УДК 004.75 EDN: TPNAZP

МУЛЬТИАГЕНТНАЯ МИКРОСЕРВИСНАЯ АРХИТЕКТУРА

Е.Ф. Моргунов morgunov@bmstu.ru
А.Н. Алфимцев alfim@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Микросервисная архитектура — это неотъемлемая часть распределенных программных систем, требующих постоянного масштабирования и независимого развертывания всех элементов. Преимущества микросервисов позволяют значительно повысить эффективность современных веб-приложений и открывают новые возможности для развития бизнеса. Однако динамическая изменчивость современных интернет-сервисов, эволюция пользовательских потребностей, а также различные внешние факторы могут нивелировать преимущества микросервисной архитектуры. Перспективным способом адаптивного управления ресурсами распределенных программных систем являются алгоритмы машинного обучения, в особенности алгоритмы глубокого обучения с подкреплением. Рассмотрена интеграция микросервисной архитектуры и мультиагентного обучения с подкреплением. Объединение указанных подходов позволяет оптимизировать работу веб-приложений в нестационарных средах, позволяя системе адаптироваться к изменениям и находить оптимальные решения. Приведены результаты обучения мультиагентного алгоритма независимого Q-обучения в сервисе выбора дорожного маршрута на основе текущего состояния погоды. Для оценки эффективности системы разработаны и введены дополнительные параметры качества обслуживания, позволяющие в полной мере оценить потенциал интеграции микросервисной архитектуры с мультиагентным обучением для решения комплексных задач в динамических средах

Ключевые слова

Распределенная система, микросервисы, глубокое обучение, мультиагентное обучение с подкреплением, нейронные сети, циклическая среда агента, QoS

Поступила 18.11.2024
Принята 30.01.2025

Работа выполнена при поддержке Минобрнауки России в рамках государственного задания (проект № FSFN-2024-0059)

Введение. Микросервисы — тип сервис-ориентированной архитектуры программного обеспечения, основанный на отказе от единой, монолитной структуры [1]. Данная архитектура состоит из нескольких независимых сервисов, которые соответствуют ограниченному контексту. Каждый сервис выполняет отдельную задачу и может быть развернут на отдельном сервере, взаимодействуя с другими сервисами по сети с помощью сетевых протоколов. Сервис рассматривается как один элементарный процесс, однако вместе они представляют единую систему со сложной архитектурой [2].

Микросервисная архитектура (MSA) имеет множество преимуществ перед монолитной, среди которых технологическая неоднородность и гетерогенность, надежность, масштабирование, компонуемость и простота развертывания, что позволяет создавать и развивать большие приложения с комплексной структурой, которые были бы невозможны или неэффективны при использовании монолитной архитектуры [3]. Благодаря этим преимуществам микросервисы широко распространены в различных компаниях с большими и распределенными командами разработки, которым необходимо регулярно масштабировать и обновлять компоненты внутри системы.

Сейчас MSA находит широкое применение не только в процессе создания веб-приложений, но и в сферах финансовых технологий, здравоохранения и менеджмента городских инфраструктур [4]. Однако для обработки больших объемов данных системы должны иметь не только сложную распределенную архитектуру, но и быть адаптивными. Чтобы эффективно подстраиваться под активно меняющиеся условия и находить оптимальные решения для каждого уникального случая, необходимо не только грамотно декомпозировать систему, но и наделить ее интеллектуальностью [5]. Поэтому совместно с распределенными системами часто применяют алгоритмы машинного обучения (ML).

Алгоритмы машинного обучения имеют ключевые преимущества по сравнению с традиционными методами анализа данных и статистики [6]. Они способны эффективно обрабатывать и анализировать большие объемы данных, в том числе неструктурированных, выявлять сложные зависимости и закономерности, адаптироваться и принимать решения в режиме реального времени. Одним из разделов машинного обучения является обучение с подкреплением (RL). В RL агент (программа или система) помещается в среду и учится взаимодействовать с ней путем проб и ошибок [7]. Агент считывает состояния среды и на основе полученной информации совершает действия, за каждое действие он получает числовое

вознаграждение (подкрепление). Сопоставляя получаемую информацию, агент должен выработать стратегию, которая приведет его к максимальной награде в процессе обучения. В RL агент фокусируется на максимизации долгосрочной награды, что позволяет находить стратегии, которые могут быть неочевидны при краткосрочном анализе.

Активное развитие алгоритмов RL привело к появлению мультиагентного обучения с подкреплением (MARL). С помощью алгоритмов MARL изучают поведение нескольких агентов, помещенных в одну среду и взаимодействующих не только с ней, но и друг с другом. Наличие нескольких агентов позволяет расширить спектр выполняемых задач и открыть новые, более комплексные стратегии, которые агент не смог бы выработать в одиночку. Этот метод обучения используется для решения задач управления поведением, коллективного принятия решений и распределения ресурсов [8]. В недетерминированных динамических системах использование MARL дает следующие преимущества: адаптивность, масштабируемость, инкапсулируемость, эффективность функционирования за счет параллельной обработки данных, а также отказоустойчивость [9]. Ввиду перечисленных преимуществ алгоритмы MARL могут стать оптимальным решением для интеграции с веб-приложениями, обрабатывающими большие объемы данных в режиме реального времени и требующие эффективного распределения ресурсов.

В настоящей работе предложена интеграция микросервисной архитектуры с мультиагентным алгоритмом независимого Q-обучения IQN. Предложены способы классификации возможных мультиагентных микросервисных архитектур, а также методы оценки качества системы. В ходе исследования разработан распределенный сервис по выбору оптимального маршрута на основе анализа текущих погодных условий, приведенный в виде среды MARL. Для оценки эффективности алгоритма IQN разработаны и введены параметры качества обслуживания QoS (Quality of Service): надежность сервиса, время выполнения программы и стоимость обслуживания. Данный подход позволяет не только точнее оценить производительность системы, но и стимулирует агентов на поиск эффективных и комплексных стратегий.

Преимущества глубокого нейросетевого обучения с подкреплением стали причиной активного развития способов интеграции RL с бизнес-функциональностью веб-сервисов [10]. Наиболее распространенный метод — использование алгоритмов обучения с подкреплением для композиции веб-сервисов (WSC). Процесс WSC представляет собой процесс объединения нескольких веб-сервисов в единую систему для достижения

более сложной функциональности. Данный процесс может быть реализован как в микросервисной архитектуре, так и в монолитной.

Композиция веб-сервисов позволяет создавать приложения с сервис-ориентированной архитектурой (SOA). При данном подходе несколько сервисов взаимодействуют друг с другом посредством сетевых вызовов для обеспечения определенного конечного набора возможностей [11]. Несмотря на схожесть сервис-ориентированной архитектуры и микросервисов, компоненты SOA могут быть сильно связаны с единой базой данных и не соответствовать концепции независимого развертывания. Микросервисную архитектуру можно рассматривать как частный способ разработки SOA, имеющей свои четкие критерии создания.

Модель WSC-MDP, использующая мультиагентное обучение для создания адаптивной структуры веб-сервиса, предложена в [12]. Несмотря на инновационность подхода, авторы отмечали, что применение модели марковского процесса принятия решений трудно достижимо на практике, а такие параметры, как стоимость взаимодействия между агентами требуют дополнительного изучения. В дальнейших исследованиях WSC некоторые авторы отходили от мультиагентных алгоритмов, ограничиваясь глубоким Q-обучением. Так, в [13] приведен способ композиции веб-сервиса с использованием алгоритма глубокого Q-обучения, использующий атрибуты QoS для оптимизации работы системы. Способ объединения модели WSC-MDP и метода skyline computing для более эффективной композиции веб-сервисов в динамических условиях рассмотрен в [14].

Концепция MAMS — мультиагентная микросервисная архитектура с разделением агентов на публичных и приватных, а также равноправным взаимодействием между агентами и сервисами без четкой иерархии и управления ресурсами — приведена в [15]. В [16] развито поведение агентов, предложены пассивные и активные стратегии управления ресурсами. Среда OFCOURSE, созданная на основе OpenAI Gym и имитирующая полноценный процесс обработки заказа с использованием нескольких взаимодействующих агентов, которые решают проблему последовательного принятия решений, предложена в [17].

Сравнение рассмотренных моделей веб-сервисных архитектур с использованием глубокого обучения с подкреплением приведены в табл. 1. Несмотря на постоянно увеличивающееся число способов оптимизации веб-сервисов с помощью RL, ни одна модель не предлагает интеграцию мультиагентного обучения в микросервисную архитектуру с применением параметров оценки качества обслуживания QoS. Предлагаемая микросервисная архитектура MARL MSA, в отличие от существующих аналогов,

использует классическое понятие микросервиса С. Ньюмана [18], интегрирует наиболее важные для бизнеса метрики качества обслуживания QoS [19], задействует современную RL-среду циклического типа AEC и стандартный алгоритм MARL.

Таблица 1

Сравнение мультиагентных микросервисных архитектур

Алгоритм	MSA	QoS	RL	MARL
WSC-MDP	–	+	+	+
RL WSC	–	+	+	–
Skyline	–	+	+	–
MAMS	+	–	+	+
CArtAgO	+	–	+	+
OFCOURSE	–	–	+	+
MARL MSA	+	+	+	+

Примечание. +/– — соответствие и несоответствие определенным критериям.

Разработка мультиагентной микросервисной архитектуры. ***Критерии создания микросервисов.*** Для создания MSA необходимо определить ключевые признаки, присущие микросервисам. Несмотря на активное использование микросервисов в различных сферах, многие системы не способны полноценно раскрыть свой потенциал. Причины заключаются в непонимании основных принципов MSA и, как следствие, неэффективном проектировании распределенной системы [20].

Основой MSA является независимое развертывание — возможность внесения изменений и создания новых микросервисов без необходимости вносить изменения в систему [21]. Независимое развертывание позволяет создавать масштабные системы, в которых отдельные элементы могут использовать разные технологии, подходящие для конкретных задач, для повышения эффективности системы. Возможность изменять микросервисы изолированно друг от друга требует четких границ между модулями, качество которых определяется тремя основными концепциями — скрытием информации, связностью и связанностью.

Сокрытие информации подразумевает сокрытие от внешнего мира внутренних деталей реализации сервиса. Чем меньше информации раскрывает сервис, тем проще взаимодействие между сервисами, что позволяет эффективнее вносить изменения в модули за счет ограниченного контекста между ними. Для автономности микросервисов каждый сервис должен иметь необходимый минимум знаний о других сервисах, а число вызовов

между сервисами должно быть ограничено. Одним из способов достижения независимого развертывания является скрытие баз данных — каждый сервис использует свою базу данных для сохранения информации, а взаимодействие нескольких сервисов с одной базой данных минимизировано. Скрытие базы данных помогает обеспечить гибкость системы и защиту данных.

Связность определяет, насколько тесно внутренние функции и элементы сервиса связаны между собой. В микросервисных архитектурах функциональность каждого модуля не распределена по всей системе, а представляет собой единое целое. Степень взаимосвязи между сервисами называется связанностью, и чем она ниже, тем меньше зависимость между модулями в системе. Эффективная микросервисная архитектура требует сильной связности и слабой связанности, что позволяет сгруппировать функционал для минимизации межсервисных изменений.

Многие распределенные системы построены на классической трехуровневой архитектуре — веб-интерфейс, уровень бизнес-логики в виде монолитного бэкенда и традиционная база для хранения данных. За каждый уровень отвечает отдельная команда разработки. Такая система универсальна, но при этом может быть неэффективна в процессе обновления и масштабирования системы. При необходимости внесения преобразований в бизнес-функциональность высока вероятность, что изменения затронут всю систему, поскольку функциональность распределена по всем трем уровням. Микросервисная архитектура MSA фокусируется на связности бизнес функциональности, а не технологий [22]. Таким образом, разделив архитектуру на микросервисы, отвечающие за конкретный функционал, а не объединенные единым технологическим стеком, можно создать сильно связную и слабо связанную архитектуру, в которой внесение преобразований ограничивается одним модулем. Вышедший из строя компонент системы можно изолировать, сохранив работоспособность системы в целом. Данные особенности позволяют эффективно использовать микросервисную систему для создания сложных, неоднородных веб-приложений и добавлять новый функционал по мере развития сервиса.

Способы интеграции MARL в микросервисы. Перед объединением микросервисной архитектуры и мультиагентного обучения необходимо определить, каким образом агенты будут встроены в распределенную систему. Поскольку каждый интеллектуальный агент будет управлять ресурсами системы, его взаимодействие с другими сервисами является основой интеграции. Можно выделить три различных способа интеграции агентов в MSA.

Один агент–один микросервис. Такой способ предоставляет возможность каждому агенту управлять внутренними процессами и данными своего микросервиса. Способ является одновременно простым и эффективным, поскольку позволяет создавать и использовать в системе большое число агентов при минимальных изменениях в функционале, что особенно полезно для масштабируемых архитектур. Внедрение агентов в микросервисы наделяет их интеллектуальностью, при этом сохраняя независимость каждого компонента системы. В случае сбоя интеллектуального сервиса проблему можно быстро отследить и решить, изолировав проблемный элемент. Преимущества этого способа — простота реализации, масштабируемость и надежность.

Несколько агентов–один микросервис. Этот способ может подойти для комплексных компонентов системы со сложным функционалом, требующим одновременной обработки большого объема данных. Для эффективного использования агентов в таких сервисах могут использоваться централизованные мультиагентные алгоритмы с децентрализованным исполнением, например, MADDPG и VDN. Интеграция нескольких агентов позволит декомпозировать функционал сервиса на отдельные локальные задачи, значительно повышая производительность системы. Данный способ является универсальным и особенно эффективным для решения сложных задач, требующих кооперативной или конкурентной стратегии.

Один агент–несколько микросервисов. Такая концепция позволит использовать агента для управления ресурсами внутри нескольких сервисов. Агент может выступать как менеджер по взаимодействию между элементами системы, собирая данные и запросы с микросервисов и направляя их нужным сервисам. Такой способ позволяет значительно повысить скорость и качество коммуникации между сервисами, минимизируя число запросов и, следовательно, снижая связанность микросервисов. Для данного метода могут подойти централизованные алгоритмы MARL, а также классические одноагентные алгоритмы глубокого обучения, например DQN.

Каждый способ интеграции будет эффективен для решения определенного типа задач. Выбор способа и алгоритма MARL зависит не только от функционала сервиса, но и от ожидаемой от агентов стратегии — независимой, кооперативной, конкурентной или смешанной [23]. Благодаря технологической неоднородности MSA в системе может использоваться сразу несколько подходов для выявления наиболее эффективного метода взаимодействия агентов и микросервисов. На рис. 1 приведены структурные схемы каждого возможного способа интеграции агентов в микросервисную систему.

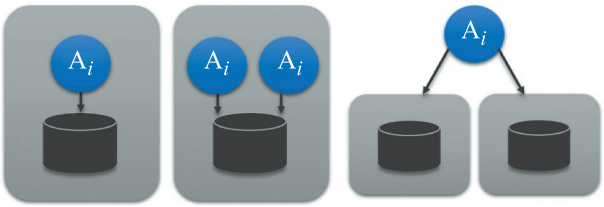


Рис. 1. Способы интеграции агентов (A_i, A_j) в MSA:
один агент–один микросервис (*а*), несколько агентов–один микросервис (*б*),
один агент–несколько микросервисов (*в*)

Разработка мультиагентной системы. Ключевая задача современных веб-приложений — взаимодействие с клиентом в режиме реального времени для предоставления наиболее подходящих услуг и предложений. Комплексные веб-приложения часто являются динамическими системами, где различные факторы и параметры могут меняться с течением времени и оказывать влияние на работоспособность системы, а также на качество обслуживания клиентов [24]. Такие задачи требуют сложных вычислений, способных подстраиваться под внешние изменения и потребности клиентов. Примером такого веб-приложения являются сервисы выбора дорожных маршрутов, которые зависят от текущей загруженности на дорогах, погодных условий и числа одновременных запросов от клиентов. Для эффективной работы таких систем может быть недостаточно заранее прописанных инструкций, поскольку число возможных состояний среды слишком велико и не все стратегии можно спрогнозировать заранее. Алгоритмы MARL способны решить проблему управления комплексными веб-приложениями, поскольку обучают агентов эффективно реагировать на динамические условия среды и адаптироваться к ним, изменяя свои стратегии для достижения наилучшего результата [25].

Для исследования интеграции мультиагентного обучения в микросервисную систему разработан сервис выбора туристического маршрута. На основе информации о текущих погодных условиях и загруженности направлений система предлагает клиенту наиболее эффективный маршрут до искомой точки, минимизирующий затраты клиента на путешествие. Сервис представляет собой классическую микросервисную архитектуру, сочетающую в себе все признаки микросервиса с его преимуществами перед монолитными структурами.

Обобщенная мультиагентная микросервисная архитектура MARL MSA приведена на рис. 2. Система состоит из пользовательского интерфейса UI и нескольких микросервисов. В предлагаемой экспериментальной реали-

зации это сервисы генерации погоды, среды для реализации маршрута агентов, а также нейронной сети. Вместо горизонтальной многоуровневой архитектуры и организации используется вертикальная, благодаря чему каждый компонент — это отдельная бизнес-функциональность. В настоящей работе рассмотрена концепция создания MSA по типу один сервис–один агент, поэтому в систему помещены два агента-микросервиса, управляющие выбором маршрута в среде. Микросервисная архитектура реализована с помощью нескольких языков программирования — для реализации UI использованы языки Python, HTML и Java Script, а среда и алгоритм IQN написаны на языке Python с использованием библиотек NumPy, Py Torch и Matplotlib. Программный код сервиса выбора дорожного маршрута доступен на сайте[1].

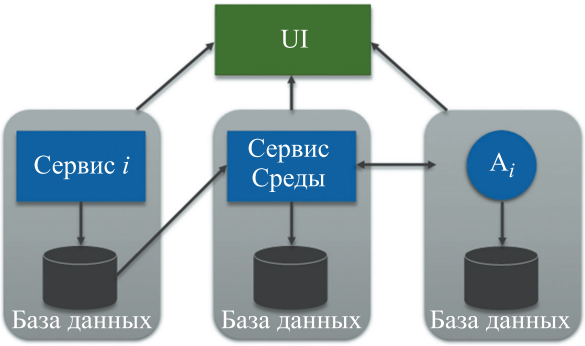


Рис. 2. Обобщенная мультиагентная микросервисная архитектура MARL MSA

Все элементы системы имеют четкие границы и собственный функционал, скрытый от других сервисов. В процессе работы каждый микросервис использует свою базу данных для сохранения информации. Остальные сервисы не могут вносить изменения в чужую базу данных, а лишь считывают информацию, если в этом есть необходимость. В резудьтате пользовательский интерфейс считывает информацию от сервисов через базы данных, составляя для пользователя исчерпывающий ответ с минимальным раскрытием внутреннего функционала. Таким образом, вводится ограниченный контекст и минимизируется число взаимосвязей между сервисами и число их обращений друг к другу.

Мультиагентная среда для обучения создана на основе всех достоинств библиотек одноагентного и мультиагентного обучений и существующих

[1] MARL MSA — архитектура микросервисов многоагентного обучения с подкреплением. URL: https://github.com/egormorgunov/marl_msa (дата обращения: 15.02.2025).

математических моделей для обучения с подкреплением и представляет собой сервис выбора дорожного маршрута. В отличие от классических сред OpenAI Gym, использующих математическую модель частично наблюдаемого марковского процесса принятия решений (POMDP), а также популярной в мультиагентных системах модели частично наблюдаемой стохастической игры (POSG), при создании сервиса выбора маршрута использована модель цикла среды агентов (AEC) [26]. Для создания сред PettingZoo модель AEC по сути представляет собой пошаговую версию POSG. На каждом шаге агент последовательно получает свои состояния, совершает действие и получает награду от среды, а затем выбирается следующий агент для выполнения действия. Пошаговые действия агентов позволяют избавиться от необходимости использования множества фиктивных действий для одновременного выполнения строго последовательных действий в системе, обычных для среды POSG.

Еще одной важной особенностью мультиагентной микросервисной архитектуры MARL MSA является введение нового агента — агента «среды». Когда агент совершает действие, происходит обновление самой среды, которая считывает действия агентов и реагирует на них. Таким образом, сама среда по сути становится действующим агентом игры. Это позволяет точнее определить награды агентов, причины их действий и текущих состояний. Модель AEC отлично подходит для сред с большим числом агентов и учитывает ситуации, когда их число может меняться в процессе обучения (при добавлении нового агента или смерти старого). Структура двухагентной игры приведена на рис. 3.

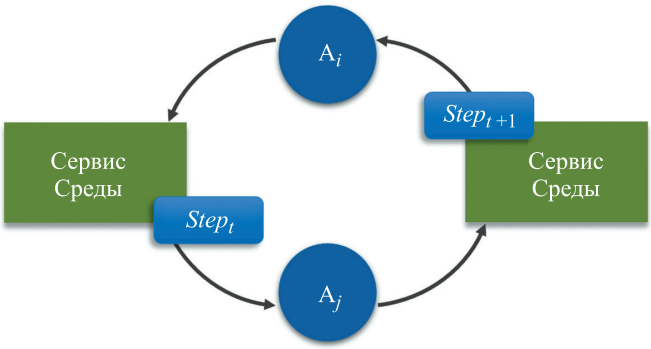


Рис. 3. Структура игры с двумя агентами на основе AEC-модели ($step_t$ и $step_{t+1}$ — шаги среды)

Структурная схема разработанной среды приведена на рис. 4. Каждое состояние среды представляет собой шаг, на котором может оказаться агент в процессе оказания услуги. У каждого агента есть набор действий,

которые меняются по мере продвижения по среде. Для выработки более комплексной стратегии агент может не только продвигаться вперед, выполняя заказ, но и вернуться в предыдущее состояние или пропустить шаг, не совершая действия. Состояние, которое получает агент от среды, представляет собой список из трех элементов — текущего шага среды, выбранного агентом действия и типа погоды, которая оказывает прямое воздействие на дорожные маршруты.

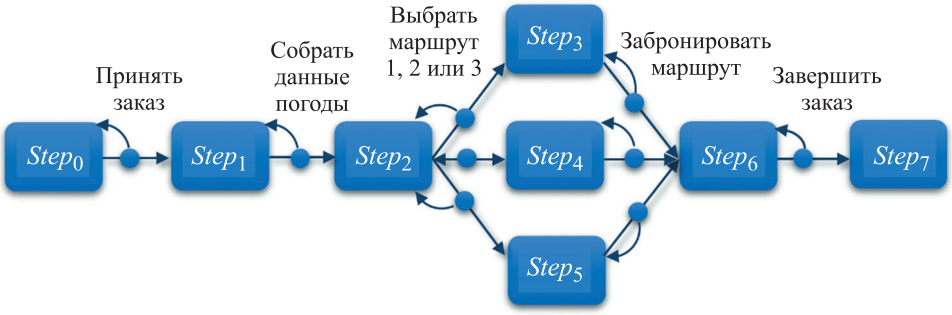


Рис. 4. Структурная схема мультиагентной среды для выбора маршрута

Здесь при проектировании системы использована концепция один агент–один микросервис. Каждый агент представляет собой обособленный модуль, взаимодействующий со средой и своей базой данных. При данном подходе число агентов можно с легкостью увеличить, не прибегая к серьезным изменениям во всей системе, а независимость обучения позволит каждому агенту выработать свою уникальную стратегию.

Разработанная система соответствует всем критериям MSA — независимое развертывание, масштабируемость, скрытие данных, сильная связность и слабая связанность. Однако подобная система не сможет эффективно работать в динамических условиях без добавления алгоритма MARL.

Мультиагентное обучение с подкреплением. Одним из популярных методов обучения с подкреплением является алгоритм глубокого Q-обучения (DQN) [27]. Алгоритм DQN объединяет алгоритм Q-обучения, использующий оценку Q-функций для нахождения максимальной долгосрочной награды, и глубокие нейронные сети для обработки многомерных пространств состояний. Глубокие нейронные сети значительно повысили применимость алгоритмов, позволив работать с непрерывными действиями агентов и автоматизировать создание характерных признаков.

Для повышения сходимости и стабильности обучения в DQN используется буфер воспроизведения, сохраняющий все кортежи $(s, a, r, s'$, где s — состояние агента; a — совершаемое действие; r — получаемая

награда; s' — следующее состояние) в качестве будущих обучающих данных. Такой подход обеспечивает алгоритм независимыми данными, позволяет снизить осцилляцию обучения и повышает эффективность обучения за счет повторного использования данных.

Для повышения стабильности обучения алгоритм DQN использует дополнительную целевую нейронную сеть, которая дублирует основную [28]. Все веса целевой нейронной сети $Q(s', a', \theta')$ берутся из весов $Q(s, a, \theta)$ основной сети, но обновление происходит не сразу, а с определенной задержкой, во время которой они остаются неизменными. Данный метод значительно повышает стабильность обучения, стимулируя агента вырабатывать долгосрочную стратегию.

Алгоритм DQN является эффективным способом решения широкого спектра задач, однако предназначен для обучения только одного агента. Для мультиагентного обучения с подкреплением существует несколько модификаций DQN, одной из которых является алгоритм независимого Q-обучения IQN [29]. В алгоритме IQN каждый агент обучается независимо от других, используя только собственный опыт и наблюдения. В каждого агента встроена одинаковая нейросетевая архитектура без явной коммуникации между агентами, что позволяет каждому агенту выработать свою уникальную стратегию.

Алгоритм IQN сохранил все достоинства глубокого Q-обучения и имеет следующие преимущества: децентрализованность архитектуры, простота реализации, масштабируемость, надежность и совместимость. В связи с этим именно алгоритм IQN выбран для обучения агентов в микросервисной архитектуре [30, 31].

Структура IQN приведена в алгоритме 1, где α — скорость обучения; γ — шаг дисконтирования; ε — параметр; N_e — число эпизодов обучения; E_b — буфер воспроизведения; B_l — объем буфера воспроизведения; B_s — объем мини-выборки из буфера воспроизведения; C_s — период копирования весов из основной нейронной сети в целевую (в эпизодах); θ и θ' — веса основной и целевой нейронных сетей.

Алгоритм 1. IQN.

1. **Инициализировать:**

$a \in (0, 1]$; $\gamma \in (0, 1]$; $\varepsilon \in [0.1, 1]$; $N_e \in (0, N]$;

$E_b \leftarrow 0$; $B_l \leftarrow 10\,000$; $B_s \leftarrow 32$; $C_s \leftarrow 100$;

$i \leftarrow 1, \ldots, n$; $a \in A_i$; $j \leftarrow 1, \ldots, B_s$; $K \leftarrow |A_i|$;

$\theta \leftarrow \xi$; $\theta' \leftarrow \xi$; $\forall_s \in S$.

2. **Цикл:**

3. Получить состояние s.

4. Определить возможные действия $a \in A_i$ в состоянии s.

5. Передать состояние s в основную нейронную сеть и выбрать возможное действие a в соответствии с максимальным значением $Q_1(s, a; \theta)$, $Q_2(s, a; \theta)$, …, $Q_k(s, a; \theta)$ и с учетом параметра ε.

6. Выполнить возможное действие a.

7. Получить награду r и следующее состояние s'.

8. Сохранить (s, a, r, s') в буфере воспроизведения E_b. Если объем B_l превышен, то заменить значение по принципу очереди *FIFO*.

9. Случайно выбрать мини-выборку (s, a, r, s') объемом B_s из буфера воспроизведения E_b.

10. Для каждого j кортежа (s, a, r, s') из мини-выборки вычислить $Q^j(s', a'; \theta')$, передав в целевую нейронную сеть s'.

11. Вычислить $y^j \leftarrow \begin{cases} r, & \text{если } t = T; \\ r + \gamma \max\limits_{a'} Q^j(s', a'; \theta'), & \text{если } t \neq T. \end{cases}$

12. Для каждого j кортежа (s, a, r, s') из мини-выборки вычислить $Q^j(s', a'; \theta')$, передав в целевую нейронную сеть s.

13. Вычислить функцию потерь $L^j(\theta) \leftarrow \frac{1}{K} \sum\limits_{p=1}^{K} (y_p^j - Q_p^j(s, a, \theta)^2)$.

14. Обновить веса θ основной нейронной сети с использованием $L^j(\theta)$ и скорости обучения α.

15. Заменить веса θ' целевой нейронной сети весами θ основной нейронной сети, если закончился очередной период C_s.

В настоящей работе в алгоритме IQN используется нейронная сеть, которая включает в себя три полносвязных линейных слоя, с входными и выходными размерами соответственно 3 × 36, 36 × 36, 36 × 3. На вход нейронная сеть получает состояние среды s, заданное тремя отсчетами, а на выходе возвращает три Q-значения, соответствующие возможным действиям агента.

Качество обслуживания микросервисов. Для более точной оценки эффективности мультиагентного обучения в MSA использована концепция QoS — индикатор, позволяющий оценить нефункциональные характеристики системы и способность сервиса удовлетворять требованиям пользователей [32]. В этой работе QoS использовано в виде набора параметров, применяющихся для расчета награды агентов и непосредственно

отражающих индустриальные требования к системе. Использовано три атрибута качества QoS.

1. *Время выполнения заказа*. Поскольку агенту необходимо как можно быстрее обработать заказ пользователя, данный параметр позволяет оценить производительность интеллектуальных сервисов и стимулировать агентов оперативно решать поставленные задачи.

2. *Стоимость выполнения заказа*. Процесс обслуживания клиентов является энергозатратной задачей, оказывающей нагрузку на сервис. Поскольку каждая операция в системе имеет свою стоимость, данный параметр стимулирует агентов грамотно управлять ресурсами, минимизируя финансовые затраты, потраченные на выполнение задачи.

3. *Надежность сервиса*. Интеллектуальные агенты должны выполнять заказы не только быстро, но и правильно, удовлетворяя запросам пользователя. В процессе обработки заказов система определяет, все ли условия для успешного завершения заказа выполнены, и затем награждает или штрафует агента в зависимости от совершенных ошибок. Данный параметр стимулирует агентов анализировать свои действия и искать стратегии для максимизации эффективности выбора маршрута и дальнейшей процедуры бронирования.

Эксперименты. *Визуализация обучения и награды.* Внешний вид пользовательского интерфейса разработанного сервиса выбора маршрута приведен на рис. 5. Для наглядности обучения пользовательский интерфейс UI разработанного сервиса в режиме реального времени отображает текущие погодные условия, этапы выполнения заказа, состояние агента и информацию по бронированию маршрута.

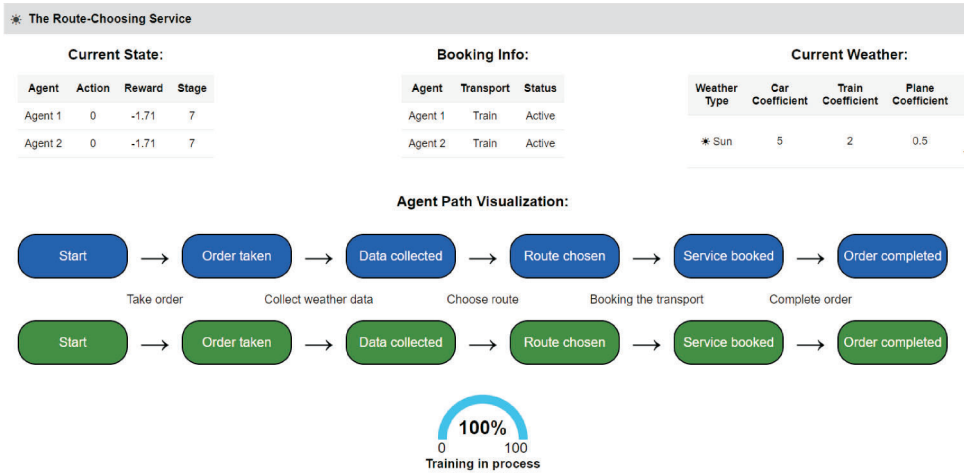


Рис. 5. Пользовательский интерфейс разработанного сервиса выбора маршрута

Сервис генерации погоды в случайном порядке генерирует погодные условия для всего маршрута. Каждый тип погоды влияет на состояние маршрутов с помощью погодных коэффициентов. Данный подход имитирует сложности, возникающие у пользователей в процессе путешествия из-за изменяющихся внешних факторов. Динамическое изменение погоды стимулирует агентов искать более комплексную стратегию и оперативно реагировать на изменения состояния среды.

Агенты начинают выполнение заказов одновременно и пошагово совершают действия по обслуживанию клиентов. Эпизод завершается, когда оба агента выполняют заказы, а более оперативный агент не получает штрафов за ожидание другого. Такой подход позволяет точнее отследить и сравнить независимые стратегии агентов друг с другом.

Агенты получают награду как за каждый шаг, так и по результату выполнения заказа и завершения эпизода. Награда агентов отражает основную цель интеграции агентов в MSA, которая заключается в оптимизации процессов микросервисов для решения задач с динамическими условиями. Для каждого агента i на шаге s определяем награду r как

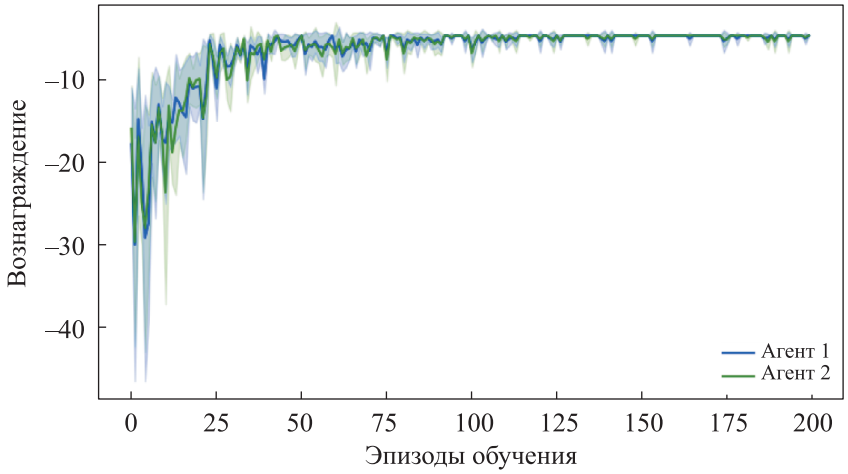
$$r_s^i = \begin{cases} -c_s, & s = h; \\ -c_s - n_a, & s = l; \\ -c_s - m_a, & s = b; \\ -c_s - t, & s = e, \end{cases}$$

где c_s — стоимость обслуживания сервиса на шаге s; $s = h$ определяет обычный шаг обслуживания клиента; n_a — погодный коэффициент для выбранного маршрута a; $s = l$ — шаг выбора дорожного маршрута; m_a — надежность сервиса при бронировании маршрута a; $s = b$ — шаг бронирования маршрута; t — время выполнения заказа; $s = e$ — последний шаг эпизода. Для стимуляции агента к эффективной обработке заказов, а также во избежание эксплуатации действий, дающих краткосрочную положительную награду, все награды в ходе обучения отрицательны.

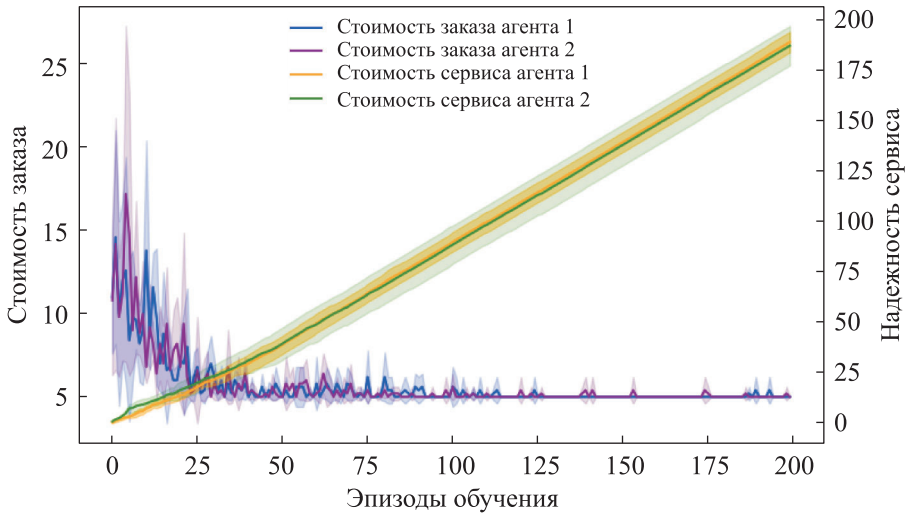
Выбранные параметры QoS оказывают прямое влияние на награды агентов, стимулируя их к минимизации стоимости выполнения заказов и повышению производительности сервисов. В процессе экспериментов эффективность веб-приложения рассмотрена не только со стороны разработчика, но и со стороны пользователя.

Обсуждение полученных результатов. Результаты усредненного обучения алгоритма IQN в разработанной мультиагентной микросервисной среде с участием двух агентов приведены на рис. 6. Для обучения ал-

горитма заданы следующие параметры: $\gamma = 0{,}95$ — фактор дисконтирования; $\alpha = 0{,}001$ — скорость обучения; $\varepsilon = 1$ — параметр в начале обучения; $\varepsilon_d = 0{,}995$ ε_d — уменьшение за эпизод; $\varepsilon_{\min} = 0{,}01$ — минимальное значение; batch_size = 32 — размер пакета. Эксперимент разбит на 200 эпизодов. Для каждого параметра приведены усредненные результаты по результатам пяти независимых экспериментов. В экспериментах оценивалась награда, получаемая агентами за выполнение заказа, и качество обслуживания сервиса.



а



б

Рис. 6. Результаты обучения алгоритма IQN в разработанной микросервисной среде (*а*) и качество обслуживания сервисов в процессе обучения (*б*)

Согласно результатам эксперимента, оба агента достаточно быстро обучались выполнять заказы в режиме реального времени, минимизируя стоимость обслуживания. Однако это сильно сказывалось на надежности системы, что приводило к неоптимальному выбору маршрута или к ошибкам при его бронировании. В процессе обучения оба агента смогли независимо прийти к оптимальной стратегии, позволяющей максимизировать награду и эффективно обслуживать пользователей. Оценка параметров QoS демонстрирует, что в процессе обучения надежность сервисов монотонно возрастает, а стоимость обработки заказа снижается, что соответствует индустриальным требованиям к системе. Таким образом, алгоритм IQN сумел успешно обучить агентов обрабатывать заказы в микросервисной системе, отвечая всем требованиям системы и соответствуя необходимым параметрам QoS в динамической среде с меняющимися условиями.

Заключение. Проведено исследование интеграции мультиагентного обучения с подкреплением в микросервисную архитектуру. Разработанный сервис выбора дорожного маршрута соответствует всем критериям и особенностям MSA: независимое развертывание, сильная связность и слабая связанность, масштабируемость и технологическая неоднородность. Предложены способы интеграции агентов в микросервисную систему, а также рассмотрены параметры качества обслуживания: время и стоимость выполнения заказа, надежность сервиса, позволяющие в полной мере оценить производительность сервисов и удовлетворенность пользователей. В ходе экспериментов использован алгоритм независимого глубокого обучения с использованием полносвязной нейронной сети IQN, который достиг сходимости в процессе поиска оптимальной стратегии. Результаты экспериментов обосновывают способность мультиагентного обучения с подкреплением успешно интегрироваться в микросервисные системы, а совмещение MARL и MSA позволяет повысить производительность распределенных систем, отвечая ожиданиям пользователей и соответствуя индустриальным требованиям к микросервисной архитектуре.

В будущих работах планируется рассмотреть другие найденные способы интеграции мультиагентного обучения и микросервисных систем. Концепции несколько агентов–один микросервис и один агент–несколько микросервисов могут стать эффективным решением для различных задач внутри микросервисов, требующих более комплексного взаимодействия между агентами. Для данных концепций планируется применить гетерогенные алгоритмы MARL и подробнее изучить поведение разных агентов и их совместные стратегии [33]. Кооперативное гетерогенное взаимодействие агентов позволит решить комплексные задачи, но вместе с тем может

привести и к конфликтам среди агентов, что влечет за собой появление социальных дилемм, которые необходимо оперативно идентифицировать и решать [34].

ЛИТЕРАТУРА

[1] Nadareishvili I., Mitra R., McLarty M., et al. Microservice architecture. Sebastopol, O'Reilly Media, 2016.

[2] Thönes J. Microservices. *IEEE Softw.,* 2015, vol. 32, no. 1, p. 116.
DOI: https://doi.org/10.1109/MS.2015.11

[3] Blinowski G., Ojdowska A., Przybyłek A. Monolithic vs. microservice architecture: a performance and scalability evaluation. *IEEE Access*, 2022, vol. 10, pp. 20357–20374.
DOI: https://doi.org/10.1109/ACCESS.2022.3152803

[4] Hasselbring W., Steinacker G. Microservice architectures for scalability, agility and reliability in e-commerce. *IEEE ICSAW,* 2017, pp. 243–246.
DOI: https://doi.org/10.1109/ICSAW.2017.11

[5] Camero A., Alba E. Smart City and information technology: a review. *Cities*, 2019, vol. 93, pp. 84–94. DOI: https://doi.org/10.1016/j.cities.2019.04.014

[6] Jordan M.I., Mitchell T.M. Machine learning: trends, perspectives, and prospects. *Science*, 2015, vol. 349, no. 6245, pp. 255–260.
DOI: https://doi.org/10.1126/science.aaa8415

[7] Sutton R.S., Barto A.G. Reinforcement learning. An introduction. London, MIT Press Cambridge, 2018.

[8] Canese L., Cardarilli G.C., Di Nunzio L., et al. Multi-agent reinforcement learning: a review of challenges and applications. *Appl. Sc.*, 2021, vol. 11, no. 11, art. 4948.
DOI: https://doi.org/10.3390/app11114948

[9] Lee D., He N., Kamalaruban P., et al. Optimization for reinforcement learning: from a single agent to cooperative agents. *IEEE Signal Process. Mag.*, 2020, vol. 37, no. 3, pp. 123–135. DOI: https://doi.org/10.1109/MSP.2020.2976000

[10] Putta P., Mills E., Garg N., et al. Agent Q: advanced reasoning and learning for autonomous AI agents. arXiv:2408.07199. DOI: https://doi.org/10.48550/arXiv.2408.07199

[11] Niknejad N., Ismail W., Ghani I., et al. Understanding Service-Oriented Architecture (SOA): a systematic literature review and directions for further investigation. *Inf. Syst.*, 2020, vol. 91, art. 101491. DOI: https://doi.org/10.1016/j.is.2020.101491

[12] Wang H., Wang X., Hu X., et al. A multi-agent reinforcement learning approach to dynamic service composition. *Inf. Sc.*, 2016, vol. 363, pp. 96–119.
DOI: https://doi.org/10.1016/j.ins.2016.05.002

[13] Wang H., Gu M., Yu Q., et al. Adaptive and large-scale service composition based on deep reinforcement learning. *Knowl.-Based Syst.*, 2019, vol. 180, no. 10, pp. 75–90.
DOI: https://doi.org/10.1016/j.knosys.2019.05.020

[14] Wang H., Hu X., Yu Q., et al. Integrating reinforcement learning and skyline computing for adaptive service composition. *Inf. Sc.*, 2020, vol. 519, pp. 141–160. DOI: https://doi.org/10.1016/j.ins.2020.01.039

[15] Collier R., O'Neill E., Lillis D., et al. MAMS: multi-agent microservices. *WWW'19*, 2019, pp. 655–662. DOI: https://doi.org/10.1145/3308560.3316509

[16] O'Neill E., Lillis D., O'Hare G.M., et al. Delivering multi-agent MicroServices using CArtAgO. In: *Engineering multi-agent systems*. Cham, Springer International Publishing, 2020, pp. 1–20. DOI: https://doi.org/10.1007/978-3-030-66534-0_1

[17] Zhu Y., Zhan Y., Huang X., et al. OFCOURSE: a multi-agent reinforcement learning environment for order fulfillment. *NeurIPS*, 2024, vol. 36. URL: https://proceedings.nips.cc/paper_files/paper/2023/hash/6d0cfc5db3feeabf6762129ba91bd3a1-Abstract-Datasets_and_Benchmarks.html (дата обращения: 15.02.2025).

[18] Newman S. Building microservices. Sebastopol, O'Reilly Media, 2021.

[19] Li S., Zhang H., Jia Z., et al. Understanding and addressing quality attributes of microservices architecture: a systematic literature review. *Inf. Softw. Technol.*, 2021, vol. 131, art. 106449. DOI: https://doi.org/10.1016/j.infsof.2020.106449

[20] Velepucha V., Flores P. A survey on microservices architecture: principles, patterns and migration challenges. *IEEE Access*, 2023, vol. 11, pp. 88339–88358. DOI: https://doi.org/10.1109/ACCESS.2023.3305687

[21] Newman S. Monolith to microservices. Sebastopol, O'Reilly Media, 2019.

[22] Waseem M., Liang P., Shahin M. A systematic mapping study on microservices architecture in devops. *J. Syst. Softw.*, 2020, vol. 170, art. 110798. DOI: https://doi.org/10.1016/j.jss.2020.110798

[23] Leroy P., Morato P.G., Pisane J., et al. IMP-MARL: a suite of environments for large-scale infrastructure management planning via MARL. *NeurIPS*, 2024. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/a7a7c0c92f195cce85f99768621ac6c0-Abstract-Datasets_and_Benchmarks.html (дата обращения: 15.02.2025).

[24] Bacchiani L., Bravetti M., Giallorenzo S., et al. Microservice dynamic architecture-level deployment orchestration. In: *Coordination Models and Languages*. Cham, Springer Nature, 2021, pp. 257–275. DOI: https://doi.org/10.1007/978-3-030-78142-2_16

[25] Vinyals O., Ewalds T., Bartunov S., et al. Starcraft II: a new challenge for reinforcement learning. arXiv:1708.04782. DOI: https://doi.org/10.48550/arXiv.1708.04782

[26] Terry J., Black B., Grammel N., et al. Pettingzoo: gym for multi-agent reinforcement learning. *NeurIPS*, 2021, vol. 34, pp. 15032–15043.

[27] Fan J., Wang Z., Xie Y., et al. A theoretical analysis of deep Q-learning. *Proc. 2nd Conf. on Learning for Dynamics and Control*, 2020, pp. 486–489.

[28] Bolshakov V.E., Alfimtsev A.N. Hierarchical method for cooperative multiagent reinforcement learning in Markov decision processes. *Dokl. Math.*, 2023, vol. 108, no. 2S, pp. S382–S392. DOI: https://doi.org/10.1134/S1064562423701132

[29] Ozdaglar A., Sayin M.O., Zhang K. Independent learning in stochastic games. In: *International Congress of Mathematicians*, 2021, pp. 5340–5373.
DOI: https://doi.org/10.4171/icm2022/152

[30] Zhang Z., Yang J., Zha H. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. arXiv:1909.10651.
DOI: https://doi.org/10.48550/arXiv.1909.10651

[31] Morgunov E.F., Alfimtsev A.N. The “stag hunt” social dilemma in multi-agent reinforcement learning. *REEPE*, 2024.
DOI: https://doi.org/10.1109/REEPE60449.2024.10479770

[32] Zheng H., Zhao W., Yang J., et al. QoS analysis for web service compositions with complex structures. *IEEE Trans. Serv. Comput.*, 2013, vol. 6, no. 3, pp. 373–386.
DOI: https://doi.org/10.1109/TSC.2012.7

[33] Kuba J.G., Feng X., Ding S., et al. Heterogeneous-agent mirror learning: a continuum of solutions to cooperative marl. arXiv:2208.01682.
DOI: https://doi.org/10.48550/2208.01682

[34] Leibo J.Z., Zambaldi V., Lanctot M., et al. Multi-agent reinforcement learning in sequential social dilemmas. arXiv:1702.03037.
URL: https://doi.org/10.48550/arXiv.1702.03037

Моргунов Егор Феликсович — ассистент кафедры «Информационные системы и телекоммуникации» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Алфимцев Александр Николаевич — д-р техн. наук, заведующий кафедрой «Информационные системы и телекоммуникации» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Просьба ссылаться на эту статью следующим образом:

Моргунов Е.Ф., Алфимцев А.Н. Мультиагентная микросервисная архитектура. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2025, № 2 (151), с. 78–101. EDN: TPNAZP

MULTI-AGENT MICROSERVICE ARCHITECTURE

E.F. Morgunov morgunov@bmstu.ru
A.N. Alfimtsev alfim@bmstu.ru

BMSTU, Moscow, Russian Federation

Abstract

Microservice architecture is an integral part of the distributed software systems that require continuous scaling and independent deployment of all their elements. The microservice advantages could significantly increase efficiency of the modern web applications and

Keywords

Distributed system, microservices, deep learning, multi-agent reinforcement learning, neural networks, cyclic agent environment, QoS

open up new opportunities in the business development. However, dynamic variability of the modern Internet services, evolution in the user needs, as well as various external factors could negate advantages of the microservice architecture. One of the promising methods in adaptive resource management of the distributed software systems includes the machine learning algorithms, especially the deep reinforcement learning algorithms. The paper considers integration of the microservice architecture and the multi-agent reinforcement learning. Combining these approaches makes it possible to optimize the web applications operation in the non-stationary environments allowing the system to adapt to alterations and find the optimal solutions. The paper provides results of learning a classical multi-agent independent Q-learning algorithm in the road route selection service based on the current weather conditions. To evaluate the system efficiency, additional service quality parameters were developed and introduced making it possible to fully evaluate potential of integrating the microservice architecture and the multi-agent learning in solving complex problems in the dynamic environments

Received 18.11.2024
Accepted 30.01.2025

The work was performed with support by the Ministry of Science and Higher Education of the Russian Federation within the framework of the State Task (project no. FSFN-2024-0059)

REFERENCES

[1] Nadareishvili I., Mitra R., McLarty M., et al. Microservice architecture. Sebastopol, O'Reilly Media, 2016.

[2] Thönes J. Microservices. *IEEE Softw*., 2015, vol. 32, no. 1, p. 116.
DOI: https://doi.org/10.1109/MS.2015.11

[3] Blinowski G., Ojdowska A., Przybyłek A. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 2022, vol. 10, pp. 20357–20374.
DOI: https://doi.org/10.1109/ACCESS.2022.3152803

[4] Hasselbring W., Steinacker G. Microservice architectures for scalability, agility and reliability in e-commerce. *IEEE ICSAW*, 2017, pp. 243–246.
DOI: https://doi.org/10.1109/ICSAW.2017.11

[5] Camero A., Alba E. Smart City and information technology: a review. *Cities*, 2019, vol. 93, pp. 84–94. DOI: https://doi.org/10.1016/j.cities.2019.04.014

[6] Jordan M.I., Mitchell T.M. Machine learning: trends, perspectives, and prospects. *Science*, 2015, vol. 349, no. 6245, pp. 255–260.
DOI: https://doi.org/10.1126/science.aaa8415

[7] Sutton R.S., Barto A.G. Reinforcement learning. An introduction. London, MIT Press Cambridge, 2018.

[8] Canese L., Cardarilli G.C., Di Nunzio L., et al. Multi-agent reinforcement learning: a review of challenges and applications. *Appl. Sc*., 2021, vol. 11, no. 11, art. 4948. DOI: https://doi.org/10.3390/app11114948

[9] Lee D., He N., Kamalaruban P., et al. Optimization for reinforcement learning: from a single agent to cooperative agents. *IEEE Signal Process. Mag*., 2020, vol. 37, no. 3, pp. 123–135. DOI: https://doi.org/10.1109/MSP.2020.2976000

[10] Putta P., Mills E., Garg N., et al. Agent Q: advanced reasoning and learning for autonomous AI agents. arXiv:2408.07199. DOI: https://doi.org/10.48550/arXiv.2408.07199

[11] Niknejad N., Ismail W., Ghani I., et al. Understanding Service-Oriented Architecture (SOA): a systematic literature review and directions for further investigation. *Inf. Syst*., 2020, vol. 91, art. 101491. DOI: https://doi.org/10.1016/j.is.2020.101491

[12] Wang H., Wang X., Hu X., et al. A multi-agent reinforcement learning approach to dynamic service composition. *Inf. Sc*., 2016, vol. 363, pp. 96–119. DOI: https://doi.org/10.1016/j.ins.2016.05.002

[13] Wang H., Gu M., Yu Q., et al. Adaptive and large-scale service composition based on deep reinforcement learning. *Knowl.-Based Syst*., 2019, vol. 180, no. 10, pp. 75–90. DOI: https://doi.org/10.1016/j.knosys.2019.05.020

[14] Wang H., Hu X., Yu Q., et al. Integrating reinforcement learning and skyline computing for adaptive service composition. *Inf. Sc*., 2020, vol. 519, pp. 141–160. DOI: https://doi.org/10.1016/j.ins.2020.01.039

[15] Collier R., O'Neill E., Lillis D., et al. MAMS: multi-agent microservices. *WWW'19*, 2019, pp. 655–662. DOI: https://doi.org/10.1145/3308560.3316509

[16] O'Neill E., Lillis D., O'Hare G.M., et al. Delivering multi-agent MicroServices using CArtAgO. In: *Engineering multi-agent systems*. Cham, Springer International Publishing, 2020, pp. 1–20. DOI: https://doi.org/10.1007/978-3-030-66534-0_1

[17] Zhu Y., Zhan Y., Huang X., et al. OFCOURSE: a multi-agent reinforcement learning environment for order fulfillment. *NeurIPS*, 2024, vol. 36. Available at: https://proceedings.nips.cc/paper_files/paper/2023/hash/6d0cfc5db3feeabf6762129ba91bd3a1-Abstract-Datasets_and_Benchmarks.html (accessed: 15.02.2025).

[18] Newman S. Building microservices. Sebastopol, O'Reilly Media, 2021.

[19] Li S., Zhang H., Jia Z., et al. Understanding and addressing quality attributes of microservices architecture: a systematic literature review. *Inf. Softw. Technol*., 2021, vol. 131, art. 106449. DOI: https://doi.org/10.1016/j.infsof.2020.106449

[20] Velepucha V., Flores P. A survey on microservices architecture: principles, patterns and migration challenges. *IEEE Access*, 2023, vol. 11, pp. 88339–88358. DOI: https://doi.org/10.1109/ACCESS.2023.3305687

[21] Newman S. Monolith to microservices. Sebastopol, O'Reilly Media, 2019.

[22] Waseem M., Liang P., Shahin M. A systematic mapping study on microservices architecture in devops. *J. Syst. Softw*., 2020, vol. 170, art. 110798.
DOI: https://doi.org/10.1016/j.jss.2020.110798

[23] Leroy P., Morato P.G., Pisane J., et al. IMP-MARL: a suite of environments for large-scale infrastructure management planning via MARL. *NeurIPS*, 2024. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/hash/a7a7c0c92f195cce85f99768621ac6c0-Abstract-Datasets_and_Benchmarks.html (accessed: 15.02.2025).

[24] Bacchiani L., Bravetti M., Giallorenzo S., et al. Microservice dynamic architecture-level deployment orchestration. In: *Coordination Models and Languages*. Cham, Springer Nature, 2021, pp. 257–275. DOI: https://doi.org/10.1007/978-3-030-78142-2_16

[25] Vinyals O., Ewalds T., Bartunov S., et al. Starcraft II: a new challenge for reinforcement learning. arXiv:1708.04782. DOI: https://doi.org/10.48550/arXiv.1708.04782

[26] Terry J., Black B., Grammel N., et al. Pettingzoo: gym for multi-agent reinforcement learning. *NeurIPS*, 2021, vol. 34, pp. 15032–15043.

[27] Fan J., Wang Z., Xie Y., et al. A theoretical analysis of deep Q-learning. *Proc. 2nd Conf. on Learning for Dynamics and Control*, 2020, pp. 486–489.

[28] Bolshakov V.E., Alfimtsev A.N. Hierarchical method for cooperative multiagent reinforcement learning in Markov decision processes. *Dokl. Math*., 2023, vol. 108, no. 2S, pp. S382–S392. DOI: https://doi.org/10.1134/S1064562423701132

[29] Ozdaglar A., Sayin M.O., Zhang K. Independent learning in stochastic games. In: *International Congress of Mathematicians*, 2021, pp. 5340–5373.
DOI: https://doi.org/10.4171/icm2022/152

[30] Zhang Z., Yang J., Zha H. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. arXiv:1909.10651.
DOI: https://doi.org/10.48550/arXiv.1909.10651

[31] Morgunov E.F., Alfimtsev A.N. The “stag hunt” social dilemma in multi-agent reinforcement learning. *REEPE*, 2024.
DOI: https://doi.org/10.1109/REEPE60449.2024.10479770

[32] Zheng H., Zhao W., Yang J., et al. QoS analysis for web service compositions with complex structures. *IEEE Trans. Serv. Comput*., 2013, vol. 6, no. 3, pp. 373–386.
DOI: https://doi.org/10.1109/TSC.2012.7

[33] Kuba J.G., Feng X., Ding S., et al. Heterogeneous-agent mirror learning: a continuum of solutions to cooperative marl. arXiv:2208.01682.
DOI: https://doi.org/10.48550/2208.01682

[34] Leibo J.Z., Zambaldi V., Lanctot M., et al. Multi-agent reinforcement learning in sequential social dilemmas. arXiv:1702.03037.
Available at: https://doi.org/10.48550/arXiv.1702.03037

Morgunov E.F. — Assistant, Department of Information Systems and Telecommunications, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Alfimtsev A.N. — Dr. Sc. (Eng.), Head of the Department of Information Systems and Telecommunications, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Please cite this article in English as:

Morgunov E.F., Alfimtsev A.N. Multi-agent microservice architecture. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2025, no. 2 (151), pp. 78–101 (in Russ.). EDN: TPNAZP