каждого приходящего объекта высчитывается оптимальный сосед, на основе хранимого в узле вектора. В случае изменения локальной топологии (например, в случае обрыва связи с одним из узлов), информация об этом событии передается всем соседям, а те, в свою очередь, могут распространить ее далее. Данный метод хорошо работает только в небольших сетях, так как при масштабировании сильно увеличивается нагрузка на сеть из-за интенсивного обмена информацией между узлами.

Алгоритмы состояния связи (*link-state*) [15], такие как OLSR (Optimized Link State Routing Protocol) [16], используют алгоритм Дейкстры [17]. В каждом узле сети хранится матрица смежности всего графа со стоимостью путей между узлами. При изменении топологии сети (обрыве или восстановлении связи) информация о событии передается сразу всем узлам. На практике, как правило, используются статические или редко обновляемые таблицы.

Третье семейство образовано алгоритмами на основе обучения с подкреплением [18], первым из которых являлся *Q-routing* [3], где каждая вершина в графе независимый агент, а средой выступает граф. Каждый агент хранит в том или ином виде информацию об оптимальном пути $Q_i(d, n_i)$, где i — индекс агента, d — вершина назначения, n_i — индекс соседа агента i. Значением таких функций Q будет являться оценочная стоимость перемещения объекта до пункта назначения через вершину n_i.

Алгоритм *Q-routing* записывает информацию о значениях Q в таблицы, однако современные алгоритмы [7, 8, 19] аппроксимируют функцию Q при помощи глубоких нейронных сетей. Используем данный подход, так как он показывает лучшие результаты для оптимизации конвейерных лент [8].

Алгоритм DQN-LE. Для корректной работы алгоритма нейросетевого обучения необходимо преобразовать номера вершин графа в некоторое векторное представление, в противном случае сеть может сделать ложный вывод об упорядоченности вершин в соответствии с нумерацией. В работе [8] для алгоритма *DQN-LE* использованы алгоритмы обучения векторных представлений вершин – наиболее эффективно в экспериментах показал себя алгоритм *Laplacian Eigenmaps* (*LE*) [20]. Несмотря на простоту реализации, метод *LE* хорошо работает в системах с меняющимися топологиями.

Приведем описание реализации алгоритма *DQN-LE* [8], использованного в настоящей работе. На вход в алгоритм подадим граф и запустим алгоритм предобучения агентов и графовых представлений. Предобучение важно для качественной работы алгоритмов обучения с подкреплением, иначе сходимость к точке оптимума во время исполнения алгоритма может потребовать очень большого числа итераций. Предобучение произойдет на фиксированном статичном графе. Все агенты предобучаются параллельно с векторными представлениями, причем веса нейронных сетей всех агентов и значения векторных представлений *LE*, обновляются одинаково. Полученные предобученные модели сохраним в файлы для последующего использования на стадии исполнения.

Далее запустим основную часть алгоритма – симуляцию работы конвейеров по заданному сценарию. Есть несколько типов возможных событий: поступление n новых грузов из точек входа с заданным временным интервалом; поломка одного или нескольких конвейеров и их восстановление. Принцип работы агента в основной части опишем следующим образом.

1. В маршрутизатор, к которому привязан агент, приходит груз.
2. Номера вершин, соответствующие текущему агенту, пункту назначения груза и соседям текущего агента, преобразуем в векторное представление с помощью алгоритма *LE*, получая на выходе векторы $\mathbf{v}_i$, $\mathbf{d}$ и $\mathbf{n}_i$ соответственно.
3. Полученные векторы формируют матрицу размера $3k \times m$, где k — размерность векторных представлений; m — число соседей вершины. Сформированная матрица подается на вход нейронной сети, на выходе которой получим вектор размера $m \times 1$ с оценками наилучших соседей для перенаправления груза. С помощью функции *Softmax* [21] выберем наилучшего соседа.
4. Получившееся взаимодействие агента со средой (тройка: состояние–действие–награда) занесем во внутреннюю память агента и выберем некоторое случайное подмножество произошедших событий для его дообучения.
5. Выполним обновление весов нейронной сети агента, в качестве награды передадим предсказанные значения вероятности перехода в соседнюю вершину. Память агента имеет некоторый фиксированный объем, таким образом дообучение проведем только на недавно произошедших событиях.

Одновременно с работой агента происходит дообучение связанного с ним векторного представления. При изменении топологии графа собственные векторы матрицы *LE* необходимо пересчитать заново с помощью алгоритма *DQN-LE*. Таким образом, дообучение агента и векторного представления на этапе симуляции никак не взаимосвязано.

Алгоритм *QSNDE*

Совместное обучение агентов и векторного представления. В алгоритме *DQN-LE* векторное представление графа обучается отдельно и независимо от обу-

чения агентов, осуществляющих маршрутизацию по этому графу. Выбранный алгоритм векторного представления *LE* подбирает отображение из множества вершин в векторное пространство на основе теоретико-информационных принципов без влияния задачи, которая решается агентами. Агенты вынуждены работать с таким представлением.

Сформулируем подход совместного обучения агентов и векторного представления, в котором нейронные сети, осуществляющие векторное представление графов, должны обучаться по ошибкам агентов, выполняющих маршрутизацию на основе используемых ими представлений. Такой подход может учесть в векторном представлении сложную информацию из недекомпозируемых функций стоимости.

Более формально, пусть Γ — множество графов; Embeddings — семейство отображений E из $v \in V$ множества вершин графа $\mathbf{G}$, $\mathbf{G} \in \Gamma$ в некоторое векторное пространство, элементом этого семейства является алгоритм *LE*. Введем также параметрическое семейство $\{Q_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\Theta}$ аппроксимаций стратегий агентов в мультиагентном алгоритме маршрутизации MULTIAGENT, причем каждый элемент семейства $Q^i_{\boldsymbol{\theta}}$ принимает на вход векторное представление соответствующей агенту вершины i.

Для графа $\mathbf{G} \in \Gamma$ и заданной меры ошибки алгоритма маршрутизации $\mathcal{L}$ задача обучения агентов маршрутизации имеет вид

$$\arg \min_{\{\boldsymbol{\theta}_i\},\boldsymbol{\theta}_i\in\Theta} \mathcal{L}(\text{MULTIAGENT}((Q^i_{\boldsymbol{\theta}_i}(E(v_i)))_{i=1\ldots|V|})),$$

и E выбирается независимо от $\mathcal{L}$.

Выделим параметрическое семейство отображений $\{E_{\boldsymbol{\psi}}\}_{\boldsymbol{\psi}\in\Psi}$. *Задача совместного обучения агентов и векторного представления* ставится следующим образом:

$$\arg \min_{\substack{\{\boldsymbol{\theta}_i\},\boldsymbol{\psi}\\ \boldsymbol{\theta}_i\in\Theta,\boldsymbol{\psi}\in\Psi}} \mathcal{L}(\text{MULTIAGENT}((Q^i_{\boldsymbol{\theta}_i}(E_{\boldsymbol{\psi}}(v_i)))_{i=1\ldots|V|})).$$

Поскольку Q принимает результат работы E в качестве аргумента, решение такой задачи может быть найдено за счет выбора модели (глубокой нейронной сети) для E и разработки алгоритма совместного обучения Q и E.

Выбор модели для обучаемого векторного представления. Выделим три основные группы алгоритмов представлений вершин в динамических графах [22]:

1) алгоритмы матричной факторизации: *LE* [20], *HOPE* [23];
2) алгоритмы на основе случайного обхода вершин (*random walk*): *node2vec* [24];
3) алгоритмы глубокого обучения: на основе автокодировщиков, *Structural Deep Network Embedding* (*SDNE*) [25]; с использованием рекуррентных слоев, *dyngraph2vec* [26] и DGNN [27]; на основе графовых нейронных сетей [28].

Несмотря на применимость всех перечисленных алгоритмов к решению задачи векторного представления вершин в конвейерной системе, именно алгоритм *SDNE* взят за основу для реализации совместного обучения векторных представлений и агентов. Выбор основан на неспособности других алгоритмов эффективно обмениваться информацией о соседях с агентом на этапе дообучения без существенной перестройки архитектуры исходного алгоритма. В свою очередь, в основе алгоритма *SDNE* лежит нейронная сеть, что позволит ее дообучить на информации от агента с помощью проталкивания градиента. Отметим, что несмотря на то, что в основе методов с *LSTM* также лежат нейронные сети, их применение к задаче маршрутизации имеет мало практического смысла, поскольку изменения топологии (поломка/восстановление конвейерной ленты) происходит не столь часто. Использование графовых нейронных сетей затратно по времени и не подходит для адаптивной маршрутизации. Приведем описание алгоритма *SDNE*.

Будем называть вершины близкими друг другу относительно метрики близости первого порядка, если они смежные. Соответственно, близостью второго порядка для пары вершин является наличие общих вершин в каждом из двух множеств их соседей больше некоторого заданного значения. Алгоритм *SDNE* преобразует вершины графа в некоторое векторное пространство, сохраняя близость вершин относительно метрик близости первого и второго порядков. Обучение нахождению близости первого порядка является задачей обучения с учителем, поскольку имеется информация о ребрах между вершинами. С другой стороны, для обучения близости второго порядка правильных меток нет, и модель должна уметь сама находить закономерности. Таким образом, получим полууправляемую модель обучения алгоритма.

Архитектура нейронной сети представляет автокодировщик с одним или несколькими внутренними слоями (рис. 1).

Задача автокодировщика — минимизировать следующую функцию ошибки:

$$\mathcal{L}_{\text{mix}} = \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{\text{reg}} =$$

$$= \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \odot b_i + \alpha \sum_{i,j=1}^{n} s_{i,j}\|E(\mathbf{x}_i) - E(\mathbf{x}_j)\|_2^2 +$$

$$+ \nu\frac{1}{2}\sum_{k=1}^{K} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2,$$

где $\mathbf{x}_i$ — вектор смежности вершины i; $b_i = 1$, если $s_{i,j} = 0$, иначе $b_i = \beta > 1$; $s \in \mathbf{G}_{n,n}$ — элемент матрицы смежности $\mathbf{G}_{n,n}$; $E(\mathbf{x}_i)$ — векторное представление для i-ой вершины; $\mathcal{L}_{\text{reg}}$ — функция L2-регуляризации, используемой для предотвращения переобучения; α и ν — коэффициенты функций регуляризации.

Алгоритм совместного обучения сети агента и сети векторного представления графа. Зафиксируем веса декодировщика сети и дообучим кодировщик на текущей топологии графа, подменив градиент на выходном слое кодировщика на градиент, полученный от агента, на основе которого при помощи обратного спуска обновим веса. Таким образом, кодировщик будет дообучен на актуальном состоянии своего агента. После распространения градиента обнулим память векторного представления до следующего вызова процедуры.

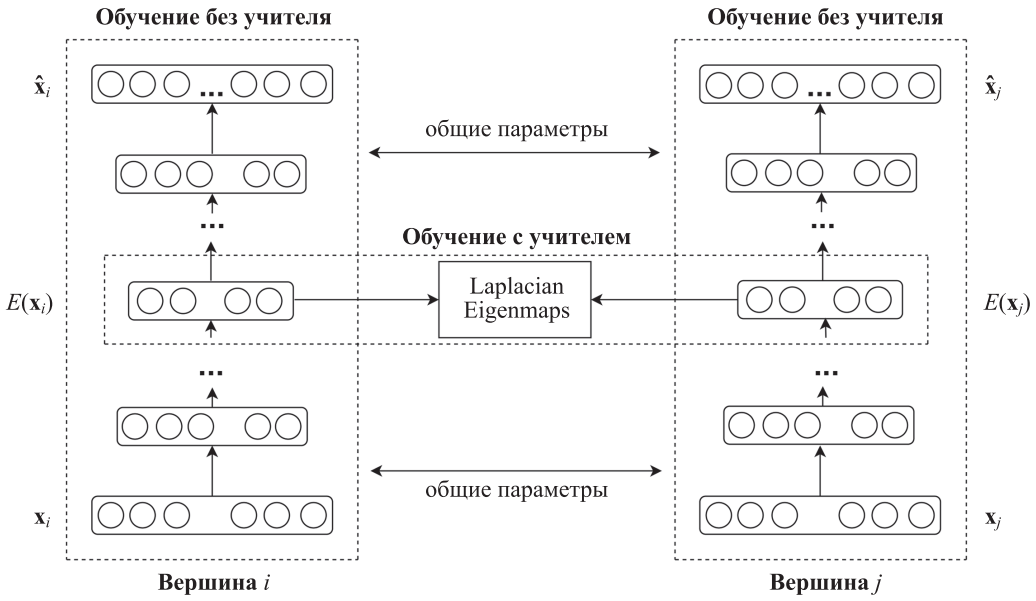

Рис. 1. Архитектура *SDNE* [25]
Fig. 1. *SDNE* architecture [25]

Опишем формально, как происходит распространение градиента от сети агента $Q^i_{\boldsymbol{\theta}}$ к сети векторного представления $E_{\boldsymbol{\psi}}$. На шаге дообучения агента Q^i вычислим значение функции потерь $\mathcal{L}(Q^i_{\boldsymbol{\theta}})$. С помощью метода обратного распространения ошибки получим значение градиента $\nabla_{\boldsymbol{\theta}} = \partial\mathcal{L}(Q^i_{\boldsymbol{\theta}})/\partial\boldsymbol{\theta}$. Опуская шаги пересчета параметров во внутренних слоях, выпишем итоговый вектор градиентов:

$$\begin{bmatrix} \dfrac{\partial Loss_i}{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_i)} \\ \dfrac{\partial Loss_i}{\partial E_{\boldsymbol{\psi}}(\mathbf{x}^d)} \\ \dfrac{\partial Loss_i}{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_j^i)} \end{bmatrix},$$

$$Loss_i = \mathcal{L}(Q^i_{\boldsymbol{\theta}}(E_{\boldsymbol{\psi}}(\mathbf{x}_i), E_{\boldsymbol{\psi}}(\mathbf{x}^d), E_{\boldsymbol{\psi}}(\mathbf{x}_j^i))),$$

где $\mathbf{x}_i$ — вектор смежности для текущей вершины i; $\mathbf{x}^d$ — вектор смежности для вершины точки назначения; $\mathbf{x}_j^i$ — вектор для j-го соседа текущей вершины. Обновим соответствующие строки матрицы градиентов:

$$\nabla\mathbf{x}_i[j] \leftarrow \nabla\mathbf{x}_i[j] + \frac{\partial Loss_i}{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_i)}.$$

Изменим параметры кодировщика, используя обратное распространение градиента ошибки и вычислим градиент:

$$\nabla\boldsymbol{\psi} = \frac{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_i)}{\partial\boldsymbol{\psi}}.$$

В данном случае вместо значения функции потерь на выходной слой кодировщика подадим вектор $1 \times k$, где k — размерность векторного представления. В результате после обратного распространения градиента сквозь скрытые слои получим:

$$\nabla E_{\boldsymbol{\psi}}(\mathbf{v}_i) \leftarrow \frac{\partial E_{\boldsymbol{\psi}}(\mathbf{v}_i)}{\partial\mathbf{v}_i}.$$

Рассчитаем градиент по $\mathbf{G}_{n,n}$:

$$\frac{\partial L(Q^i_{\boldsymbol{\theta}})}{\partial\mathbf{G}_{n,n}} \underset{E,A}{\otimes} \frac{\partial Loss_i}{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_i)} \times \frac{\partial E_{\boldsymbol{\psi}}(\mathbf{x}_i)}{\partial\mathbf{x}_i},$$

где $\otimes$ — произведение Кронекера.

Архитектура и описание алгоритма *QSDNE*. На вход алгоритма подадим матрицу смежности графа $\mathbf{G}$, $\mathbf{G}_{n,n}$. Архитектура *QSDNE* содержит архитектуру агента, совпадающую с алгоритмом *DQN-LE*, и архитектуру векторного представления графа, совпадающую с *SDNE* (рис. 2).

Этап предобучения *QSDNE* полностью совпадает с предобучением в *DQN-LE*.

На этапе дообучения используем алгоритм совместного обучения весов сетей агента и векторного представления графа. Для этого введем внутреннюю память векторных представлений $\mathbf{M}_{n,k}$, где n — число вершин графа. На шаге дообучения агента Q запишем в память получившийся частичный градиент по входным переменным, который имеет вид (i_{grad}, d_{grad}, $n_{i,grad}$). Добавим градиенты к соответствующим строкам $\mathbf{M}_{n,k}$. При совершении заданного числа итераций дообучения агента, запускается процедура обратного распространения градиента по сети *SDNE*.

При изменении топологии графа векторные представления дообучаются так же, как и в *DQN-LE*, однако скорость дообучения декодировщика будет выше, чем у кодировщика. Это необходимо по той причине, что веса кодировщика дополнительно обновлялись вместе с обновлением весов сети агента, поэтому для лучшей сходимости декодировщик должен дообучаться быстрее.

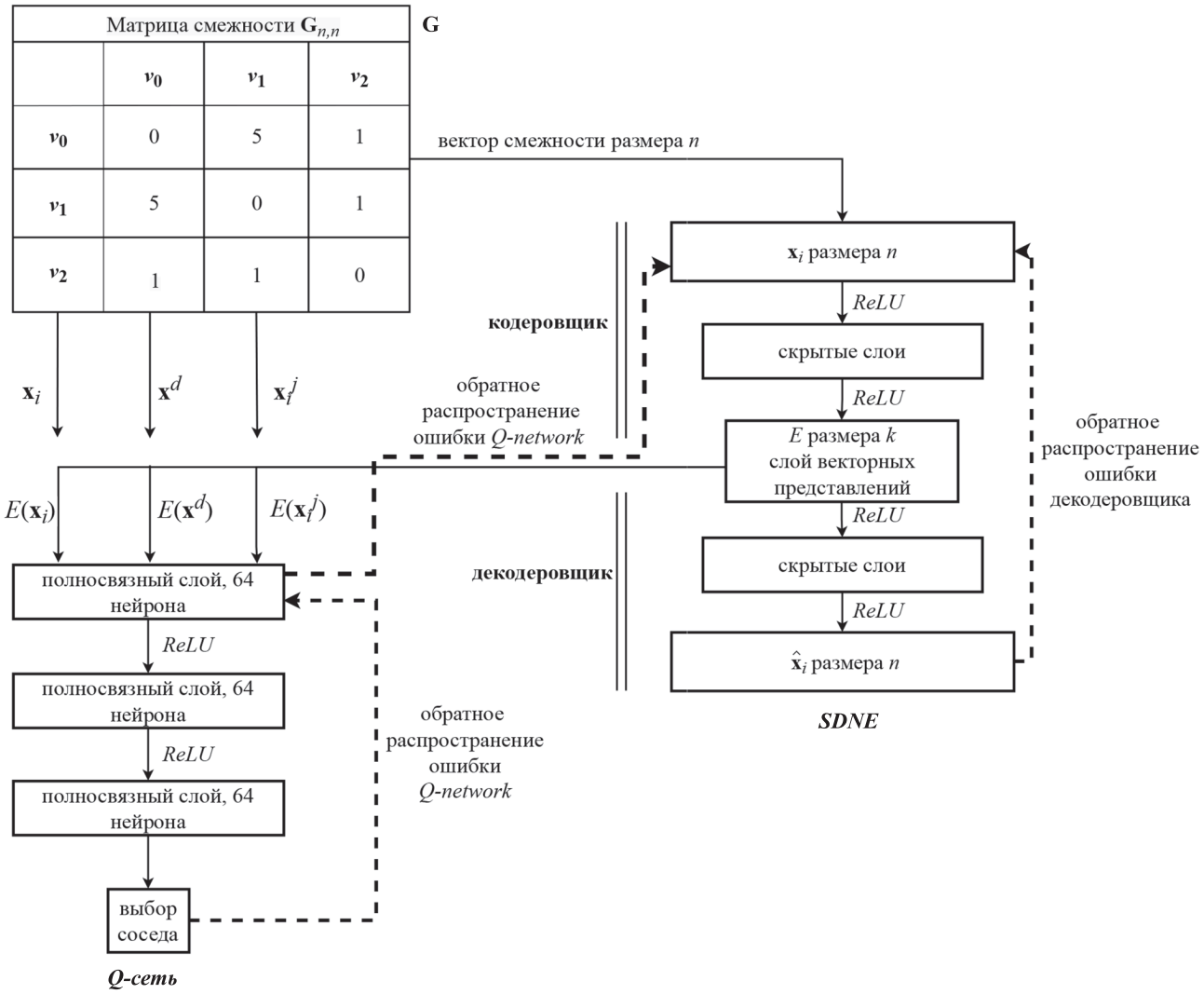

Рис. 2. Архитектура *QSDNE*.

G — граф, подаваемый на вход; *SDNE* — архитектура одного блока автокодировщика (рис. 1); *Q-сеть* — архитектура агента *DQN*

Fig. 2. *QSDNE* architecture

G is input graph. *SDNE* is the architecture of one *SDNE* autoencoder block (Fig. 1).

Q-network is the architecture of an agent's *Q-network* (*DQN*)

Экспериментальное исследование алгоритма *QSDNE*

Симуляционная среда для тестирования. Тестирование алгоритма выполним в симуляционной среде *DQN-routing*, что мотивировано рядом причин. Во-первых, среда позволяет симулировать работу конвейеров максимально приближенно к реальным условиям. Во-вторых, среда дает возможность выполнить сравнение в удобном виде с существующими подходами к обучению векторных представлений.

Процесс симуляции работы конвейерных систем предназначен для работы с дискретно-событийными моделями — системой конвейеров для транспортировки грузов. Приоритетная задача в работе конвейеров — предотвращение столкновений багажа. Принимающие маршрутизаторы исходят из первоочередной задачи оптимизации числа столкновений (в идеале равному нулю).

Симуляционная среда позволяет рассчитывать основные метрики эффективности доставки по конвейерной ленте: время доставки груза, измеряемое в секундах, и затрачиваемую на это электроэнергию, измеряемую в условных единицах симуляционной системы. Последняя метрика является недекомпозируемой, поскольку учитывает затраты на перемещение конвейерной ленты, на которой могут одновременно находиться сразу несколько перемещаемых объектов. Также система учитывает число столкновений перемещаемых объектов.

Поместим агенты в маршрутизаторы, которые стоят в точках сочленений конвейеров и отвечают за изменение пути груза до пункта назначения. Места маршрутизаторов в конвейере зададим расстоянием от начала конвейера до точки сочленения. Выполним преобразование конвейерной системы в граф, при этом маршрутизаторы и пункты отправки/приема грузов являются

вершинами графа, а части конвейеров — его ребрами. Пример такого графа представлен на рис. 3.

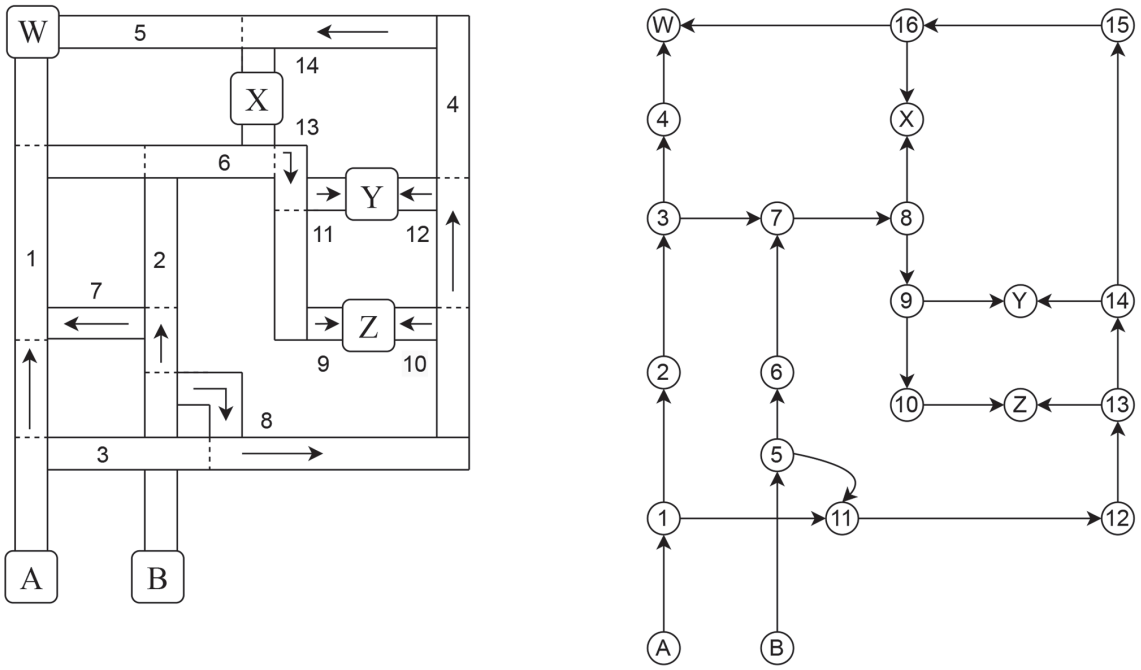

Рис. 3. Пример конвейерной системы и ее представление в виде графа [7].
A и B — начало конвейера; W, X, Y, Z — точки сочленения

Fig. 3. An example of a conveyor system and its representation as a graph [7], A and B and conveyer entry points, W, X, Y, Z are diverters

Технические особенности реализации. Представим разработанный модуль генерации данных для предобучения модели *QSDNE*, который собирает данные для предобучения на заданном графе. Генератор находит индексы ненулевых элементов в матрице смежности и выдает на их основе список пар соответствующих им вершин. Полученные вершины передаются на вход алгоритма *QSDNE*.

Перед началом сравнения произведем отбор гиперпараметров модели *QSDNE* с использованием библиотеки *hyperopt* [29]. Отбор выполним на стадии предобучения модели без использования реальных сценариев поведения. Число итераций метода *QSDNE* примем равным 500, при этом оптимальный коэффициент обучения — 0,02. Кроме гиперпараметров, подберем оптимальное число слоев в автокодировщике. Автокодировщик имеет по одному скрытому слою в 15 нейронов каждый в кодировщике и декодировщике. Размер выходного векторного представления равен 10. На этапе предобучения кодировщик и декодировщик обучены с одинаковой скоростью.

Сравнение *QSDNE* с другими алгоритмами. Так как применение векторных представлений неразрывно связано с оптимизацией решения задачи маршрутизации, то и сравнение алгоритмов векторного представления вершин произведем в контексте оптимизации метрик работы симуляционной системы. Выполним сравнение по следующим метрикам: среднее время доставки груза, среднее энергопотребление всей системы и число столкновений сумок. Для уменьшения дисперсии произведем по 20 запусков на каждом сценарии.

Проведем сравнение четырех алгоритмов: *LE* [20], *node2vec* [24], *SDNE* [25] и *QSDNE*. Отметим, что векторные представления дообучались только в случае алгоритма *QSDNE*, а алгоритм *LE* соответствует *DQN-LE*.

Рассмотрим два сценария записи системы. Первый сценарий (базовый) предполагает две поломки разных конвейеров и последующее их восстановление. Во втором сценарии (усложненном) увеличена интенсивность изменения топологии графа: при меньшем количестве загружаемого на ленты багажа, события поломки и восстановления конвейеров стали происходить чаще. В этом сценарии тестировалось, как эффективно алгоритм *QSDNE* способен приспосабливаться к изменениям окружения по сравнению с остальными подходами. Результаты работы алгоритмов представлены в таблице и на рис. 4. Полученные результаты по среднему времени доставки статистически значимы при параметре p-value, который равен 10^{-6}.

На рис. 4, *a* видно, что алгоритм *QSDNE* большую часть работы имеет лучшие результаты, чем реализация *LE*. Из рис. 4, *b* следует, что график *QSDNE* ведет себя ровнее, а значит, быстрее происходит адаптация к изменениям в графе. Несмотря на то, что результаты не являются статистически значимыми, они позволяют говорить, что *QSDNE* не менее эффективен на данном сценарии, чем метод *LE*. Тем не менее, с учетом вышеизложенного, его использование может быть более востребовано на реальных примерах.

Таблица. Сравнение времени доставки в базовом и усложненном сценариях
Table. Comparison of delivery time in basic and advanced scenarios

Алгоритм векторного представления графа	Среднее время доставки, с		Минимальное время доставки, с		Максимальное время доставки, с		Число столкновений
	базовый	усложненный	базовый	усложненный	базовый	усложненный	
LE	53,49	53,04	40,56	40,46	**74,64**	**74,73**	0
node2vec	57,67	58,06	41,19	41,24	78,89	79,77	0
SDNE	59,19	59,99	41,22	**41,21**	79,68	80,88	0
QSDNE	**50,49**	**52,63**	**39,43**	41,29	75,75	76,97	0

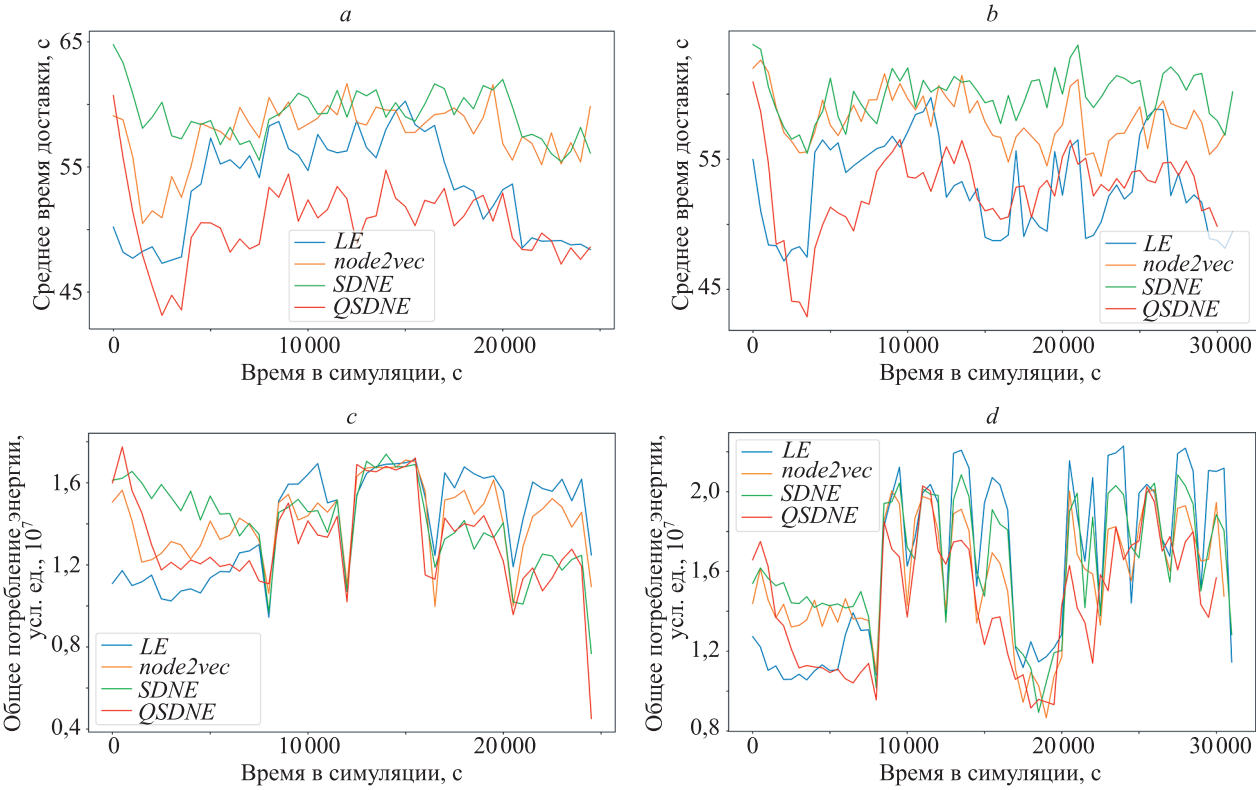

Рис. 4. Сравнение *QSDNE* с другими алгоритмами на базовом и усложненном сценариях по параметрам: время доставки (*a*, *b*) и энергия (*c*, *d*)

Fig. 4. Comparison of *QSDNE* with other algorithms in the basic and advanced scenarios in terms of parameters: delivery time (*a*, *b*) and energy (*c*, *d*)

Отдельно рассмотрим графики метрики — энергопотребления (рис. 4, *c*, *d*). Можно заметить, что *QSDNE* эффективнее алгоритма *LE* и стандартной реализации *SDNE*.

Несмотря на то, что полученные результаты по усложненному сценарию для *QSDNE* не показали существенную разницу в энергопотреблении с алгоритмом *node2vec*, нельзя сказать, что *node2vec* является лучшим для использования. Это доказано результатами статистического теста (по энергопотреблению *node2vec* не обходит статистически *QSDNE*) и с помощью сравнения алгоритмов по среднему времени доставки. При этом алгоритм *node2vec* показал один из худших результатов, тогда как *QSDNE* уменьшил среднее время доставки на 6 % по сравнению с лучшим результатом, который был достигнут алгоритмом *LE*.

Заключение

В работе предложен подход совместного обучения агентов и векторных представлений графа в парадигме мультиагентной маршрутизации на основе глубокого обучения с подкреплением, и предложен алгоритм *QSDNE*, реализующий этот подход для решения задачи маршрутизации в системе конвейерных лент.

Результаты работы алгоритма *QSDNE* позволили говорить о его эффективности для оптимизации времени доставки и энергопотребления в конвейерных системах.

Разработанный подход может быть использован для решения задач маршрутизации со сложными функциями оценки пути и динамически меняющимися топологиями графов. Предложенный алгоритм может применяться для управления конвейерными лентами в аэропортах и в цеховых производствах.

Литература

1. The World of Air Transport in 2019. ICAO Annual Report, 2019 [Электронный ресурс]. URL: https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx (дата обращения: 01.10.2022).
2. COVID-19 Air Travel Recovery, OAG, 2022 [Электронный ресурс]. URL: https://www.oag.com/coronavirus-airline-schedules-data. (дата обращения: 01.10.2022).
3. 2019 SITA Baggage IT Insights report, SITA, 2019 [Электронный ресурс]. URL: https://www.sita.aero/resources/surveys-reports/baggage-it-insights-2019. (дата обращения: 06.09.2022)
4. Sørensen R.A., Nielsen M., Karstoft H. Routing in congested baggage handling systems using deep reinforcement learning // Integrated Computer-Aided Engineering. 2020. V. 27. N 2. P. 139–152. https://doi.org/10.3233/ICA-190613
5. Kang Y., Wang X., Lan Z. Q-adaptive: A multi-agent reinforcement learning based routing on dragonfly network // Proc. of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC). 2021. P. 189–200. https://doi.org/10.1145/3431379.3460650
6. Choi S., Yeung D.Y. Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control // Advances in Neural Information Processing Systems. 1995. V. 8. P. 945–951.
7. Mukhutdinov D., Filchenkov A., Shalyto A., Vyatkin V. Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system // Future Generation Computer Systems. 2019. V. 94. P. 587–600. https://doi.org/10.1016/j.future.2018.12.037
8. Мухудинов Д. Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением: магистерская диссертация. СПб.: Университет ИТМО, 2019, 92 с. [Электронный ресурс]. URL: http://is.ifmo.ru/diploma-theses/2019/2_545846477102619143O.pdf (дата обращения: 01.10.2022).
9. Dantzig G.B. Origins of the simplex method // A history of scientific computing. 1990. P. 141–151. https://doi.org/10.1145/87252.88081
10. Кузнецов А.В., Холод Н.И., Костевич Л.С. Руководство к решению задач по математическому программированию. Минск: Вышэйшая школа, 1978. 256 с.
11. Vutukury S., Garcia-Luna-Aceves J.J. MDVA: A distance-vector multipath routing protocol // Proc. of the 20th Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM). 2001. V. 1. P. 557–564. https://doi.org/10.1109/INFCOM.2001.916780
12. Albrightson R., Garcia-Luna-Aceves J.J., Boyle J. EIGRP — A fast routing protocol based on distance vectors. 1994.
13. Verma A., Bhardwaj N. A review on routing information protocol (RIP) and open shortest path first (OSPF) routing protocol // International Journal of Future Generation Communication and Networking. 2016. V. 9. N 4. P. 161–170. https://doi.org/10.14257/ijfgcn.2016.9.4.13
14. Bang-Jensen J., Gutin G.Z. Digraphs: Theory, Algorithms and Applications. Springer Science & Business Media, 2009. 795 p.
15. Clausen T., Jacquet P. Optimized link state routing protocol (OLSR). 2003. N RFC3626. https://doi.org/10.17487/RFC3626
16. Jacquet P., Muhlethaler P., Clausen T., Laouiti A., Qayyum A., Viennot L. Optimized link state routing protocol for ad hoc networks // Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. 2001. P. 62–68. https://doi.org/10.1109/INMIC.2001.995315
17. Noto M., Sato H. A method for the shortest path search by extended Dijkstra algorithm // Smc 2000 conference proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. 2000. V. 3. P. 2316–2320. https://doi.org/10.1109/icsmc.2000.886462
18. Mammeri Z. Reinforcement learning based routing in networks: Review and classification of approaches // IEEE Access. 2019. V. 7. P. 55916–55950. https://doi.org/10.1109/ACCESS.2019.2913776
19. Yao Z., Wang Y., Qiu X. DQN-based energy-efficient routing algorithm in software-defined data centers // International Journal of Distributed Sensor Networks. 2020. V. 16. N 6. https://doi.org/10.1177/1550147720935775
20. Belkin M., Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering // Advances in Neural Information Processing Systems. 2002.
21. Gao B., Pavel L. On the properties of the softmax function with application in game theory and reinforcement learning // arXiv. 2017. arXiv:1704.00805. https://doi.org/10.48550/arXiv.1704.00805

References

1. *The World of Air Transport in 2019. ICAO Annual Report, 2019*. Available at: https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx (accessed: 01.10.2022).
2. *COVID-19 Air Travel Recovery, OAG, 2022*. Available at: https://www.oag.com/coronavirus-airline-schedules-data. (sccessed: 01.10.2022).
3. *2019 SITA Baggage IT Insights report, SITA, 2019*. Available at: https://www.sita.aero/resources/surveys-reports/baggage-it-insights-2019. (accessed: 06.09.2022)
4. Sørensen R.A., Nielsen M., Karstoft H. Routing in congested baggage handling systems using deep reinforcement learning. *Integrated Computer-Aided Engineering*, 2020, vol. 27, no. 2, pp. 139–152. https://doi.org/10.3233/ICA-190613
5. Kang Y., Wang X., Lan Z. Q-adaptive: A multi-agent reinforcement learning based routing on dragonfly network. *Proc. of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2021, pp. 189–200. https://doi.org/10.1145/3431379.3460650
6. Choi S., Yeung D.Y. Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control. *Advances in Neural Information Processing Systems*, 1995, vol. 8, pp. 945–951.
7. Mukhutdinov D., Filchenkov A., Shalyto A., Vyatkin V. Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system. *Future Generation Computer Systems*, 2019, vol. 94, pp. 587–600. https://doi.org/10.1016/j.future.2018.12.037
8. Mukhudinov D. *Decentralized conveyor system control algorithm using multi-agent reinforcement learning methods. MSc Dissertation*. St. Petersburg, ITMO University, 2019, 92 p. Available at: http://is.ifmo.ru/diploma-theses/2019/2_545846477102619143O.pdf (accessed: 01.10.2022). (in Russian)
9. Dantzig G.B. Origins of the simplex method. *A history of scientific computing*, 1990, pp. 141–151. https://doi.org/10.1145/87252.88081
10. Kuznetcov A.V., Kholod N.I., Kostevich L.S. *Guide to Solving Mathematical Programming Problems*. Minsk, Vyshjejshaja shkola Publ., 1978, 256 p. (in Russian)
11. Vutukury S., Garcia-Luna-Aceves J.J. MDVA: A distance-vector multipath routing protocol. *Proc. of the 20th Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM)*, 2001, vol. 1, pp. 557–564. https://doi.org/10.1109/INFCOM.2001.916780
12. Albrightson R., Garcia-Luna-Aceves J.J., Boyle J. *EIGRP — A fast routing protocol based on distance vectors*, 1994.
13. Verma A., Bhardwaj N. A review on routing information protocol (RIP) and open shortest path first (OSPF) routing protocol. *International Journal of Future Generation Communication and Networking*, 2016, vol. 9, no. 4, pp. 161–170. https://doi.org/10.14257/ijfgcn.2016.9.4.13
14. Bang-Jensen J., Gutin G.Z. *Digraphs: Theory, Algorithms and Applications*. Springer Science & Business Media, 2009, 795 p.
15. Clausen T., Jacquet P. *Optimized link state routing protocol (OLSR)*. 2003, no. RFC3626. https://doi.org/10.17487/RFC3626
16. Jacquet P., Muhlethaler P., Clausen T., Laouiti A., Qayyum A., Viennot L. Optimized link state routing protocol for ad hoc networks. Proceedings. *IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century*, 2001, pp. 62–68. https://doi.org/10.1109/INMIC.2001.995315
17. Noto M., Sato H. A method for the shortest path search by extended Dijkstra algorithm. Smc 2000 conference proceedings. *2000 IEEE International Conference on Systems, Man and Cybernetics*, 2000, vol. 3, pp. 2316–2320. https://doi.org/10.1109/icsmc.2000.886462
18. Mammeri Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access*, 2019, vol. 7, pp. 55916–55950. https://doi.org/10.1109/ACCESS.2019.2913776
19. Yao Z., Wang Y., Qiu X. DQN-based energy-efficient routing algorithm in software-defined data centers. *International Journal of Distributed Sensor Networks*, 2020, vol. 16, no. 6. https://doi.org/10.1177/1550147720935775
20. Belkin M., Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 2002.
21. Gao B., Pavel L. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv*, 2017, arXiv:1704.00805. https://doi.org/10.48550/arXiv.1704.00805

22. Barros C.D., Mendonça M.R., Vieira A.B., Ziviani A. A survey on embedding dynamic graphs // ACM Computing Surveys. 2021. V. 55. N 1. P. 1–37. https://doi.org/10.1145/3483595
23. Ou M., Cui P., Pei J., Zhang Z., Zhu W. Asymmetric transitivity preserving graph embedding // Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. P. 1105–1114. https://doi.org/10.1145/2939672.2939751
24. Grover A., Leskovec J. Node2vec: Scalable feature learning for networks // Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. P. 855–864. https://doi.org/10.1145/2939672.2939754
25. Wang D., Cui P., Zhu W. Structural deep network embedding // Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. P. 1225–1234. https://doi.org/10.1145/2939672.2939753
26. Goyal P., Chhetri S.R., Canedo A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning // Knowledge-Based Systems. 2020. V. 187. P. 104816. https://doi.org/10.1016/j.knosys.2019.06.024
27. Ma Y., Guo Z., Ren Z., Tang J., Yin D. Streaming graph neural networks // Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020. P. 719–728. https://doi.org/10.1145/3397271.3401092
28. Kool W., Van Hoof H., Welling M. Attention, learn to solve routing problems! // Proc. of the 7th International Conference on Learning Representations (ICLR). 2019.
29. Komer B., Bergstra J., Eliasmith C. Hyperopt-sklearn // Automated Machine Learning. 2019. P. 97–111. https://doi.org/10.1007/978-3-030-05318-5_5

22. Barros C.D., Mendonça M.R., Vieira A.B., Ziviani A. A survey on embedding dynamic graphs. *ACM Computing Surveys*, 2021, vol. 55, no. 1, pp. 1–37. https://doi.org/10.1145/3483595
23. Ou M., Cui P., Pei J., Zhang Z., Zhu W. Asymmetric transitivity preserving graph embedding. *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1105–1114. https://doi.org/10.1145/2939672.2939751
24. Grover A., Leskovec J. Node2vec: Scalable feature learning for networks. *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864. https://doi.org/10.1145/2939672.2939754
25. Wang D., Cui P., Zhu W. Structural deep network embedding. *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234. https://doi.org/10.1145/2939672.2939753
26. Goyal P., Chhetri S.R., Canedo A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 2020, vol. 187, pp. 104816. https://doi.org/10.1016/j.knosys.2019.06.024
27. Ma Y., Guo Z., Ren Z., Tang J., Yin D. Streaming graph neural networks. *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 719–728. https://doi.org/10.1145/3397271.3401092
28. Kool W., Van Hoof H., Welling M. Attention, learn to solve routing problems! *Proc. of the 7th International Conference on Learning Representations (ICLR)*, 2019.
29. Komer B., Bergstra J., Eliasmith C. Hyperopt-sklearn. *Automated Machine Learning*, 2019, pp. 97–111. https://doi.org/10.1007/978-3-030-05318-5_5

Авторы

Рыбкин Константин Евгеньевич — инженер, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0003-0856-4276, matveich1903@gmail.com
Фильченков Андрей Александрович — кандидат физико-математических наук, инженер, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 55507568200, https://orcid.org/0000-0002-1133-8432, afilchenkov@itmo.ru
Азаров Артур Александрович — кандидат технических наук, научный сотрудник, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; заместитель директора, Северо-Западный институт управления — филиал РАНХиГС, Санкт-Петербург, 199178, Российская Федерация, SC 56938354700, https://orcid.org/0000-0003-3240-597X, artur-azarov@yandex.ru
Забашта Алексей Сергеевич — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 56902663900, https://orcid.org/0000-0002-7494-4309, azabashta@itmo.ru
Шалыто Анатолий Абрамович — доктор технических наук, профессор, главный научный сотрудник, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 56131789500, https://orcid.org/0000-0002-2723-2077, shalyto@mail.ifmo.ru

Authors

Konstantin E. Rybkin — Engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0003-0856-4276, matveich1903@gmail.com
Andrey A. Filchenkov — PhD (Physics & Mathematics), Engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 55507568200, https://orcid.org/0000-0002-1133-8432, afilchenkov@itmo.ru
Artur A. Azarov — PhD, Scientific Researcher, ITMO University, Saint Petersburg, 197101, Russian Federation; Deputy Director, North-West Institute of Management - branch of the Russian Presidential Academy of National Economy and Public Administration, Saint Petersburg, 199178, Russian Federation, SC 56938354700, https://orcid.org/0000-0003-3240-597X, artur-azarov@yandex.ru
Alexey S. Zabashta — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 56902663900, https://orcid.org/0000-0002-7494-4309, azabashta@itmo.ru

Anatoly A. Shalyto — D. Sc., Professor, Chief Reseacher, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 56131789500, https://orcid.org/0000-0002-2723-2077, shalyto@mail.ifmo.ru

Статья поступила в редакцию 30.10.2022
Одобрена после рецензирования 03.11.2022
Принята к печати 29.11.2022

Received 30.10.2022
Approved after reviewing 03.11.2022
Accepted 29.11.2022