سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)

Fall 2024, 5 (3), 101-113
DOR:

Received: 26 June 2024
Accepted: 6 Aug 2024

مقاله پژوهشی

Task Scheduling in Cloud Computing Using Deep Reinforcement Learning

Behnam Salari Hamzehkhani[1], Mehdi Akbari[2]*, Faramarz Safi-Esfahani[3], Behrang Barekatain[4]

1. Ph.D. Student, Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.
2. Assistant Professor, Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.
**Corresponding Author*, mehdi.akbari@iaun.ac.ir
3. Associate professor, Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.
4. Associate professor, Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.

Abstract

Introduction: Task scheduling and energy consumption are important issues in cloud computing. Failure to use an appropriate scheduling approach in cloud computing may result in high energy consumption and low resource efficiency. Due to the dynamics and limitations of cloud resources to execute diverse and time-varying requests from users, an effective scheduling mechanism is required that adapts to dynamic system conditions. Deep learning, an applied method for dealing with cloud computing resource management problems, has been considered as an innovative idea in recent years. Among the deep learning algorithms used in this field, we can mention DQL, ERDQL, and DRL-LSTM algorithms, which have achieved acceptable results in large datasets. However, these algorithms have problems such as unrealistic reward estimation, long-term training, random sampling of memory, and lack of proper use of state space, which affect the results obtained from their use in optimization problems.

Method: In this research, an improved deep reinforcement learning algorithm for task scheduling in cloud computing is proposed to improve the problems of unrealistic reward estimation, long-term training, and random sampling from memory. Also, the Markov decision process structure in the modeling part is defined in such a way that it can mitigate the problems of not using the state space properly.

Results: The proposed algorithm has been evaluated with two scenarios including a synthetic dataset and GOCJ (Google Cloud Jobs) dataset. The results of the evaluations in the simulated environment show that the proposed algorithm has achieved better results in less time in all evaluated parameters.

Discussion: In this paper, an improved algorithm based on PERDQN is presented. Then, with a proper definition of the cloud environment based on the Markov decision process, the proposed algorithm is used to schedule tasks on virtual machines in cloud computing. The results show that using the proposed algorithm, response time, waiting time, throughput, MakeSpan and the number of SLA violations are improved, which is more significant on a larger scale.

Keywords: Task scheduling, cloud computing, deep reinforcement learning, Markov decision process

سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)
انجمن علمی تجارت الکترونیکی ایران
واحد نجف‌آباد

زمانبندی کارها در محاسبات ابری با استفاده از یادگیری عمیق تقویتی

تاریخ دریافت: ۰۶/ ۰۴/ ۱۴۰۳
تاریخ پذیرش: ۱۶/ ۰۵/ ۱۴۰۳

دورهٔ پنجم، پاییز ۱۴۰۳
شمارهٔ سوم، صص: ۱۰۱-۱۱۳

بهنام سالاری‌حمزه‌خانی[۱]، مهدی اکبری[۲*]، فرامرز صافی‌اصفهانی[۳]، بهرنگ برکتین[۴]

۱. دانشجوی دکتری، گروه مهندسی کامپیوتر، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد ، ایران. salaribe@yahoo.com
۲. استادیار، گروه مهندسی کامپیوتر، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد ، ایران. (نویسندهٔ مسئول) mehdi.akbari@iaun.ac.ir
۳. دانشیار، گروه مهندسی کامپیوتر، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد ، ایران. fsafi@iaun.ac.ir
۴. دانشیار، گروه مهندسی کامپیوتر، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد ، ایران. Behrang_Barekatain@iaun.ac.ir

چکیده: زمان‌بندی وظایف و مصرف انرژی از جمله مسائل مهم در محاسبات ابری هستند. عدم استفاده از یک رویکرد زمان‌بندی مناسب در محاسبات ابری ممکن است باعث مصرف انرژی بالا و بهره‌وری پایین منابع شود. به علت پویایی و محدودیت منابع ابری برای اجرای درخوا ست‌های متنوع و متغیر با زمان کاربران، یک مکانیزم زمان‌بندی مؤثر مورد نیاز ا ست که خود را با شرایط پویای سی ستم وفق دهد. یادگیری عمیق، یک روش کاربردی برای مقابله با مشکلات مدیریت منابع رایانش ابری به عنوان یک ایده نوآورانه در سال‌های اخیر مورد توجه قرارگرفته‌است. از جمله الگوریتم‌های یادگیری عمیق استفاده‌شده در این زمینه می‌توان از الگوریتم‌های DQL، ERDQL و DRL-LSTM نام برد که در دیتا ست‌های بزرگ به نتایج قابل‌قبولی ر سیده‌اند. ولی این الگوریتم‌ها دارای م شکلاتی از جمله تخمین غیر واقعی پاداش، آموزش طولانی مدت، نمونه‌برداری ت صادفی از حافظه و عدم ا ستفاده منا سب از فضای حالت می‌با شند که بر نتایج حاصل از استفاده آن‌ها در مسائل بهینه‌سازی مؤثر است. در این تحقیق یک الگوریتم یادگیری عیمق تقویتی بهبودیافته برای زمانبندی کارها در محا سبات ابری ارائه شده‌ا ست تا م شکلات تخمین غیرواقعی پاداش، آموزش طولانی‌مدت و نمونه‌برداری ت صادفی از حافظه را بهبوددهد. همچنین ساختار فرآیند ت صمیم‌گیری مارکوف در ق سمت مدل سازی به گونه‌ای تعریف‌می شود که م شکلات عدم استفاده مناسب از فضای حالت را تعدیل نماید. لذا با استفاده از الگوریتم پیشنهادی نتایج مورد انتظار این است که منجر به بهبود زمان پاسخ، زمان انتظار، برون‌ده، زمان‌کل کارها و تعداد نقض SLA گردد. در این مقاله، الگوریتم توسعه‌یافته با م‌جموعه داده GOCJ (Google Cloud Jobs) مورد ارزیابی قرارمی‌گیرد.

واژه‌های کلیدی: زمانبندی کارها، محاسبات ابری، یادگیری عمیق تقویتی، فرآیند تصمیم‌گیری مارکوف.

۱. مقدمه

هدف اصلی زمان‌بندی کار، به حداقل رساندن اتلاف زمان و به حداکثر رساندن عملکرد است به‌طوری که زمان پاسخ کار دریافت‌شده در محدوده قابل‌قبول بوده و بهره‌وری منابع نیز رعایت‌شده‌باشد[۱]. روش‌های رایج در زمان‌بندی دارای معایبی از قبیل پیچیدگی زمانی زیاد و افزایش زمان اجرای برنامه است. بنابراین، روشی مناسب است که تولید زمان اجرای کل کمینه گردد[۲-۴]. زمان‌بندی کارها با الگوریتم‌هایی مانند اکتشافی سنتی قادر به تطبیق سریع با تغییر و رسیدن کار تصادفی نیستند[۵]. ساختار پیچیده کارها با توالی پردازش انعطاف‌پذیر تصمیم‌گیری را مشکل‌نموده‌است. رویکردهای موجود در برآوردن الزامات دینامیک و پاسخ سریع و یا برای مواجهه با وظایف با توالی‌های پردازش انعطاف‌پذیر مشکل دارند[۶].

با توجه به اینکه بارکاری برنامه های مبتنی بر ابر غیرقابل پیش‌بینی و متغیر هستند، انجام زمان‌بندی کار بصورت آنلاین برای استفاده در محیط ابری مناسب است و روش‌های مبتنی بر RL[۱] برای این سناریو مطلوب هستند، زیرا نیازی به دانستن حجم کاری کاربر، انتقال وضعیت و سایر اطلاعات سیستم اصلی ندارند. عامل RL در طول زمان تصمیمات مربوط به اختصاص کار به ماشین مجازی[۲] را دنبال‌می‌کند و بنابراین داده‌های آموزشی عظیمی را جمع‌آوری کرده تا یاد بگیرد که چگونه در آینده تصمیمات بهتری بگیرد و برای حل مسائل بهینه‌سازی پیچیده عملکرد مناسبی دارد و در طیف وسیعی از مسائل بهینه‌سازی قابل‌استفاده است[۷-۹]. به‌طور کلی استفاده از تکنیک یادگیری عمیق تقویتی در تحقیقات اخیر جهت زمانبندی کارها در محاسبات ابری مورد توجه قرارگرفته است[۱۰-۱۲]. الگوریتم‌های یادگیری عمیق تقویتی در زمان‌بندی ابری به‌علت اینکه پویایی محیط را در نظرگرفته و موجب بهبود عملکرد محاسبات ابری در بلندمدت می‌گردند لذا مبنای این تحقیق قرار گرفته‌است.

یادگیری تقویتی نوعی از روش‌های یادگیری ماشین است که یک عامل را قادر به یادگیری در محیطی تعاملی با استفاده از آزمون و خطا و استفاده از بازخوردهای اعمال و تجربیات خود می‌سازد و از پاداش‌ها و تنبیه‌ها به عنوان سیگنال‌هایی برای رفتار مثبت و منفی استفاده‌می شود. یادگیری عمیق تقویتی، ترکیبی از یادگیری تقویتی و یادگیری‌عمیق است که در آن عامل سعی در حداکثر کردن پاداش دارد. از جمله این الگوریتم‌ها SARSA و learning-Q است. learning-Q یک الگوریتم معروف و مستقل از مدل برای یادگیری تقویتی است. تمایز این الگوریتم با سایر الگوریتم‌ها در استراتژی‌های جست‌وجوی آن محسوب‌می‌شود. به‌دلیل اینکه اقدام بعدی با هدف حداکثر کردن وضعیت بعدی value-Q انتخاب شده‌است، لذا learning-Q یک روش مستقل از سیاست است. این روش حول محور به‌روز رسانی ارزش‌های Q است که بر اساس معادله بلمن می‌باشد. [۱۳].

در این الگوریتم جدولی برای Q ها تشکیل و بر اساس آن تصمیم‌گیری انجام‌می شود. با توسعه این الگوریتم برای رفع مشکلات آن، الگوریتم LQD[۳] مورد استفاده قرارگرفت. در این الگوریتم فرمول‌های QL وجودداشته و به‌جای جدول Qها از شبکه عصبی استفاده شده‌است. ورودی شبکه عصبی حالت‌ها و خروجی آن مقادیر Q به ازای هر عمل می‌باشد. در این الگوریتم از TD[۴] برای محاسبه خطا استفاده‌می‌شود[۱۴, ۱۵]. الگوریتم DQN دارای ایراداتی از جمله تخمین غیرواقعی پاداش می‌باشد[۱۶].

سپس الگوریتم ERDQL[۵] توسعه‌یافت در این الگوریتم با ذخیره تجربیات قبلی در حافظه و نمونه‌برداری تصادفی از آن جهت آموزش شبکه عصبی، کارایی الگوریتم DQL بهبود یافته و پایدارتر می‌باشد. از دو شبکه عصبی یکی برای محاسبه مقادیر Q در حالت فعلی و دیگری برای حالت بعدی استفاده‌می‌شود. در این الگوریتم در هر اپیزود، تجربیات عامل در حافظه Replay ذخیره‌می‌شوند و از این حافظه به‌صورت تصادفی جهت آموزش شبکه نمونه‌برداری می‌شود. در ابتدای نمونه‌برداری محیط کاوش شده و یک عمل به‌صورت تصادفی انتخاب‌می شود و در مراحل بعدی بر اساس الگوریتم حریصانه عمل با بالاترین مقدار Value-Q انتخاب‌می شود که درواقع یک off-trade بین استخراج و اکتشاف است[۱۷, ۱۸].

بنابراین، الگوریتم های یادگیری تقویتی عمیق که تاکنون برای زمان‌بندی وظایف در رایانش ابری استفاده‌شده‌اند، دارای مزایا و معایبی هستند. به عنوان مثال، در الگوریتم NDQ، تخمین بیش از حد پاداش و استفاده از شبکه یکسان برای انتخاب عمل و ارزیابی آن، ممکن است باعث نتایج غیربهینه شود. الگوریتم ERDQN ممکن است به دلیل نمونه‌برداری تصادفی از حافظه، دوره آموزشی طولانی داشته‌باشد و الگوریتم‌های ترکیبی، پیچیدگی محاسباتی را در محیط ابری افزایش می‌دهند. برای رفع مشکل نمونه‌برداری تصادفی از حافظه در الگوریتم ERDQL که ممکن است تجربیات مفید کمتر یا اصلاً مورد استفاده‌قرارنگیرند الگوریتم PERDQL[۶] پیشنهادگردید. لذا با اولویت‌بندی تجربیات توسط درخت باینری و قرارگیری اولویت‌ها در برگ‌های این درخت یک آرایه تشکیل‌می‌شود که به برگ‌های درخت اشاره‌نموده و حاوی تجربیات است. برای پیاده‌سازی این الگوریتم تجربیات قبلی در نوعی از حافظه با ظرفیت محدود ذخیره‌می‌گردند و در طول فرآیند یادگیری، تعدادی از رکوردهای ذخیره‌شده در حافظه بر اساس اولویت انتخاب‌می‌شوند[۱۹, ۲۰]. در این الگوریتم با وجود اینکه تا حدودی مشکلات الگوریتم‌های قبلی رفع شده‌است اما برای اطمینان از کارایی آن برای زمان‌بندی وظایف در محیط محاسبات ابری نیازمند بهبود می‌باشد. در الگوریتم پایه، وزن‌های شبکه هدف به صورت دوره‌ای تغییرمی‌کنند. وقتی وزن های شبکه ها با یکدیگر مقایسه‌می‌شوند بدلیل به‌روزرسانی وزن‌های شبکه هدف در هر d تکرار الگوریتم، باعث افزایش تعداد تکرار مورد نیاز برای همگرایی و در نتیجه افزایش زمان یادگیری می‌شود. این مسأله برای محیط‌های پیچیده و بزرگ مانند محاسبات ابری اهمیت بیشتری دارد.

یادگیری تقویتی معمولاً در قالب فرآیند تصمیم‌گیری مارکوف مدلسازی[۷] می‌شود که یک چارچوب ریاضی است برای مدل‌سازی تصمیم‌گیری، در شرایطی که نتایج تا حدودی تصادفی و تحت کنترل

تصمیم‌گیر می‌باشد. این چارچوب برای مطالعه طیف گسترده‌ای از مسائل بهینه‌سازی که از طریق برنامه‌نویسی پویا حل‌می‌شوند مفید است. این فرآیند شامل پنج عنصر حالت[۸]، عمل[۹]، احتمال[۱۰]، پاداش[۱۱] و ضریب کاهش[۱۲] می‌باشد[۲۱]. در این تحقیق برای تشریح محیط در یادگیری تقویتی مورد استفاده قرارگرفته‌است که هرچه خصوصیات بیشتری از کارها در نظر گرفته‌شود فضای حالت فرآیند تصمیم‌گیری مارکوف کامل‌تر می‌شود به‌عنوان نمونه یکی از دلایل شکست یک وظیفه، تخصیص نامناسب کارها و عبور از مهلت زمانی[۱۳] می‌باشد[۲۲]. بنابراین مهلت زمانی کارها در این تحقیق در نظر گرفته‌شده‌است. سپس از طریق طراحی تابع پاداش، مدل را می‌توان برای اهداف مختلف و بارهای کاری بهینه‌کرد.

با توجه به مطالب عنوان شده، اگر بتوان به‌روزرسانی شبکه هدف را در هر تکرار الگوریتم یادگیری عمیق انجام‌داد آنگاه همگرایی در الگوریتم PERDQL افزایش یافته و در زمان کوتاه‌تری نسبت به الگوریتم پایه آموزش دیده و نتایج بهتری نیز به‌دست خواهدآمد. این موضوع برای زمانبندی کارها در محیط محاسبات ابری حائز اهمیت است. موضوع دیگری که در این تحقیق به آن پرداخته‌شده‌است چگونگی تعریف محیط با استفاده از MDP برای الگوریتم پیشنهادی است به‌طوری‌که ویژگی‌های بیشتری از کارها را پوشش‌دهد.

۲. کارهای مرتبط

در سال‌های اخیر تحقیقات متعددی در رابطه با زمان‌بندی کارها در محیط محاسبات ابری با استفاده از الگوریتم‌های یادگیری ماشین انجام شده‌است. از جمله الگوریتم‌های یادگیری ماشین که برای زمانبندی وظایف در ابر به جواب‌های قابل‌قبولی دست‌یافته‌اند می‌توان به یادگیری عمیق تقویتی اشاره‌نمود که در اغلب آن‌ها از مشتقات الگوریتم Learning_Q استفاده شده‌است[۲۳].

در مقاله ارائه‌شده توسط Wei و همکاران[۸] یک چارچوب تصمیم‌گیری مبتنی بر رویداد جهت زمان‌بندی وظایف بر روی ماشین‌های مجازی بیان‌شده‌است. بدین صورت که هر تصمیم زمان‌بندی پس از ورود درخواست کار جدید گرفته‌می شود. با توجه به این‌که دستیابی به مدیریت منابع و بهینه سازی عملکرد در محیط پویا و نامطمئن برای ارائه‌دهندگان برنامه‌های مبتنی بر ابر یک چالش بزرگ است لذا به تخصیص کارها به منابع مناسب و تأمین نیازمندی‌های QOS مدنظر کاربر پرداخته‌شده‌است. در این مقاله منظور از QOS زمان پاسخ کارها می‌باشد که از مجموع زمان مورد نیاز برای اجرای کار روی ماشین مجازی و زمان سپری‌شده در صف انتظار به‌دست می‌آید. در پایان نیز با سناریوهای مختلف از کارهایی که بیشتر محاسباتی یا ورودی و خروجی هستند نتایج مقایسه‌گردیده‌است. در این تحقیق علی‌رغم مدلسازی مناسب محیط در MDP، اما در قسمت الگوریتم، زمان آموزش زیادی مورد نیاز است.

در مقاله Swarup و همکاران [۲۴] از الگوریتم یادگیری عمیق double deep q learning جهت زمانبندی کارها در ابر استفاده شده‌است. برای پیاده سازی روش پیشنهادشده در این مقاله، محیط شبیه‌سازی کلودسیم و پایتون مورد استفاده قرارگرفته‌است. برای مجموعه اولیه تسک‌ها به صورت تصادفی زمانبندی انجام شده و برای تسک‌ها در مراحل بعدی سریع‌ترین ماشین مجازی انتخاب‌می‌شود برای این کار مدل Sequential با چهار لایه dense مورد استفاده قرارمی‌گیرد و در نهایت سعی در کاهش زمان و هزینه اجرا دارد. با توجه به تحلیل نویسندگان این مقاله انجام زمانبندی سریع همچنان به‌عنوان چالش مطرح است.

در مقاله Rjoub و همکاران[۲۵] چهار الگوریتم RL, DQN, RNN-LSTM, LSTM-DRL برای زمانبندی در محاسبات ابری مورد بررسی قرارگرفته‌است. بعد از حدود ۱۰۰۰ تکرار، الگوریتم‌ها آموزش دیده و برای موارد جدید با داده تست، دقت الگوریتم بررسی شده‌است. در نهایت با نمودار مشخص شده‌است که LSTM-DRL عملکرد بهتری نسبت به بقیه دارد و در کل دقت آموزش در این چهار الگوریتم به ترتیب زیر بیان‌شده‌است:

RL<LSTM-RNN <DQN < LSTM-DRL

در حالت کلی با استفاده از روش‌های ترکیبی در شبکه‌های عصبی عمیق، دقت و عملکرد مدل بهبود می‌یابد[۲۶]. لذا با الگوریتم ترکیبی با DRL عملکرد بهتری جهت زمانبندی ابر نسبت به DQL به‌دست آمده‌است. اگرچه نتایج بسیار امیدوارکننده‌ای به‌دست آمده‌است، اما مسئله‌ای که باید مورد توجه قرارگیرد زمان زیاد مورد نیاز رویکرد LSTM-DRL است که ناشی از این واقعیت است که لایه LSTM باید به عقب برگردد و تاریخچه کامل حالت‌ها را بررسی‌نماید.

تعداد زیادی از مقالات نیز از الگوریتم‌های ترکیبی برای زمانبندی کارها در محاسبات ابری استفاده شده‌است . به‌عنوان نمونه در مقالات [۲۷] و[۲۸] الگوریتم A2C برای زمانبندی کارها پیشنهاد شده‌است این الگوریتم متشکل از دو عامل است یکی از آن‌ها مسئول یادگیری خودکار زمانبندی و دیگری خطای تخمین را کاهش می‌دهد. در واقع از دو شبکه عصبی یکی برای عمل و دیگری برای تخمین ارزش مورد استفاده قرارگرفته‌است. به‌طور کلی، استفاده از الگوریتم‌های ترکیبی برای زمان‌بندی وظایف، سربار محاسباتی در محیط ابری دارند.

در مقاله Li وهمکاران[۲۹] از الگوریتم DDQL جهت زمان‌بندی در محیط ابر استفاده‌شده‌است. زمان‌بند یک وظیفه را به صف ماشین مناسب اختصاص‌می‌دهد. فضای حالت در این تحقیق بر اساس خصوصیات کار و ماشین مجازی که کار به آن اختصاص داده شده‌است تعیین‌می‌شود. عمل مورد نظر انتخاب ماشین مجازی است و محاسبه پاداش بر اساس دو هدف زمان‌کل[۱۴] و هزینه تعریف شده‌است که این اهداف با هم تضاد دارند و می‌بایست بین این دو تعادل برقرار شود. به عنوان مثال، یک استراتژی کم هزینه، میزان استفاده از ماشین مجازی با قدرت پردازش پایین را افزایش می‌دهد که زمان

صرف‌شده را افزایش می‌دهد. اگر زمان‌کل هدف اصلی باشد میزان استفاده از ماشین مجازی با قدرت پردازش بالا افزایش خواهدیافت. بنابراین با استفاده از وزن‌ها تعادل بین این دو برقرار می‌شود. لذا پاداش به‌صورت رابطه (۱) می‌باشد [۲۰].

$$R=(r^1 * r^1_{rate} + r^2_i * r^2_{rate}) * \max(1-r^3 , 0.1) - r^3 \quad (۱)$$

که در معادله فوق r^1 زمان‌کل، r^2 هزینه کار، r^3 زمان انتظار کار و r_{rate} وزن جهت کنترل پاداش می‌باشد. به‌طور خلاصه در این مقاله از الگوریتم DDQL که تکامل‌یافته DQL می‌باشد استفاده شده‌است. الگوریتم DDQL تجربیات قبلی را در حافظه ذخیره‌کرده و سپس با نمونه‌برداری تصادفی شبکه عصبی آموزش‌دیده و موجب بهبود عملکرد الگوریتم DQL می‌شود. اما شبکه عصبی را با نمونه‌گیری تصادفی آموزش‌می‌دهد که موجب افزایش زمان یادگیری می‌شود.

همچنین در مقاله های [۳۰] و [۳۱] از الگوریتم یادگیری عمیق تقویتی ERDQL برای بهینه‌سازی زمان‌کل و مصرف انرژی استفاده شده‌است. در مرجع [۳۰] زمان‌کل از مجموع دو زمان اجرا و انتظار محاسبه‌می شود، جهت محاسبه مصرف انرژی نیز ابتدا بهره‌وری بر اساس تقسیم تعداد ماشین های مجازی فعال روی سرور بر حداکثر تعداد ماشین مجازی قابل راه‌اندازی روی یک سرور به‌دست‌می‌آید و در نهایت مصرف انرژی در دوحالت استاتیک و داینامیک با استفاده از بهره‌وری، محاسبه‌شده و بر اساس تعداد سرورهای فیزیکی در دیتا سنتر با هم جمع‌می شوند. با توجه به اینکه دو هدف مطرح بوده لذا پاداش بر اساس زمان‌کل و میزان مصرف انرژی است و در پایان نرمالسازی جهت قرار دادن پاداش در بازه ۰ و ۱ انجام‌می شود. در این مقاله، فضای حالت تعریف شده برای زمان بندی وظایف در محیط ابری به صورت محدود تعریف شده‌است. در تحقیق [۳۱] نیز روشی جهت کاهش هزینه انرژی در دیتاسنترها و زمانبندی کارها ارائه‌شده‌است که موجب بهبود قابل‌ملاحظه‌ای در زمینه مصرف انرژی شده و همگرایی مناسبی دارد. دو مرحله یکی برای اختصاص کارها به یکی از نواحی سرورها و دیگری برای مشخص‌کردن سرور دارد. برای رسیدن به اهداف مشخص‌شده از DQN Replay Experience استفاده‌شده‌است که در آن اطلاعات در حافظه ذخیره شده و با نمونه برداری تصادفی از آن شبکه عصبی آموزش می بیند. از دو شبکه عصبی یکی برای محاسبه eQ_Valu در حالت فعلی و دیگری برای محاسبه در حالت بعدی مورد استفاده قرارمی‌گیرد و سپس به‌روزرسانی انجام‌می شود. که در نهایت پایداری این الگوریتم را نسبت به DQN پایه افزایش می‌دهد، اما به زمان آموزش بیشتری نیاز دارد. به‌طور کلی، استفاده از الگوریتم‌های یادگیری عمیق تقویتی برای دو هدف زمان‌بندی وظایف در محیط ابری و کاهش مصرف انرژی مرکز داده، نیازمند تعادل در تعریف تابع پاداش است.

با توجه به کارهای گذشته در خصوص استفاده از الگوریتم‌های یادگیری عمیق تقویتی برای زمانبندی کارها در محاسبات ابری که به نتایج قابل‌قبولی نسبت به سایر روش‌های زمانبندی رسیده‌اند. بنابراین مبنای این تحقیق قرارگرفته‌است ولی همچنان موارد یادشده در قسمت مقدمه پابرجاست. لذا بهبود الگوریتم یادگیری عمیق تقویتی مورد استفاده برای زمانبندی کارها در ابر با توجه به پارامترهای مورد نظر و همچنین تعریف مناسب محیط به‌طوری‌که ویژگی‌های بیشتری از کارها از جمله مهلت زمانی را شامل‌شود در این تحقیق مورد بررسی قرارگرفته‌است.

۳. الگوریتم پیشنهادی

الگوریتمDQN در محیط های دارای نویز تمایل به برآورد بیش از حد پاداش دارد. همچنین در این الگوریتم انتخاب عمل و ارزیابی عمل توسط یک شبکه یکسان انجام‌می شود در نتیجه آموزش شبکه عصبی پایدار و بهینه نیست. به‌دلیل اینکه در الگوریتم DQN انتخاب بهترین عمل برای حالت بعدی بر اساس حالت با بالاترین مقدار Value-Q قابل اطمینان نیست لذا با استفاده از الگوریتم DQN Double انتخاب عمل از ارزیابی عمل جدا شده‌است و از دو شبکه عصبی برای این‌کار استفاده می‌شود. شبکه اصلی DQN بهترین عمل برای حالت بعدی را انتخاب‌می‌کند و شبکه دیگر Value-Q target را محاسبه‌می‌کند. معادله نهایی به‌صورت رابطه (۲) است[۳۲, ۳۳].

$$Q_1(S, A)= Q_1(S, A) + \alpha (R + \gamma Q_2(S' + \text{argmax } Q_1(S', a)) - Q_1(S, A)) \quad (۲)$$

همچنین در برخی مواقع مقدار دقیق هر عمل مورد نیاز نیست بنابراین یادگیری تابع Value-State برای شبکه عصبی کافی می‌باشد لذا در الگوریتم DQN Dueling شبکه عصبی در لایه آخر به دو قسمت تقسیم‌می‌شود. یک قسمت برای تخمین value-state در حالت موردنظر و دیگری برای تعیین مزایای هر عمل، سپس با هم ترکیب می‌شوند لذا معادله Value-Q مطابق رابطه (۳) است[۳۴].

$$Q(s, a) = V(s) + A(s, a) \quad (۳)$$

که در معادله فوق V(s) تابع ارزش و A(s,a) مزیت یک عمل نسبت به سایر عمل‌ها را مشخص‌می‌نماید.

تکرار تجربه یک تکنیک کلیدی در یادگیری عمیق تقویتی است که عامل برای یادگیری از خاطرات قبلی استفاده‌کرده و سرعت یادگیری را با همبستگی مطلوب افزایش می‌دهد[۳۵]. در الگوریتم‌های مختلفی از جمله ERDQL و NTM[۱۵] از ماژول حافظه جهت ذخیره تجربیات قبلی و استفاده مجدد از آن‌ها استفاده‌شده‌است[۳۶, ۳۷]. ایرادی که در الگوریتم ERDQN وجود دارد نمونه برداری تصادفی از حافظه می‌باشد که در الگوریتم PERDQL این مشکل برطرف شده‌است. این الگوریتم در حالت کلی مشابه الگوریتم ERDQL است با این تفاوت که در الگوریتم ERDQL انتخاب نمونه به‌صورت تصادفی از حافظه انجام‌می‌شود و ممکن است تجربیات مفید کمتر یا اصلاً مورد استفاده قرارنگیرند لذا با اولویت‌بندی تجربیات توسط درخت باینری mTreeSu و قرارگیری اولویت‌ها در برگ‌های این درخت یک آرایه تشکیل‌می‌شود

که به برگ‌های درخت اشاره نموده که حاوی تجربیات است. اولویت نمونه‌ها در الگوریتم PERDQL بر اساس معادله (۴) تعیین می‌شود که در این رابطه ε مقداری ثابت است که از صفر شدن احتمال نمونه‌ها جلوگیری‌می‌کند[۱۹, ۳۸].

$$P_i = |\delta_i| + \varepsilon \quad (۴)$$

در معادله فوق δ_i خطای TD را نشان‌می‌دهد که بر اساس رابطه(۵) محاسبه‌می‌شود

$$\delta_I = R_t + \gamma Q[S_{t+1}, \text{argmax}(Qs_{t+1}, a; \theta_t); \overline{\theta}_t] - Q(s_t, a_t; \theta_t) \quad (۵)$$

در معادله فوق، θ_t و $\overline{\theta}_t$ به ترتیب مربوط به وزن شبکه های اولیه و هدف هستند. احتمال انتخاب یک نمونه بر اساس معادله (۶) قابل محاسبه است که در این رابطه α مقداری بین ۰ و ۱ بوده و جهت کنترل میزان اولو یت مورد استفاده قرارمی‌گیرد. برای جلوگیری از over-fitting به نمو نه ها با اولو یت بالا که ز یاد در شبکه استفاده‌می شوند وزن کوچک بر اساس معادله (۷) اختصاص داده‌شود که در آن N سایز بافر و β نشان‌دهنده بایاس می‌باشد.

$$P_i = p_i^{\alpha} / \Sigma_k p_k^{\alpha} \quad (۶)$$

$$W_i = (1/N * 1/p_i)^{\beta} \quad (۷)$$

برای درج در بافر با توجه به اولویت کارها به زمان o(nlogn) و جهت نمونه‌برداری از آن به زمان o(n) نیاز می‌باشد لذا با توجه به زمان‌بر بودن این عملیات نمی‌توان تمامی بافر را بر اساس اولویت مرتب نمود بنابراین از درخت باینری SumTree ا ستفاده شده و یک شیء حافظه برای این درخت باینری و داده‌ها ساخته‌می‌شود.

بنابراین با استفاده از Dueling Double Deep Q-Network و آنچه که در الگوریتم PERDQL برای ذخیره و باز یابی تجربیات قبلی از حافظه انجام‌شده‌است اکثر مشکلات الگوریتم‌های مورد استفاده تا کنون برای زمانبندی کارها در محا سبات ابری برطرف می‌گردد. اما در الگوریتم Dueling Double Deep Q-Network وزن های Network target در هر d تکرار به‌روزر سانی می شوند در صورتی که پریود تغییر وزن‌های شبکه افزایش یابد موجب همگرایی بهتر الگوریتم شده و به‌صورت تجربی نیز در زمان‌بندی کارها در محاسبات ابری نتایج بهتری به‌دست‌می‌آید. برای این منظور در این مقاله از معادله (۸) استفاده‌می‌شود.

$$w'_i = w'_{i-1} + \varepsilon(w_{i-1} - w'_{i-1}) \quad (۸)$$

که در این معادله w_{i-1} وزن شبکه فعلی و w'_i وزن شبکه Network target در مرحله بعدی است.همچنین در الگوریتم پایه انتخاب action به‌صورت تصادفی با احتمال ثابت ε انجام‌می‌شود که می‌تواند بر کارایی الگوریتم اثرات منفی داشته‌باشد. لذا در الگوریتم پیشنهادی برای ε در ابتدا مقداری بین صفر و یک در نظر گرفته‌شده و سپس با معادله (۹) با توجه به تعداد تکرار لازم برای همگرایی مسأله، به‌روزرسانی انجام‌می‌شود.

$$\varepsilon = \varepsilon - ((\varepsilon / \text{total episode}) * \text{number of episode}) \quad (۹)$$

با ایجاد تغییر فوق در مقدار ε در مراحل اولیه، کاوش بیشتری انجام‌می‌شود و عامل محیطی را به‌خوبی بررسی‌می‌کند، سپس به تدریج کاهش می‌یابد تا از تجربیات به دست آمده بیشتر استفاده‌شود و بهره‌برداری بیشتری حاصل‌شود. با نزدیک شدن به مراحل پایانی اجرای الگوریتم، معادله فوق به صفر همگرا شده و الگوریتم یادگیری عمیق تقویتی به‌صورت حریصانه‌تر عمل‌می‌کند.

بنابراین با اصلاح پریود به‌روزرسانی وزن‌های target Network و مقدار ε برای انتخاب action و همچنین استفاده از Dueling Double Deep Q-Network شبه کد نهایی الگوریتم پیشنهادی بر اساس توسعه الگوریتم PERDQL به‌صورت الگوریتم ۱ است. در این الگوریتم، نمونه‌ها با استفاده از SumTree بر اساس اولویت ذخیره و بازیابی می‌شوند، این اولویت طبق رابطه (۴) تعیین‌می‌شود.

Algorithm 1: Extended Prioritized Experience Replay Deep Q Network (EPERDQN)

Input: minibatch k, replay period K, exponents α and β, update factor ε, budget T

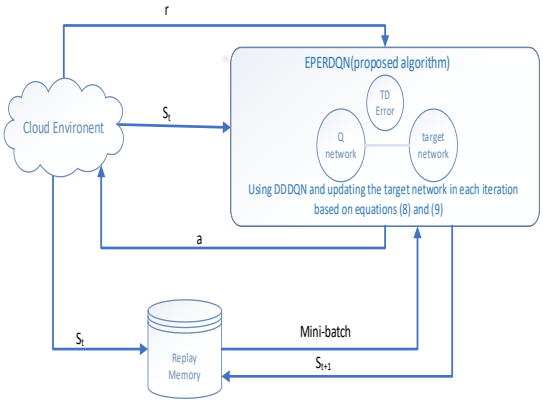
1 Initialize replay memory D, Q-network parameters θ, p1=1
2 **For** t=1 to T **do**:
3 Observe St and choose action At according to ε-greedy policy
4 Store transition(S_t,A_t,R_t,S_{t+1}) in D with maximal priority $P_t=\max_{i<t} P_i$
5 **If** t mod K==0 **do**:
6 **For** j=1 to k **do**:
7 Sample transition $j \sim P(j) = p_j^{\alpha} / \Sigma_i p_t^{\alpha}$
8 Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
9 Compute TD-error $\delta_j = R_j + \gamma_j Q_{target}(S_j, \text{argmax}_a(Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
10 Update transition priority p_j |δ
11 **End For**
12 Update target network in each iteration $\theta'_t = \theta'_{t-1} + \varepsilon(\theta_{t-1} - \theta'_{t-1})$
13 **End If**
14 **End For**

در خط ۱۲ الگوریتم ۱، وزن‌های شبکه هدف بر ا ساس معادلات (۸) و (۹) به‌روز می شوند. با این روش به روز سانی، پارامترهای شبکه هدف تغییر چندانی نمی‌کنند و الگوریتم پایدار است و در هر تکرار آموزش داده‌می‌شود، بنابراین زمان آموزش شبکه عصبی کاهش می‌یابد و سریعتر همگرا می‌شود که برای اهداف مورد نظر در این تحقیق مهم است. بنابراین این الگوریتم برای استفاده در محیط‌هایی با حالت‌های پیچیده و مقیاس های بزرگ مناسب است. در این تحقیق از آن برای زمان‌بندی وظایف در محیط محاسبات ابری استفاده‌شده‌است.

۴. پیاده‌سازی

در محیط محاسبات ابری با درخواست کار جدید، یک نمونه ماشین مجازی برای اجرای آن انتخاب‌می شود در صورتی که ما شین مجازی آزاد باشد کار بلافاصله اجرامی‌شود در غیر اینصورت در صف FIFO قرار

می‌گیرد. در این مدل زمانبندی به‌صورت آنلاین انجام می‌شود. چارچوب روش پیشنهادی در شکل ۱ بیان شده‌است. ابتدا محیط محاسبات ابری بر اساس کارها و ماشین‌های مجازی و خصوصیات مورد نظر Iaas[۱۶] جهت زمان‌بندی، طبق فرآیند تصمیم‌گیری مارکوف پیاده‌سازی می‌شود. سپس بر اساس الگوریتم یادگیری عمیق تقویتی در زمان t مقدار S_t بعنوان ورودی شبکه عصبی به عامل داده‌می شود و عامل عمل a را انجام داده و پاداش r را دریافت‌می‌کند. سپس S_t و a_t و r_t و S_{t+1} به‌عنوان یک رکورد در حافظه ذخیره‌می شود و این کار ادامه‌می‌یابد تا زمانی که تعداد رکوردها در حافظه از مقدار آستانه بیشتر شود و سپس از این حافظه براساس اولویت در درخت تصمیم دودویی برای آموزش مدل نمونه برداری می‌شود.



شکل ۱: چارچوب روش پیشنهادی

طراحی و پیاده‌سازی محیط یادگیری عمیق تقویتی معمولا بر اساس فرآیند تصمیم‌گیری مارکوف انجام‌می‌شود. کارهایی که برای زمانبندی روی ماشین‌های مجازی وارد می شوند دارای خصوصیاتی از جمله اندازه ، زمان اجرا و مهلت زمانی می‌باشند. همچنین مورد دیگری که برای حالت باید در نظرگرفت وضعیت فعلی ماشین‌های مجازی می‌باشد که بر اساس زمان تأخیر کار بر روی هر ماشین مجازی بر اساس بارکاری آن مشخص‌می‌شود. لذا در این مقاله حالت برای محیط از مجموع دو مورد ذکرشده به‌صورت زیر تعریف‌شده‌است. زمان اجرا بر اساس اندازه کار، تعداد پردازنده هر ماشین مجازی و سرعت آن‌ها محاسبه‌می‌شود.

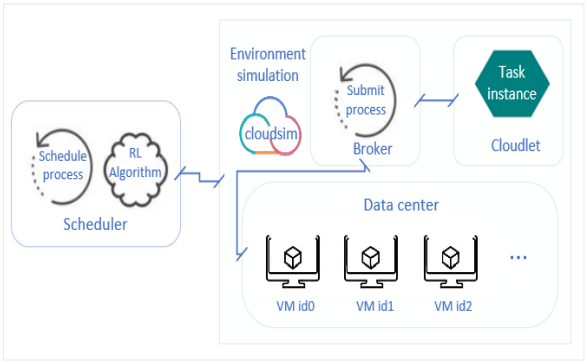
$$State\ Spac = \{job\_size, job\_ExecutionTime, job\_constraints, VMs\_WaitingTime\}$$

عامل می‌تواند هر یک از ماشین‌های مجازی را با توجه به حالت آن انتخاب‌نماید و هر ماشین مجازی محدودیتی در پذیرفتن کارها ندارد بنابراین فضای عمل به‌صورت زیر تعریف‌می‌شود.

$$Action\ Sapce = \{VM_{id^1}, VM_{id^2}, \dots, VM_{id^n}\}$$

عامل در یادگیری عمیق تقویتی سعی دارد که ماکزیمم پاداش را به‌دست بیاورد و به‌عنوان هدف از آن استفاده‌می‌کند. در این تحقیق پاداش در جهت دستیابی به اهداف مورد نظر به‌صورت $(d_i - T_i)/(1/n \sum_{i=1}^{n} |d_i - T_i| + \varepsilon)$ تعریف‌شده‌است. در این معادله d_i نشان‌دهنده مهلت زمانی کار i و T_i نشان‌دهنده جمع زمان تأخیر کار i می‌باشد. کاربران محیط ابری انتظار دارند وظایف آن‌ها در مهلت مقرر انجام شود، بنابراین الگوریتم زمان بندی باید مهلت زمانی را نیز در نظر بگیرد.

در این مقاله ابتدا مطابق چارچوب تعریف شده در شکل ۲ جهت انجام شبیه سازی محیط از کلودسیم استفاده شده است[۳۹]. همچنین از کراس که یک کتابخانه متن باز شبکه عصبی به زبان پایتون بوده و بر روی تنسورفلو قابل‌اجرا می‌باشد جهت پیاده‌سازی شبکه عصبی استفاده شده‌است[۴۰]. کلود سیم یک بسته نرم‌افزاری متن‌باز به زبان جاوا است که برای شبیه سازی محاسبات ابری مناسب بوده و امکانات متنوعی برای مقایسه نتایج اجرای الگوریتم‌های گوناگون بر محیط شبیه‌سازی شده با توجه به پارامترهای مورد نظر دارد. بعد از شبیه سازی محیط در کلود سیم، با فعال سازی listener جاوا، از طریق py4j ارتباط با محیط مورد استفاده برای پایتون برقرار می‌شود. Py4j امکان دسترسی پویا به اشیاء جاوا در یک ماشین مجازی را برای برنامه های پایتون با استفاده از شماره پورت و آدرس شبکه فراهم‌می کند.



شکل۲: چارچوب ارزیابی الگوریتم EPERDQN در محیط شبیه‌سازی شده

در مرحله اول، آزمایش با یک مثال حاوی تعداد کمی از وظایف با داده‌های مصنوعی انجام‌می‌شود، سپس در مرحله دوم، الگوریتم EPERDQN با مجموعه داده GOCJ ارزیابی می‌شود. این مجموعه داده شامل چندین فایل متنی است که هر کدام شامل تعداد مشخصی از وظایف است و به محققان بینشی از رفتار واقعی کاربر می‌دهد[۴۱]. بنابراین، با توجه به ویژگی های ارائه‌شده از این مجموعه داده، حاوی داده‌های لازم برای آزمایش الگوریتم پیشنهادی است.

۵. ارزیابی

در مقالات مختلف، از جمله مراجع [۴۲-۴۴]، مقایسه‌ای بین الگوریتم‌های یادگیری عمیق تقویتی و سایر الگوریتم‌های مورد استفاده برای زمان‌بندی وظایف در فضای ابری بیان شده‌است و مزایای استفاده از الگوریتم DRL برای زمان‌بندی مورد بحث قرار گرفته‌است. بنابراین در این تحقیق مقایسه بین الگوریتم های یادگیری تقویتی انجام شده‌است.

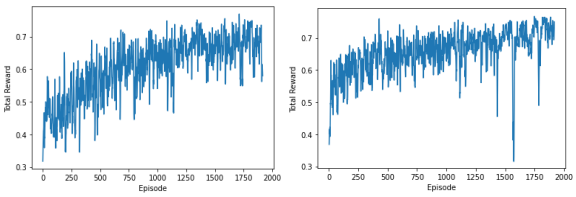
به منظور بررسی عملکرد الگوریتم پیشنهادی در این تحقیق و مقایسه آن با سایر الگوریتم‌های ذکرشده، دو سناریو مورد بررسی قرار گرفته‌است. برای شبیه‌سازی این دو سناریو، شبیه‌ساز کلودسیم با Eclipse IDE استفاده‌شده‌است.

سناریوی ۱: برای شبیه‌سازی در محیط کلودسیم با داده های مصنوعی، فرض شده یک دیتاسنتر با ۴ ماشین مجازی با خصوصیات مشخص از جمله تعداد و سرعت پردازنده، پهنای باند، ظرفیت حافظه و ... وجود دارد در این سناریو تعداد ۱۰ کار درنظر گرفته شده‌است. با تست الگوریتم‌ها با سناریوهای مختلف و بار کاری بالا نتایج مشابهی به‌دست‌آمده‌است. ابتدا محیط بر اساس اطلاعات فوق در کلودسیم ساخته‌می‌شود که پارامترها در جدول ۱ آورده‌شده‌است.

جدول ۱: محیط شبیه‌سازی برای سناریوی۱

type	parameter	Value
Data Center	Number of DC	1
	VM-Scheduler	Time-Shared
Host	Host Number	1
	Processing Element	4000 MIPS
	RAM	2 GB
	Storage	1TB
	Bandwidth	10 GB/s
VM	Number of VMs	4
	Processing Element	220-450 MIPS
	Memory	512MB
	Bandwidth	1 GB/s
	VMM	Xen
Cloudlet	Task Number	10
	Length of Tasks	2000-10000

حال می‌توان الگوریتم EPERDQN را با زبان پایتون پیاده سازی نموده و ارتباط آن با کلودسیم را از طریق py4j برقرار نمود. پس از دو هزار بار تکرار الگوریتم یادگیری عمیق تقویتی نتایج به شرح شکل‌های ۳(a) و ۳(b) به‌دست آمده‌است.



شکل۳(a):اجرای الگوریتم DQN شکل(b) ۳: اجرای الگوریتم EPERDQN

در الگوریتم EPERDQN به نسبت الگوریتم‌های DQN، ERDQN و PERDQN پاداش بالاتری به‌دست آمده‌است. همچنین سریع‌تر از سایر الگوریتم‌های عنوان شده همگرا شده و پایدارتر می‌باشد. این نتایج با تست بارهای کاری مختلف تکرار شده‌است. اثرات اجرای الگوریتم‌ها بر پارامترهای مورد ارزیابی به شرح جدول ۲ می‌باشد.

جدول ۲: مقایسه پارامترهای خروجی بر اساس اهداف تعریف‌شده در سناریوی ۱

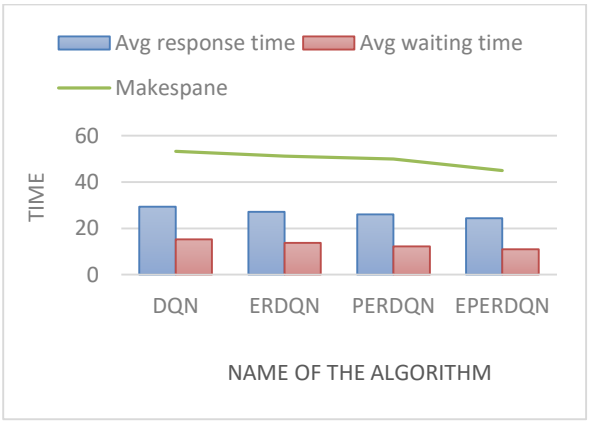
Algorithm	DQN	ERD QN	PER DQN	EPERDQN (proposed algorithm)
Avg response time	29.4	27.1	26	24.4
Avg waiting time	15.3	13.7	12.2	11
Makespan	53.24	51.24	50	45
Throughput	0.19	0.2	0.2	0.22

در این جدول زمان انتظار از تقسیم بار روی ماشین مجازی بر سرعت پردازنده آن به‌دست‌می‌آید. زمان پاسخ از مجموع دو زمان اجرا و انتظار محاسبه‌می شود. زمان‌کل نیز زمان اتمام آخرین کار بر روی آخرین پردازنده را نشان‌می‌دهد[۴۵]. زمان تکمیل نیز بر اساس معادلات (۱۰) و (۱۱) در محیط کلودسیم محاسبه‌می‌شود.

$$execTime = cloudlet.getCloudletLength() / (vm.getMips()*vm.getNumberOfPes()) \quad (۱۰)$$

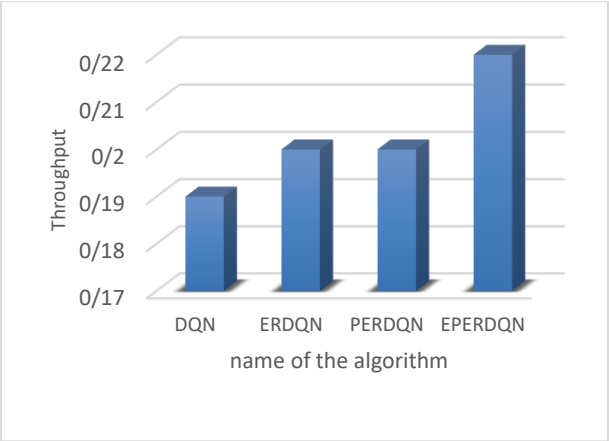
$$completionTime = execTime + waitingTime[vm\_id] \quad (۱۱)$$

با افزایش تعداد کارها درصد بهبود الگوریتم پیشنهادی افزایش می‌یابد. به‌طور کلی الگوریتم EPERDQN مطابق نمودار مقایسه‌ای شکل ۴ عملکرد بهتری در پارامترهای مورد ارزیابی در این تحقیق دارد.



شکل ۴: نمودار مقایسه‌ای پارامترهای خروجی در محیط کلودسیم در سناریوی ۱

توان عملیاتی[۱۷] تعداد وظایف انجام‌شده در واحد زمان را نشان‌می‌دهد. در این تحقیق تعداد cloudlets اجراشده در ثانیه به عنوان توان عملیاتی در نظر گرفته شده‌است. مقایسه بین الگوریتم‌های مورد استفاده در این تحقیق در شکل ۵ نشان داده شده‌است.



شکل ۵: مقایسه توان عملیاتی در سناریوی ۱

بر اساس نتایج حاصل در پیاده‌سازی سناریوی ۱، الگوریتم EPERDQN در تمامی پارامترهای مورد ارزیابی نتایج بهتری به‌دست آورده‌است. همچنین در مدت زمان کمتری آموزش دیده و اهداف مورد انتظار را برآورده‌می‌نماید.

سناریو ۲: برای اطمینان از عملکرد الگوریتم پیشنهادی در محیط‌های واقعی و بزرگ از مجموعه داده GOCJ استفاده‌شده‌است که بر اساس ردیابی بار کاری خوشه گوگل[۱۸] است. مجموعه داده GOCJ در مخزن داده مندلی ذخیره‌می‌شود و شامل ۱۹ فایل متنی است که هر یک از آن‌ها GOCJ_dataset_xxx.text نام دارند که xxx تعداد کارهای موجود در آن فایل را نشان‌می‌دهد و هر ردیف اندازه کار را به MI نشان‌می‌دهد. وظایف موجود در این مجموعه داده بر اساس طول آن‌ها مطابق جدول ۳، به ۵ نوع تقسیم‌می‌شوند[۴۱].

جدول ۳: اندازه کارها در مجموعه داده GOCJ

Job type	Small	Medium	Large	Extra-Large	Huge
Size of jobs(MI *10^3)	15-55	59-99	101-135	150-525	525-900
Distribution	20%	40%	30%	6%	4%

در این تحقیق فایل های موجود در مجموعه داده GOCJ به دو گروه تقسیم‌می‌شوند، یک گروه بین ۱۰۰ تا ۶۰۰ کار به عنوان مجموعه داده با اندازه معمولی[۱۹] است و گروه دیگر شامل ۷۰۰ تا ۱۰۰۰ کار به عنوان مجموعه داده با اندازه بزرگ[۲۰] است[۴۶]. ۵۰ ماشین مجازی برای برای مجموعه داده‌های اندازه معمولی و ۱۰۰ ماشین مجازی برای مجموعه داده‌های اندازه بزرگ استفاده‌می شود. سایر پارامترهای مورد استفاده در این سناریو برای شبیه سازی در کلودسیم مطابق جدول ۴ می‌باشد.

جدول ۴: محیط شبیه‌سازی برای سناریوی ۲

Type	Parameter	value
Data Center	Number of DC	1
	VM-Scheduler	Time-Shared
Host	Host Number	20
	Processing Element	177730 MIPS
	RAM	16 GB
	Storage	2 TB
	Bandwidth	10 GB/s
VM	Number of VMs	5-100
	Processing Element	(3.5-100) * 10^3 MIPS
	Memory	1-4 GB
	Bandwidth	1-10 GB/s
	VMM	Xen
Cloudlet	Task Number	100-1000
	Length of Tasks	(15-900) * 10^3 MI

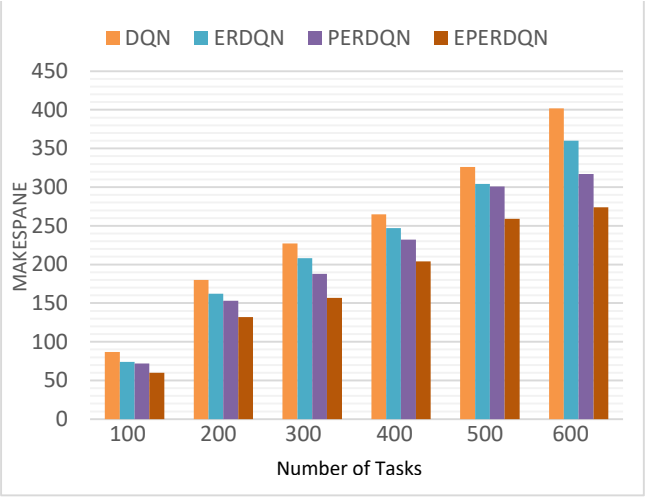
در این سناریو، زمان کل و توان عملیاتی برای دو گروه اندازه معمولی و اندازه بزرگ با مجموعه داده GOCJ ارزیابی‌می شود. نتایج با الگوریتم‌هایDQN ، ERDQN و PERDQN مقایسه‌شده‌اند. جدول ۵ و شکل ۶ نتایج زمان‌کل در الگوریتم EPERDQN را در مقایسه با سایر الگوریتم‌های ذکر شده نشان‌می‌دهد و جدول ۶ و شکل ۷ آنالیز عملکرد توان عملیاتی را برای مجموعه داده با اندازه معمولی نشان‌می‌دهد.

جدول ۵: زمان کل در سناریوی ۲ با استفاده از regular_size در مجموعه داده GOCJ

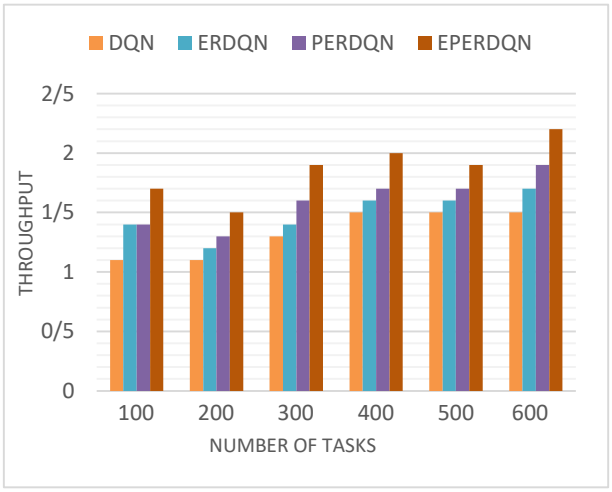
number of tasks	DQN	ERDQN	PERDQN	EPERDQN
100	87	74	72	60
200	180	162	153	132
300	227	208	188	157
400	265	247	232	204
500	326	304	301	259
600	402	360	317	274

جدول ۶: توان عملیاتی در سناریوی ۲ با استفاده از regular_size در مجموعه داده GOCJ

number of tasks	DQN	ERDQN	PERDQN	EPERDQN
100	1.1	1.4	1.4	1.7
200	1.1	1.2	1.3	1.5
300	1.3	1.4	1.6	1.9
400	1.5	1.6	1.7	2
500	1.5	1.6	1.7	1.9
600	1.5	1.7	1.9	2.2



شکل ۶: مقایسه زمان کل در سناریوی ۲ با استفاده از regular_size در مجموعه داده GOCJ



شکل۷: مقایسه توان عملیاتی در سناریوی ۲ با استفاده از regular_size در مجموعه داده GOCJ

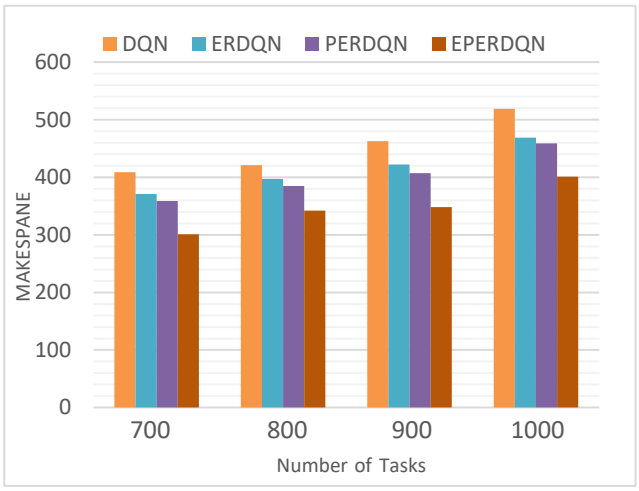
تجز یه و تحل یل تجربی برای گروه ا ندازه بزرگ مجمو عه داده GOCJ در این بخش ارائه‌شـده‌اسـت. مشـابه آزمایش قبلی، عملکرد با ا ستفاده از زمان‌کل و توان عملیاتی اندازه‌گیری‌می شود. تعداد وظایف بین ۷۰۰-۱۰۰۰ برای مجموعه داده با اندازه بزرگ در نظر گرفته شده اسـت. این وظایف بر روی ۱۰۰ ماشـین مجازی زمان‌بندی شـده‌اند. جدول ۷ و شکل ۸ تجزیه و تحلیل مقایسه‌ای مجموعه داده‌های بزرگ را از نظر زمان‌کل ن شان‌می‌دهد، همچنین جدول ۸ و شکل ۹ نتایج را برای توان عملیاتی نشان‌می‌دهد.

جدول ۷: زمان کل در سناریوی ۲ با استفاده از larg_size در مجموعه داده GOCJ

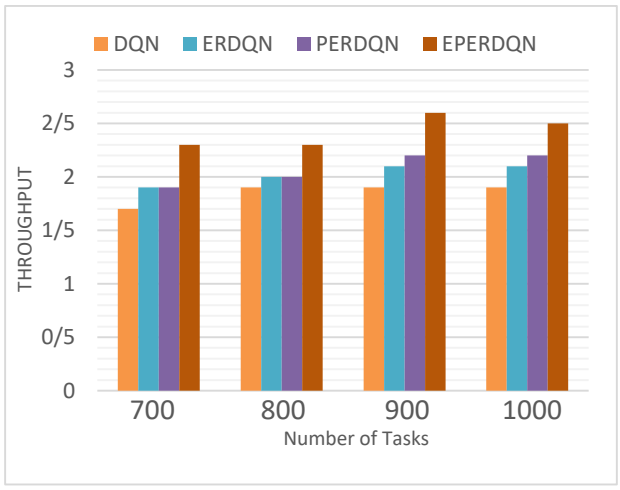
number of tasks	DQN	ERDQN	PERDQN	EPERDQN
700	409	371	359	301
800	421	397	385	342
900	463	422	407	348
1000	519	469	459	401

جدول ۸: توان عملیاتی در سناریوی ۲ با استفاده از larg_size در مجموعه داده GOCJ

number of tasks	DQN	ERDQN	PERDQN	EPERDQN
700	1.7	1.9	1.9	2.3
800	1.9	2	2	2.3
900	1.9	2.1	2.2	2.6
1000	1.9	2.1	2.2	2.5



شکل ۸: مقایسه زمان کل در سناریوی ۲ با استفاده از larg_size در مجموعه داده GOCJ



شکل ۹: مقایسه توان عملیاتی در سناریوی ۲ با استفاده از larg_size در مجموعه داده GOCJ

روند کلی ارزیابی پارامترهای مورد نظر با اسـتفاده از مجموعه داده GOCJ مشـابه سـناریوی قبلی تکرار شـده‌اسـت و عملکرد الگوریتم EPERDQNدر پارامترهای مورد نظر مطابق شکل‌های ۶-۹ بهتر است. نکته دیگری که از نتایج سناروی ۲ می‌توان ا ستنباط‌کرد این ا ست که با افزایش تعداد کارها، الگوریتم پیشنهادی مقیاس‌پذیری بهتری دارد و با اندازه بزرگ در مقایسـه با اندازه معمولی در مجموعه دادهGOCJ، بهبود بیشتری نسبت به سایر الگوریتم‌های مورد ارزیابی در این تحقیق، حاصل‌می‌شود. نتایج آزمایش انجام‌شده در مجموعه داده GOCJ نشان

می‌دهد که میانگین توان عملیاتی نسبت به DQN ۳۸ درصد، ERDQN ۲۳ درصد و PERDQN ۱۸ درصد بهبود یافته‌است. علاوه بر این، با در نظرگرفتن مهلت زمانی وظایف در تابع پاداش، ویژگی‌های بیشتری از نیازهای کاربران را برآورده‌می‌کند و تعداد نقض SLA را بهبود می‌بخشد. با توجه به نتایج سناریو های ۱ و ۲ می‌توان استنباط‌کرد که الگوریتم PERDQL پایه در پارامترهای ارزیابی‌شده بهبود قابل‌توجهی نداشته‌است، حتی در برخی از اجراها نتایج کمتری نسبت به سایر الگوریتم‌ها داشته‌است، بنابراین الگوریتم پیشنهادی توسعه‌یافت. الگوریتم EPERDQN به‌درستی بار کاری را روی سرورها توزیع‌می‌کند و در تمام پارامترهای ارزیابی‌شده عملکرد بهتری دارد.

۶. نتیجه‌گیری و کارهای آینده

در این مقاله یک الگوریتم بهبود یافته بر اساس PERDQN برای غلبه بر مشکلات الگویتم DQN شامل آموزش طولانی مدت، تخمین غیرواقعی پاداش و نمونه برداری تصادفی از حافظه ارائه شد. سپس با تعریف مناسب محیط ابری بر اساس فرآیند تصمیم‌گیری مارکوف از الگوریتم پیشنهادی برای زمانبندی کارها بر روی ماشین‌های مجازی در محاسبات ابری استفاده شده‌است. نتایج به‌دست‌آمده نشان‌می‌دهد که تمامی پارامترهای مورد نظر در الگوریتم EPERDQN بهبود می‌یابند که در مقیاس بزرگتر تفاوت بیشتر می‌شود. ولی در چندین حوزه امکان بهبود وجوددارد. از جمله اینکه می‌توان فضای حالت مسأله را توسعه داد به‌طوریکه موارد بیشتری از خصوصیات کارها را شامل‌شود. همچنین این امکان وجوددارد که علاوه بر زمان‌بندی کارها برای اهداف دیگری به‌عنوان نمونه مصرف انرژی در مرکز داده از الگوریتم EPERDQN استفاده‌نمود. با وجود مزایای زیاد الگوریتم‌های یادگیری عمیق تقویتی، اما باید سربار محاسباتی آن‌ها برای استفاده در محیط ابری در نظر گرفته‌شود، مساله کاهش زمان یادگیری در الگوریتم پیشنهادی در نظر گرفته شده است، با این حال، همچنان می‌توان آن را به عنوان یک کار آینده بهبود بخشید. در قسمت شبکه عصبی نیز می‌توان از NTM استفاده‌نمود که در قسمت ذخیره و بازیابی، از یک حافظه خارجی استفاده‌نموده که موجب افزایش کارایی الگوریتم می‌شود.

مراجع

[1] A. Keivani and J.-R. Tapamo, "Task scheduling in cloud computing: A review," in *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 2019: IEEE, pp. 1-6.

[2] N. K. Gondhi and A. Gupta, "Survey on machine learning based scheduling in cloud computing," in *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 2017, pp. 57-61.

[3] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407-415, 2019.

[4] C. Shetty, H. Sarojadevi, and S. Prabhu, "Machine learning approach to select optimal task scheduling algorithm in cloud," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 6, pp. 2565-2580, 2021.

[5] J. Zhao, M. A. Rodríguez, and R. Buyya, "A deep reinforcement learning approach to resource management in hybrid clouds harnessing renewable energy and task scheduling," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021: IEEE, pp. 240-249.

[6] X. Wang *et al.*, "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning," *Journal of Manufacturing Systems*, vol. 65, pp. 130-145, 2022.

[7] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[8] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "Drl-scheduling: An intelligent qos-aware job scheduling framework for applications in clouds," *IEEE Access*, vol. 6, pp. 55112-55125, 2018.

[9] *Z. Saadati And M. Zarei, "Solving the Multi-Objective Problem of IoT Service Placement in Fog Computing Using Reinforcement Learning Approaches," Intelligent Multimedia Processing and Communication Systems(IMPCS), pp. 29-41, 4(3) , 2023.*

[10] L. Zhang, C. Yang, Y. Yan, and Y. Hu, "Distributed Real-Time Scheduling in Cloud Manufacturing by Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8999-9007, 2022.

[11] X. Wang, L. Zhang, Y. Liu, C. Zhao, and K. Wang, "Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning," *Journal of Manufacturing Systems*, vol. 65, pp. 452-468, 2022.

[12] K. Siddesha, G. Jayaramaiah, and C. Singh, "A novel deep reinforcement learning scheme for task scheduling in cloud computing," *Cluster Computing*, vol. 25, no. 6, pp. 4171-4188, 2022.

[13] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE access*, vol. 8, pp. 209320-209344, 2020.

[14] J. Liu, F. Gao, and X. Luo, "Survey of deep reinforcement learning based on value function and policy gradient," *Chinese Journal of Computers*, vol. 42, no. 6, pp. 1406-1438, 2019.

[15] K. Sharma and S. Tripathy, "DEEP REINFORCEMENT LEARNING: A SURVEY."

[16] W. Yuan, Y. Li, H. Zhuang, C. Wang, and M. Yang, "Prioritized experience replay-based deep q learning: Multiple-reward architecture for highway driving decision making," *IEEE Robotics & Automation Magazine*, vol. 28, no. 4, pp. 21-31, 2021.

[17] W. Fedus *et al.*, "Revisiting fundamentals of experience replay," in *International Conference on Machine Learning*, 2020: PMLR, pp. 3061-3071.

[18] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Learning for dynamics and control*, 2020: PMLR, pp. 486-489.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[20] H. Zhang, C. Qu, J. Zhang, and J. Li, "Self-adaptive priority correction for prioritized experience replay," *Applied sciences*, vol. 10, no. 19, p. 6925, 2020.

[21] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Information Sciences*, vol. 512, pp. 1170-1191, 2020.

[22] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *Journal of Cloud Computing*, vol. 11, no. 1, p. 3, 2022.

[23] Y. Feng and F. Liu, "Resource Management in Cloud Computing Using Deep Reinforcement Learning: A Survey," in *Proceedings of the 10th Chinese Society of Aeronautics and Astronautics Youth Forum*, 2023: Springer, pp. 635-643.

[24] S. Swarup, E. M. Shakshuki, and A. Yasar, "Task Scheduling in Cloud Using Deep Reinforcement Learning," *Procedia Computer Science*, vol. 184, pp. 42-51, 2021.

[25] G. Rjoub, J. Bentahar, O. Abdel Wahab, and A. Saleh Bataineh, "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems," *Concurrency and Computation: Practice and Experience*, p. e5919, 2020.

[26] M. Roknaldini And E. Noroozi, "Presenting A Hybrid Method of Deep Neural Networks to Prevent Intrusion in Computer Networks," Intelligent Multimedia Processing and Communication Systems(IMPCS), 4(4), pp. 57-65, 2023.

[27] H. Che, Z. Bai, R. Zuo, and H. Li, "A deep reinforcement learning approach to the optimization of data center task scheduling," *Complexity*, vol. 2020, 2020.

[28] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, "Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911-1923, 2021.

[29] K. Li, Z. Peng, D. Cui, and Q. Li, "SLA-DQTS: SLA Constrained Adaptive Online Task Scheduling Based on DDQN in Cloud Computing," *Applied Sciences*, vol. 11, no. 20, p. 9360, 2021.

[30] J. Lin, D. Cui, Z. Peng, Q. Li, and J. He, "A Two-Stage Framework for the Multi-User Multi-Data Center Job Scheduling and Resource Allocation," *IEEE Access*, vol. 8, pp. 197863-197874, 2020.

[31] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *2018 23rd Asia and South pacific design automation conference (ASP-DAC)*, 2018: IEEE, pp. 129-134.

[32] B. Ning, F. H. T. Lin, and S. Jaimungal, "Double deep q-learning for optimal execution," *Applied Mathematical Finance*, vol. 28, no. 4, pp. 361-380, 2021.

[33] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016, vol. 30, no. 1.

[34] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, 2016: PMLR, pp. 1995-2003.

[35] R. Liu and J. Zou, "The effects of memory replay in reinforcement learning," in *2018 56th annual allerton conference on communication, control, and computing (Allerton)*, 2018: IEEE, pp. 478-485.

[36] W. Zaremba and I. Sutskever, "Reinforcement learning neural turing machines-revised," *arXiv preprint arXiv:1505.00521*, 2015.

[37] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[38] D. Fährmann, N. Jorek, N. Damer, F. Kirchbuchner, and A. Kuijper, "Double deep q-learning with prioritized experience replay for anomaly detection in smart environments," *IEEE Access*, vol. 10, pp. 60836-60848, 2022.

[39] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling," *Procedia Engineering*, vol. 38, pp. 3566-3572, 2012.

[40] K. Ramasubramanian and A. Singh, "Deep learning using keras and tensorflow," in *Machine Learning Using R*: Springer, 2019, pp. 667-688.

[41] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, p. 38, 2018.

[42] P. Amini and A. Kalbasi, "An Adaptive Task Scheduling Approach for Cloud Computing Using Deep Reinforcement Learning," in *2024 Third International Conference on Distributed Computing and High Performance Computing (DCHPC)*, 2024: IEEE, pp. 1-9.

[43] S. Mangalampalli, G. R. Karri, M. Kumar, O. I. Khalaf, C. A. T. Romero, and G. A. Sahib, "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing," *Multimedia Tools and Applications*, vol. 83, no. 3, pp. 8359-8387, 2024.

[44] S. Mangalampalli *et al.*, "Multi Objective Prioritized workflow scheduling using Deep reinforcement based Learning in Cloud Computing," *IEEE Access*, 2024.

[45] S. Mangalampalli, G. R. Karri, and U. Kose, "Multi Objective Trust aware task scheduling algorithm in cloud computing using Whale Optimization," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 2, pp. 791-809, 2023.

[46] A. R. Kadhim and F. Rabee, "Deadline and Cost Aware Dynamic Task Scheduling in Cloud Computing Based on Stackelberg Game," *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 3, 2023.

پی‌نوشت

[1] reinforcement learning
[2] Virtual Machine (VM)
[3] Deep Q Learning
[4] Temporal difference
[5] Experience Replay DQN
[6] Prioritized Experience Replay DQL
[7] Markov decision process (MDP)
[8] State
[9] Action
[10] Probability
[11] Reward
[12] Discount factor
[13] deadline
[14] MakeSpan
[15] Neural Turing Machine
[16] Infrastructure as a Service
[17] Throughput
[18] google cluster traces
[19] regular_size
[20] large_size