مقاله پژوهشی

# یادگیری تقویتی برخط مبتنی بر سیاست برای توسعه خودکار منطق خودتطبیقی سیستم‌های خود تطبیق

Doi: 10.30508/KDIP.2023.375966.1055

کاظم نیک فرجام

۱-عضو هیات علمی، دانشگاه آزاد اسلامی، واحد بیرجند، بیرجند، ایران، دانشجوی دکتری، دانشگاه آزاد قزوین، قزوین، ایران

تاریخ دریافت: ۱۴۰۱/۰۹/۱۷
تاریخ پذیرش: ۱۴۰۲/۰۱/۲۹
صفحه: ۰۰ - ۰۰

## چکیده:

سیستم‌های خودتطبیق، قادرند با وجود تغییرات در محیط‌های پویا نیازمندی‌های کیفی‌شان را حفظ کنند. برای توسعه این سیستم‌ها، مهندسان باید بتوانند منطق خودتطبیقی که زمان و نحوه تطبیق سیستم را به شکل مناسبی بیان نماید، ایجاد کنند. توسعه منطق خودتطبیق سیستم، بدلیل عدم قطعیت‌ها در زمان طراحی، دشوار است. زیرا نمی‌توان تمام تغییرات محیطی بالقوه را پیش‌بینی کرد. یادگیری تقویتی برخط با یاد گرفتن اثربخشی عملیات تطبیق از طریق تعامل سیستم با محیط در زمان اجرا، مشکل عدم‌قطعیت زمان طراحی را برطرف می‌کند، و می‌تواند توسعه منطق خودتطبیقی را به طور خودکار درآورد. روش‌های یادگیری تقویتی بر خط موجود برای توسعه سیستم‌های خودتطبیق، دو نقص دارند که درجه خودکارسازی را محدود می‌کند. اولا؛ نیازمند تنظیم دقیق نرخ اکتشاف به صورت دستی هستند. ثانیا: ممکن است برای تقویت مقیاس‌پذیری نیاز به کمی‌سازی حالت‌های محیطی به صورت دستی داشته باشد. این مقاله رویکردی را برای خودکارسازی فعالیت‌های دستی فوق‌الذکر با استفاده از یادگیری تقویتی مبتنی بر سیاست، به عنوان نوع متفاوتی از یادگیری تقویتی معرفی کرده است. همچنین امکان‌سنجی و کاربرد رویکرد را در توسعه منطق یک سیستم اطلاعاتی خودتطبیق نشان می دهد.

**کلمات کلیدی:** منطق خودتطبیق، یادگیری تقویتی بر خط مبتنی بر سیاست، عدم‌قطعیت، سیستم خودتطبیق.

## ۱- مقدمه

خودتطبیقی، باعث تسهیل در توسعه سیستم‌های شده و سیستم را قادر می‌سازد، نیازمندی‌های کیفی خود را در محیط‌های پویا حفظ کنند و در زمان اجرا به شکل انعطاف‌پذیری عمل کنند (کاپل، معمار، مطهری‌نژاد[1]، ۲۰۱۱؛ صالحی، تحویلداری[2]، ۲۰۰۹). جهت تطبیق، سیستم اطلاعاتی خودتطبیق می‌تواند ساختار، پارامترها و رفتار خود را در زمان اجرا بر اساس درک خود از محیط، و از خود و نیازمندی‌های، تغییر دهد. به عنوان مثال یک فروشگاه وب برخط، خودتطبیق را در نظر بگیرید که مهندسان حفظ کارایی سیستم را به عنوان هدف یادگیری بیان کرده‌اند. بنابراین باید کارایی سیستم تحت بارهای کاری[3] در حال تغییر در زمان اجرا حفظ کند. این سیستم در مواجهه با افزایش ناگهانی بارکاری، به صورت برخط، با غیرفعال کردن ویژگی‌های اختیاری سیستم خود را با شرایط جدید بوجود آمده، تطبیق می‌دهد. یادگیری تقویتی برخط، عملیات توسعه منطق خودتطبیقی (که قبلا دستی توسط مهندس انجام می‌شد) را به طور خودکار انجام می‌دهد. به عنوان مثال؛ موتور توصیه سیستم را هنگام بارکاری زیاد غیرفعال کند تا منابع کمتری استفاده شود، و بتواند به همه درخواست‌ها با حداقل تاخیر پاسخ دهد (کلین، ماجیو، آرزن و همندز[4]، ۲۰۱۴).

برای توسعه سیستم خودتطبیق، مهندسان باید منطق خودتطبیقی را طوری توسعه دهند که زمان و نحوه تطبیق سیستم را به شکل مناسبی نمایش دهد. برای مثال، مهندسان، ممکن است قوانینی به شکل (رویداد-شرایط-عمل) تعریف کنند که تعیین می‌کند در پاسخ به یک تغییر محیطی معین کدام عمل تطبیقی اجرا شود.

توسعه منطق خودتطبیق مستلزم درک درستی از سیستم اطلاعاتی و محیط آن دارد و اینکه عملیات تطبیق برکیفیت سیستم چه تأثیری می‌گذارند (چن، بهسون[5]، ۲۰۱۶؛ دلپویت، باربمان، کرامر، مگی، سیکس، و یوچیتل[6]، ۲۰۱۴). نگرانی مهم دیگر، باید تغییرات بالقوه محیطی که سیستم ممکن است در زمان اجرا با آن مواجه شود را پیش‌بینی کرد، باید تعریف شود که چگونه سیستم می‌تواند خود را در پاسخ به این تغییرات محیطی تطبیق دهد. با این حال، به دلیل عدم قطعیت، پیش‌بینی تمام تغییرات محیطی بالقوه در زمان طراحی در بیشتر موارد غیر ممکن است (دلپویت و همکاران، ۲۰۱۴؛ رامیرز، جنسن و چنگ[7]، ۲۰۱۲). علاوه بر این، در حالی که اثر عملیات تطبیق بر روی سیستم ممکن است شناخته شده باشد، اما پیش بینی دقیق اثر عملیات تطبیق به دلیل ساده کردن فرضیات ساخته شده در زمان طراحی دشوار است (جمشیدی، کامار، اسکامل، کاستنر، گارلان[8]، ۲۰۱۹؛ دلپویت و همکاران، ۲۰۱۴).

یکی از روش‌های نوظهور برای رسیدگی به عدم‌قطعیت زمان طراحی، استفاده از یادگیری تقویتی برخط است (باریت، هالی، دوگان[9]، ۲۰۱۳؛ داتریل، و همکاران[10]، ۲۰۱۱؛ لوریدو باتران، میگل السون، و لوزانو[11]، ۲۰۱۴؛ عرب‌نژاد، پال، جمشیدی، استرادا[12]، ۲۰۱۷؛ عمویی، صالحی، میراب، تحویلداری[13]، ۲۰۰۸؛ مصطفی، و ژانگ[14]، ۲۰۱۴). یادگیری تقویتی می‌تواند اثربخشی عملیات تطبیق را از طریق تعامل با محیط بیاموزد، سیستم بطور خودکار منطق خود تطبیقی را به کمک یادگیری تقویتی در زمان اجرا یاد می‌گیرد و نیازی به تنظیم دستی توسط مهندس نیست. به جای اینکه مهندسان مجبور باشند منطق خودتطبیقی، و مسئله یادگیری را به شکلی اعلانی، برحسب اهداف یادگیری که سیستم باید به آن دست

1- Kappel, Maamar, & Motahari-Nezhad
2- Salehie, & Tahvildari
3- Work load
4- Klein, Maggio, Årzén, & Hernández
5- Chen, & Bahsoon
6- D'Ippolito, Braberman, Kramer, Magee, Sykes, & Uchitel
7- Ramirez, Jensen, & Cheng
8- Jamshidi, Cámara, Schmerl, Käestner, & Garlan
9- Barrett, Howley, & Duggan
10- Dutreilh etal
11- Lorido-Botran, Miguel-Alonso, & Lozano
12- Arabnejad, Pahl, Jamshidi, & Estrada
13- Amoui, Salehie, Mirarab, & Tahvildari,
14- Moustafa, & Zhang

یابد، بیان کنند، خود سیستم به طور خودکار با کمک یادگیری تقویتی برخط این عملیات را می‌آموزد. روش‌های یادگیری تقویتی برخط موجود، برای توسعه سیستم‌های اطلاعاتی خود تطبیقی دو کاستی را نشان می‌دهند که درجه اتوماسیونی را که ممکن است به دست آید، محدود می‌کند. اولا؛ برای تسهیل در همگرایی الگوریتم یادگیری، مهندسان سیستم، باید به صورت دستی نرخ اکتشاف[1] را در مقابل نرخ بهره‌برداری[2] تنظیم کنند، به عنوان مثال، اینکه چند بار عملیات تطبیقی انتخاب شود که قبلا انتخاب نشده (اکتشاف)، یا اینکه چند بار از عملیات تطبیقی قبلی استفاده و بهره‌برداری شود (بهره‌برداری)، باید به طور دستی تنظیم شود. اغلب رویکردهای موجود از جدول جستجو برای نشان دادن دانش، آموخته شده، استفاده می‌کنند، که مهندسین سیستم را ملزم می‌کند، به صورت دستی حالت‌های محیط را (برای توسعه‌پذیری در صورتی که محیط دارای تعداد حالت‌های زیادی است)، کمی‌سازی و گسسته‌سازی کنند. این دو فعالیت دستی ممکن است گران و به طور بالقوه غیرقابل اعتماد باشند (جمشیدی و همکاران، ۲۰۱۹). همچنین ممکن است به اطلاعاتی نیاز داشته باشند که در زمان طراحی به دلیل عدم قطعیت زمان طراحی در دسترس نباشد. ایده اصلی این است که فعالیت‌های دستی فوق به طور خودکار با استفاده از یادگیری تقویتی مبتنی بر سیاست[3] (یا خط‌مشی) به عنوان یک نوع متفاوت از یادگیری تقویتی انجام شوند (ساتون، مک. الستر، سینگه و منصور[4]، ۱۹۹۹؛ نچام، نوروزی، و اسچارمن[5]، ۲۰۱۷). به طور ساده در یادگیری تقویتی مبتنی بر سیاست، دانش آموخته شده در قالب یک شبکه عصبی مصنوعی نشان داده می‌شود(ساتون و همکاران، ۱۹۹۹).

رویکرد پیشنهادی از لحاظ مفهومی، رسمی و فنی، یادگیری تقویتی مبتنی برسیاست را در مدل مرجع سیستم‌های خودتطبیق شناخته شده ادغام و یکپارچه می‌کند. این کار، استفاده از یادگیری تقویتی بر خط را برای توسعه سیستم‌های خودتطبیق ساده می‌کند. زیرا نیاز به؛ کمی کردن حالت‌های محیطی به صورت دستی ندارد و نیاز به تنظیم دقیق نرخ اکتشاف به طور دستی نیست. در این مقاله امکان‌سنجی و کاربرد یادگیری تقویتی برخط مبتنی بر سیاست را، روی یک سیستم اطلاعاتی خودتطبیق (برنامه وب خودتطبیق) نشان می‌دهد.

## ۲- مبانی نظری

یادگیری تقویتی به صورت موثر کمک می‌کند مهندسی منطق خودتطبیقی سیستم به طور خودکار انجام شود. به طور کلی، یادگیری تقویتی از طریق تعاملات عامل با محیط خود اثربخشی اقدامات عامل را می‌آموزد (ساتون و بارتو[6]، ۲۰۱۸). در این نوع یادگیری عامل در مرحله زمانی t، عملی را در حالت محیطی St انجام می‌دهد. در نتیجه، حالت سیستم در مرحله زمانی t+۱ به St+۱ تبدیل می‌شود و عامل برای اجرای عمل، پاداش rt+۱ را دریافت می‌کند. هدف یادگیری تقویتی، بهینه‌سازی پاداش‌های تجمعی است. هنگام بکارگیری یادگیری تقویتی برای سیستم‌های اطلاعاتی خودتطبیق، «عمل» به معنای عمل تطبیقی مشخص (مانند؛ اصلاح ساختار، پارامترها یا رفتار سیستم)، و «عامل» نقش منطق خودتطبیقی را بر عهده می‌گیرد. محیط، سیستم اطلاعاتی که در زمان اجرا باید تطبیق داده شود، را شامل می‌شود. رویکردهای موجود که از یادگیری تقویتی برای ساخت سیست های خودتطبیق استفاده می‌کنند، اکثرا از یادگیری تقویتی مبتنی بر ارزش یا مقدار[7] استفاده می‌کنند. یادگیری تقویتی مبتنی بر مقدار، یک تکنیک یادگیری تقویتی بدون مدل است که از یک تابع مقدار برای نمایش دانش آموخته شده، استفاده می‌کند. تابع مقدار، پاداش تجمعی مورد انتظار هنگام انجام یک عمل خاص در یک وضعیت مشخص را نمایش می‌دهد. عملی که بالاترین ارزش را در یک حالت معین دارد انتخاب

1- Exploration Rate
2-Rate Exploitation
3- Policy based Reinforcment Learning
4- Sutton, McAllester, Singh, & Mansour
5- Nachum, Norouzi, Xu, & Schuurmans
6- Sutton, & Barto
7- Value-based RL

می‌شود. دو نوع مشهور یادگیری تقویتی مبتنی بر مقدار عبارتند از Q-Learning و SARSA که این دو الگوریتم در نحوه به روزرسانی تابع ارزش متفاوت هستند (ساتون و بارتو، ۲۰۱۸). رویکردهای یادگیری تقویتی موجود برای سیستم‌های خودتطبیق از دو روش مختلف برای نمایش تابع ارزش (مقدار) استفاده می‌کنند که در ادامه توضیح داده شده است.

### روش اول: تابع ارزش در قالب جدول جستجو[1]

اغلب رویکردهای موجود، مقادیر تابع ارزش را در یک جدول جستجو ذخیره می کنند. اگر چه راه حل جدولی پیاده‌سازی ساده‌ای داشته و به خوبی قابل درک است (کلین و همکاران، ۲۰۱۴)، اما دو محدودیت کلیدی دارد. اول، به دلیل ماهیت گسسته جدول جستجو، تکنیک‌های حل جدولی به فضاهای حالت و عمل گسسته محدود می‌شوند و نمی‌توانند با فضاهای حالت و عمل پیوسته کنار بیایند. این بدان معنی است که حالات محیطی باید قابل شمارش باشند و نمی‌توانند توسط متغیرهای پیوسته (با ارزش واقعی) نمایش داده شوند (مصطفی و ژانگ[2]، ۲۰۱۷). دوم اینکه، اندازه جدول جستجو مستقیماً به تعداد حالت‌های محیطی که باید ذخیره شوند، بستگی دارد. اندازه جدول با افزایش تعداد متغیرهای حالت به طور تصاعدی، افزایش می‌یابد که جهت نگهداری، نیازمند حافظه بالایی است. در نتیجه، تکنیک‌های راه حل جدولی از مقیاس‌پذیری ضعیف رنج می‌برند، زیرا فرآیند یادگیری برای اینکه بتواند به طور مؤثر یاد بگیرند، باید برای همه ورودی‌های جدول، تمام داده‌ها را جمع‌آوری کند (کلین و همکاران، ۲۰۱۴؛ مصطفی و ژانگ، ۲۰۱۷). یک روش متداول برای حل این محدودیت‌ها، کمی‌سازی حالت‌های محیطی پیوسته با تعریف تعداد کمی از حالت‌های محیطی گسسته است. این کمی‌سازی یک فعالیت دستی است که باید توسط مهندسان سیستم انجام شود. بنابراین ممکن است گران و به طور بالقوه غیرقابل اعتماد باشد. همان طور که ذکر شد، یادگیری تقویتی در حین اجرا و به صورت تدریجی میزان اثربخشی عملیات عامل را از طریق تعامل عامل با محیط یاد می‌گیرد و از این طریق منطق خودتطبیقی سیستم اطلاعاتی به طور خودکار در حین اجرا ساخته می‌شود (ساتون و بارتو، ۲۰۱۸). حالت‌های محیط ممکن است به صورت درشت دانه[3] کمی‌سازی شوند، یا ممکن است خیلی ریزدانه[4] باشند، که ممکن است به دلیل اندازه بسیار بزرگ جدول جستجو؛ منجر به مقیاس‌پذیری ضعیف شود. همچنین، ممکن است وسعت فضای حالات (یعنی مجموعه همه حالت‌ها)، به دلیل عدم قطعیت در زمان طراحی، ناشناخته باشد. بنابراین تعیین کران‌های پایین و بالایی حالت‌های گسسته، امکان‌پذیر نیست. دانستن وسعت فضای حالت به این معنی است که توسعه‌دهندگان در زمان طراحی می‌توانند تمام تغییرات محیطی بالقوه‌ای که سیستم ممکن است در زمان اجرا با آن مواجه شود، را پیش‌بینی کنند.

### روش دوم: تقریب‌زدن تابع ارزش[5]

یک رویکرد جایگزین برای جلوگیری از کمی‌سازی فضای حالت، تقریب زدن تابع مقدار است. به عنوان مثال، با استفاده از تکنیک‌های خطی و غیرخطی (مانند شبکه‌های عصبی مصنوعی). این روش امکان مقابله با فضاهای حالت بزرگ را با تعمیم وضعیت‌های نادیده فراهم می‌کند. با وجود چنین تقریبی، یادگیری تقویتی مبتنی بر مقدار به طور کلی با معضل بهره برداری-اکتشاف[6] مواجه است (ساتون و بارتو، ۲۰۱۸). برای بهینه سازی پاداش‌ها، عملیاتی انتخاب می‌شوند که اثربخشی خود را قبلا نشان داده و درون پایگاه دانش ذخیره شده‌اند) معروف به بهره‌برداری (در عین حال، برای کشف چنین عملیاتی در وهله اول، باید عملیاتی را انتخاب کرد که قبلاً انتخاب نشده‌اند

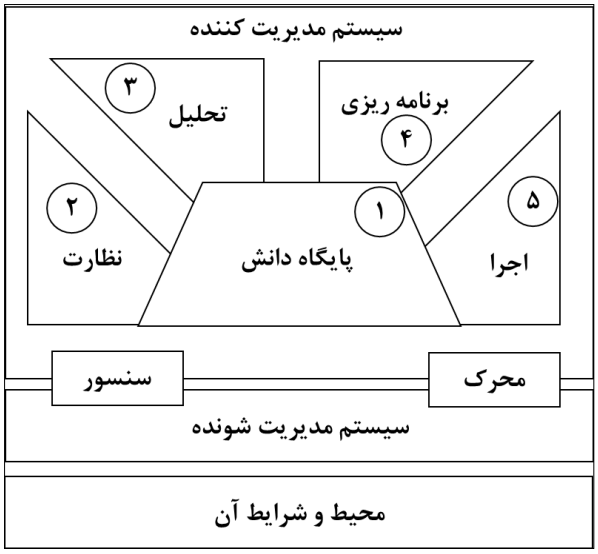
1- Value Function as Lookup Table.
2- Moustafa, A., & Zhang
3- Coarse-grained
4- Fine-grained
5- Value function Approximation.
6- Exploration – Exploitation Dilemma.

(معروف به اکتشاف). یک راه حل معمول برای معضل اکتشاف - بهره‌برداری مکانیسم حریصانه[1] است. در طول یادگیری، مکانیسم حریصانه به طور تصادفی یک عمل را با احتمال اپسیلنی (خیلی کم) انتخاب و کاوش می‌کند. چالش مهندس سیستم، تنظیم دقیق تعادل بین نرخ بهره‌برداری و نرخ اکتشاف به منظور اطمینان از همگرایی فرآیند یادگیری است. به عنوان مثال، مهندس ممکن است مکانیزمی را اجرا کند که در طول زمان اجرا مقدار اپسیلن را کاهش می‌یابد و در نتیجه میزان اکتشاف کاهش می‌یابد تا همگرایی را تسهیل کند. با این حال، در یادگیری برخط، چالش این است که چه زمانی و چگونه می‌توان دوباره این مقدار را در محیط‌های غیرثابت[2]، (یعنی محیط‌هایی که پویا هستند و در آن نتایج عملیات تطبیقی در طول زمان تغییر می‌کند) افزایش داد. به طور خلاصه درجه خودکارسازی رویکردهای موجود محدود است. مهندس سیستم مجبور است این تنظیمات را دستی انجام دهد که به کار و تلاش زیادی نیاز داشته و به دلیل عدم قطعیت در زمان طراحی، انجام آنها دشوار است.

### رویکرد یادگیری تقویتی برخط مبتنی بر سیاست

رویکرد پیشنهادی، مطابق شکل زیر یادگیری تقویتی را در مدل MAPE-K[3] که یک مدل مرجع شناخته شده برای سیستم های خودتطبیق[4] است ادغام می‌کند (ایگلسیا، وینز[5]، ۲۰۱۵؛ دیلمس، و همکاران[6]، ۲۰۱۳؛ کپارت و چیس[7]، ۲۰۰۳)

همانطور که درشکل شماره (۱) نشان داده شده است، این مدل، ساختار مفهومی یک سیستم خودتطبیق را به دو عنصر اصلی تقسیم می‌کند: منطق سیستم (یا سیستم




شکل (۱): مدل MAPE-K برای سیستم خود تطبیق

مدیریت شونده) و منطق خودتطبیق (یا سیستم مدیریت کننده). منطق خود تطبیقی به چهار فعالیت مفهومی اصلی تقسیم شده نظارت (پایش یا مانیتور)- تحلیل- برنامه‌ریزی و اجرا) که از یک پایگاه دانش مشترک استفاده می‌کند. پایگاه دانش شامل؛ اطلاعات مربوط به سیستم مدیریت شده و محیط آن است (به عنوان مثال، مدل‌های زمان اجرای کدگذاری شده). این چهار فعالیت شامل؛ نظارت بر منطق سیستم و محیط از طریق حس‌گرها[8]، تجزیه و تحلیل داده‌های نظارت شده در مرحله قبل (نظارت) برای تعیین نیاز به تطبیق سیستم، برنامه‌ریزی عملیات تطبیق، و در نهایت اجرای این عملیات تطبیقی از طریق محرک‌ها[9]، که در نهایت باعث اصلاح منطق سیستم در زمان اجرا، می‌شود.

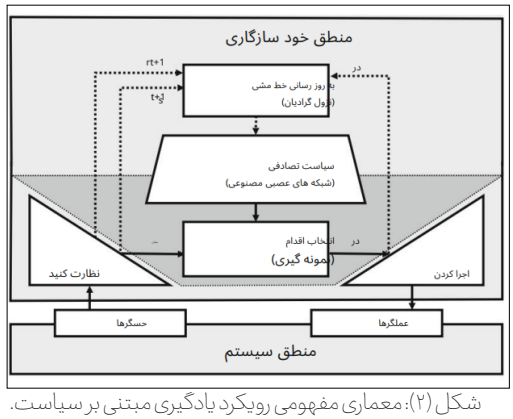
1- E-Greedy
2- Non-stationary
3- Monitor-Analys-Planning-Execute-Knowleage.
4- SelfAdaptive Systems
5- Iglesia, & Weyns
6- De Lemos etal
7- Kephart, & Chess
8- Sensors
9- Effectors

## مبانی یادگیری تقویتی مبتنی بر سیاست

ایده اساسی پشت یادگیری تقویتی مبتنی بر سیاست، استفاده مستقیم و بهینه‌سازی سیاست انتخاب عملیات تصادفی پارامتری شده است (ساتون، مک الستر، سینگه و منصور[1]، ۱۹۹۹؛ نچام، نوروزی، ایکسو، و اسچارمن[2]، ۲۰۱۷). سیاست انتخاب عمل، وضعیت‌ها را به یک توزیع احتمال در فضای عملیات (یعنی مجموعه عملیات تطبیقی ممکن) نگاشت می‌کند. بدین ترتیب عملیات با نمونه‌گیری از این توزیع احتمال انتخاب می‌شوند. از یک چرخه یادگیری با تعداد دفعات معین از قبل تعیین شده، برای به روزرسانی سیاست استفاده می‌شود. در پایان هر چرخه یادگیری، وضعیت و پاداش‌های دریافتی، بروز می‌شوند، طوری که توزیع احتمال حاصل به سمتی تغییر کند، که احتمال انتخاب عملیاتی را افزایش می‌دهد که منجر به پاداش تجمعی بالاتری می‌شوند.

## مروری مفهومی بر معماری روش پیشنهادی

شکل شماره (۲) معماری مفهومی رویکرد پیشنهادی را نشان داده و مشخص می‌کند چگونه عناصر یادگیری تقویتی مبتنی بر سیاست در حلقه MAPE-k ادغام می‌شوند. ناحیه خاکستری تیره رنگ شکل شماره (۲) نشان می‌دهد که انتخاب عمل یادگیری تقویتی، جای فعالیت تحلیل و برنامه‌ریزی از فعالیت‌های MAPE-k را می‌گیرد. همچنین سیاست تصادفی آموخته شده نقش پایگاه دانش سیستم خودتطبیق را برعهده دارد.




شکل (۲): معماری مفهومی رویکرد یادگیری مبتنی بر سیاست.

در زمان اجرا، منطق خودتطبیق (سیستم مدیریت کننده) از این سیاست استفاده می‌کند برای انتخاب (از طریق نمونه‌گیری) یک عمل تطبیق بر اساس وضعیت فعلی تعیین شده توسط فعالیت نظارت. انتخاب عملیات، تعیین می کند آیا نیازی به اجرای (با توجه به وضعیت فعلی) و برنامه‌ریزی (یعنی انتخاب) عملیات تطبیقی مربوطه است یا خیر؟ تابع به روزرسانی سیاست، از دنباله عملیات، وضعیت‌ها و پاداش‌ها برای به روزرسانی سیاست استفاده می‌کند. در این رویکرد، به روزرسانی سیاست از طریق روش‌های گرادیان نزولی (ساتون و بارتو، ۲۰۱۸) انجام شده، و سیاست، در قالب یک شبکه عصبی مصنوعی نشان داده شده است.

در معماری پیشنهادی، پاداش‌ها توسط فرآیند نظارت محاسبه می‌شود، زیرا این فرآیند به تمام داده‌های جمع‌آوری شده از حس‌گرهای سیستم و محیط دسترسی دارد.

## فرمول‌بندی رویکرد

در این مقاله مساله یادگیری در قالب فرایند تصمیم مارکوف[3] MDP=(S,A,T,R) فرمول‌بندی شده که با یک چهارتایی تعریف می‌شود: مجموعه S: فضای حالات سیستم و محیط که توسط مولفه نظارت و از طریق حس‌گرها قابل مشاهده است (مثل میزان بارکاری محیط یا کارایی سیستم).

مجموعه A: فضای عملیات مجموعه عملیات تطبیقی ممکن را نشان می‌دهد. عملیات تطبیقی ممکن که می‌توان با استفاده از محرک‌های سیستم تطبیقی انجام شود. به عنوان مثال، فعال یا غیرفعال کردن ویژگی خاصی از سیستم.

مجموعه T: تابع احتمال انتقال به حالت بعدی به شرطی که در زمان t در حالت خاصی باشیم و عمل خاصی را انجام دهیم به صورت زیر تعریف می‌شود:

رابطه(۱) $(T: S*A*S \grave{a}[0,1] = Pr(St+1|St, at)$

تابع پاداش[4] R مشخص کننده پاداش‌های عددی

1- Sutton, McAllester, Singh, & Mansour,
2- Nachum, Norouzi, Xu, & Schuurmans
3- Markov decision process.
4- Reward function

که سیستم در هر حالت که قرار گرفت از محیط دریافت می‌کند.

رابطه (۲) $R: S \to IR+$

تابع پاداش همچنین بیان‌گر هدف یادگیری است که باید بدست آید، که در مورد سیستم وب خودتطبیق حفظ نیازمندی‌های کیفی سیستم است. (به عنوان مثال، کارایی سیستم نباید زیر یک آستانه معین قرار گیرد).

یادگیری تقویتی مبتنی برسیاست، به کمک فرآیند تصمیم مارکوف راه حلی در قالب یک سیاست تصادفی پارامتری شده پیدا می‌کند:

رابطه (۳) $\Pi\theta : S * A \to [0,1] \qquad \Pi\theta(a|s)=Pr(a|s)$

احتمال انجام عمل تطبیقی a درحالت S را مشخص می‌کند. پارامترهای سیاست یادگیری (وزن‌های شبکه عصبی مصنوعی) به صورت بردار زیر نشان داده می‌شود.

رابطه (۴) $\theta \in IR+$

با توجه به عدم قطعیت‌ها در زمان طراحی سیستم، فرض می‌کنیم که S، A و R را در زمان طراحی می‌دانیم، اما T را نمی‌دانیم. به طور دقیق‌تر، حتی اگر حالت‌های دقیق سیستم را ندانیم و نتوانیم فضای حالات را تشکیل دهیم، متغیرهای حالت را می‌دانیم. به عنوان مثال، حتی اگر بارکاری دقیق یک برنامه وب را ندانیم (و شاید حتی میزان حداکثر بارکاری را هم ندانیم)، اما می‌توانیم یک متغیر برای نمایش حالت بارکاری مثل w به شکل عددی مثبت صحیح تعریف کنیم. معمولا فرض می‌کنیم که T تابع انتقال را به دلیل عدم قطعیت زمان طراحی )چگونگی تأثیر تطبیق بر کیفیت سیستم( نمی‌دانیم. به عنوان مثال، ممکن است درک دقیقی از نحوه عملکرد پیکربندی‌های مختلف سیستم تحت بارهای کاری مختلف نداشته باشیم.

برای انتخاب یک الگوریتم یادگیری تقویتی مبتنی برسیاست، دو نکته اصلی را در نظر می‌گیریم. ابتدا، همان طور که فرض می‌کنیم تابع انتقال T را نمی‌دانیم، باید از یک نوع یادگیری بدون مدل برای یادگیری تقویتی مبتنی بر سیاست استفاده کنیم. دوم، برای تسهیل یادگیری برخط، به الگوریتمی نیاز داریم که به طور مداوم سیاست را بدون انتظار برای نتیجه نهایی، (یعنی بدون انتظار برای رسیدن به حالت پایانی،) به روز کند. الگوریتم‌های منتقد- بازیگر[1] یک نوع الگوریتم بدون مدل از الگویتم‌های یادگیری تقویتی مبتنی بر سیاست هستند که در آن دانش به طور مداوم بدون انتظار برای نتیجه نهایی به روزرسانی می‌شود. در این مقاله از بهینه‌سازی سیاست تقریبی[2] PPO به عنوان الگوریتم منتقد - بازیگر استفاده می‌کنیم (اسچالمن، ولسکی، داریوال، رادفورد، و کلیموف[3]، ۲۰۱۷).

الگوریتم PPO با استفاده از یک تابع برش[4] از به روزرسانی سیاست‌های خیلی بزرگ جلوگیری می‌کند. به روزرسانی سیاست‌های خیلی بزرگ ممکن است به این معنی باشد که یادگیری تقویتی جواب بهینه سراسری را از دست داده و در یک بهینه محلی گیر می‌کند. برای نمایش مدل‌های بازیگر- منتقد، این الگوریتم از یک شبکه پرسپترون چند لایه با دو لایه پنهان ۶۴ نورونی استفاده می‌کند (تعداد نرون‌ها در لایه‌های ورودی و خروجی شبکه به تعداد متغیرهای عملیات و حالت سیستم بستگی دارد).

## پیاده‌سازی و ارزیابی تجربی

برای نشان دادن امکان‌سنجی و کاربرد یادگیری پیشنهادی، دامنه آزمایش‌ها این است که تحلیل کنیم، آیا سیستم قادر به یادگیری و بهبود منطق خودتطبیقی در زمان اجرا است یا خیر؟ در این مرحله یک تحلیل مقایسه‌ای با رویکردهای یادگیری تقویتی مبتنی بر مقدار انجام ندادیم، چنین مقایسه‌ای فراتر از محدوده مقاله فعلی خواهد بود. چنین مقایسه‌ای نیازمند تغییر و تحلیل دقیق طیف وسیعی از پارامترها برای رویکرد مبتنی بر مقدار است، از جمله تنظیم نرخ اکتشاف، و همچنین سطوح و اشکال مختلف کمی‌سازی فضای حالت. به ویژه، باید مراقب بود که مقایسه ناعادلانه‌ای انجام نشود. مقایسه ممکن است به شدت تحت تأثیر کمی‌سازی یا به اصطلاح کوانتیزاسیون انتخابی قرار گیرد. کوانتیزاسیون بسیار

1- Actor-Critic.
2- Proxcimal Policy Optimization
3- Schulman, Wolski, Dhariwal,Radford, & Klimov
4- Clipping

ریزدانه، ممکن است به این معنی باشد که یادگیری مبتنی بر ارزش (مقدار) همگرایی بسیار کندی خواهد داشت. همچنین یک کوانتیزاسیون خیلی درشت ممکن است به این معنی باشد که یادگیری مبتنی بر ارزش قادر به تمایز و تفکیک بین حالت‌های مختلف سیستم نخواهد بود و در نتیجه نمی‌تواند پاداش‌های تجمعی را بهینه کند.

### برنامه کاربردی وب خود تطبیق

برای آزمایش از یک سیستم وب خودتطبیق حراجی برخط RUBiS-Brownout استفاده می‌کنیم (کلین و همکاران، ۲۰۱۴). هنگامی که کاربر، مورد خاصی را از بین کالاهای حراجی درخواست می‌کند، موتور توصیه برنامه، لیستی از موارد توصیه شده را بر اساس حراج‌های گذشته به کاربر ارائه می‌دهد. با توجه به نیاز موتور توصیه به منابع مختلف جهت راهنمایی و توصیه به کاربر، RUBiS Brownout باید بین دو نیازمندی کیفی موازنه برقرار کند: به حداکثر رساندن تجربه کاربر با ارائه توصیه‌های بیشتر، و در عین حال به حداقل رساندن تأخیر ملاحظه شده توسط کاربران. بنابراین، نیاز به استفاده از موتور توصیه را می‌توان با تنظیم یک متغیر بنام متغیر تنظیم (دیمر[1]) در محدوده [۰،۱] که نشان دهنده احتمال فعال شدن موتور توصیه به ازای هر درخواست است، تطبیق داد. بنابراین، مقدار کم این متغیر روی هر دو نیازمندی کیفی تأثیر می‌گذارد: نرخ بالای توصیه‌ها، تجربه کاربر را افزایش می‌دهد، اما در عین حال برای اجرای این توصیه و راهنمایی، نیاز به منابع افزایش یافته است و در نتیجه ممکن است تأخیر را افزایش دهد و زمان انتظار کاربر برای پاسخ گرفتن به درخواست‌هایشان زیاد شود و باعث نارضایتی کاربر گردد.

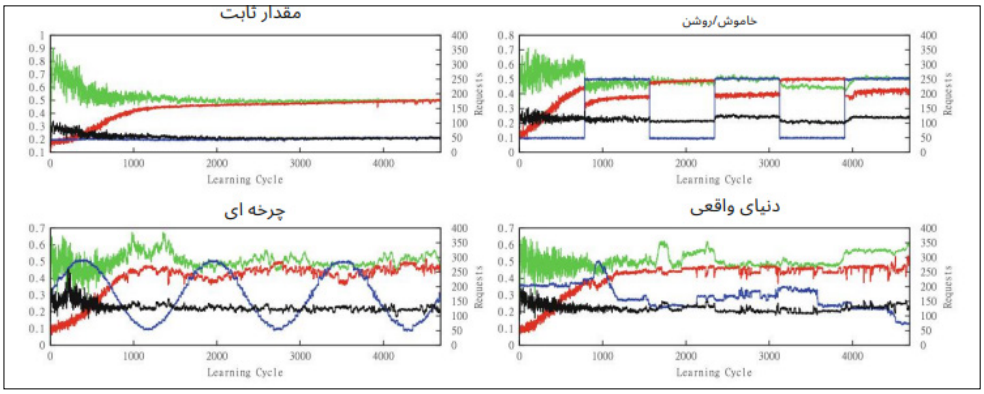
همان طور که در جدول شماره (۱) نشان داده شده، مساله یادگیری را در قالب فرآیند تصیم مارکوف MDP تعریف می‌کنیم. تابع پاداش[2] را به گونه‌ای تعریف می‌کنیم که الگوریتم یادگیری تعادل و موازنه خوبی بین تاخیر کم و نرخ‌های توصیه بالا پیدا کند. تابع پاداش طوری تعریف شده که پاداش بیشتر، بهتر است و هدف سیستم به حداکثر رساندن پاداش تجمعی است. فرض می‌کنیم که رضایت کاربر برای تأخیرهای بالاتر از λmax (که اینجا ۱۰ میلی ثانیه در نظر گرفته شده) کاهش می‌یابد و بنابراین تأخیرهای بالاتر از λmax جریمه می‌شوند.

جدول (۱): یادگیری تقویتی برخط در برنامه وب خود تطبیق

| | | |
|---|---|---|
| State St=(Ut,αt,λt) | Ut ϵ IN+:<br>αt ϵ[0,1]<br>λt ϵIR+ | تعداد درخواست‌های کاربر مورد پایش قرار گرفته در لحظه t<br>نرخ توصیه پایش شده در وضعیت جاری<br>تاخیر جاری پایش شده |
| Action<br>αt ϵ A | A= δ ϵ [0,1] | متغیر تطبیقی (دیمر) |
| Reward<br><br>r(t)=<br>αt * f(λt) | αt ϵ[0,1]<br>λt ϵ IR+<br>f(λt)= 1 if λt < λmax<br>f(λt)= 0 if λt >2* λmax<br>else f(λt)= - λt / λmax 2+ | ms 10= λmax |

1- Dimmer value
2- Reward Function.

جهت پیاده‌سازی RUBiS-Brownout روی ماشین مجازی با رم ۱۶ گیگابایتی که اوبونتو ۱۶,۰۴,۵ را اجرا می‌کند، نصب و اجرا کردیم. همچنین از [۱۷] httpmon به عنوان مولد بار کاری برای تولید انواع مختلف بارهای کاری استفاده کردیم. بارهای کاری را با استفاده از الگوهای بارکاری متفاوت طبق مقاله (بارتون و همکاران، ۲۰۱۴) استفاده کردیم: الگوی بارکاری ثابت[1] (تعداد ثابت درخواست‌ها)، الگوی خاموش/روشن[2] (منعکس کننده پردازش دسته‌ای و دوره‌ای)، و الگوی چرخه‌ای[3] (بارکاری درطی دوره افزایش وکاهش می‌یابد). همچنین بخشی از بارکاری ردیابی شده در دنیای واقعی را هم در نظر می‌گیریم (منن[4]، ۲۰۱۷). در این آزمایش از ۴۶۰۰ چرخه یادگیری برای هر کدام از الگوهای بارکاری‌ها، استفاده کرده‌ایم. هر چرخه یادگیری از داده‌های پایش شده ۱۲۸ مرحله زمانی متوالی قبل استفاده می‌کند. مقدار ۱۰ میلی ثانیه را برای $\lambda max$ به عنوان آستانه تأخیر، تنظیم می‌کنیم. نمودارهای شکل شماره (۳) وضعیت، عملیات و پاداش برای هر چرخه یادگیری را به طور میانگین برای ۱۲۸ مشاهده مراحل زمانی متوالی نشان می‌دهند. شکل شماره (۳) نتایج را برای انواع الگوهای بارکاری نشان می‌دهد. الگوی چرخه‌ای - الگوی (خاموش/روشن)- الگوی دنیای واقعی و الگوی ثابت.




شکل (۳): رفتار یادگیری تقویتی در سیستم وب خودتطبیق (بارکاری=سیاه تاخیر=سیاه متغیر تطبیق دیمر=سبز پاداش=قرمز).

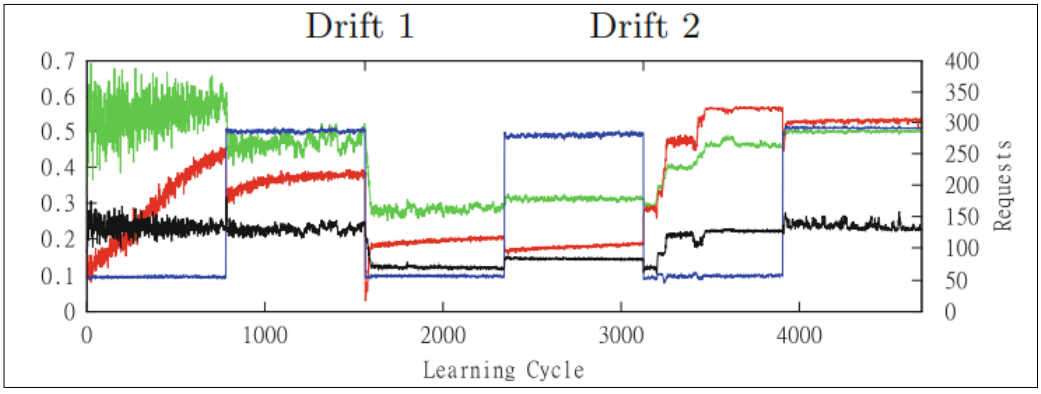
نتایج نشان می‌دهند که چگونه این نوع یادگیری سیستم را قادر می‌سازد تا به مرور زمان خود را تطبیق دهد. سیستم به طور خودکار متغیر تنظیم تطبیق (دیمر) را، بسته به الگوی بارکاری، تنظیم می‌کند، و در نتیجه تعادل بین تاخیر و تجربه کاربر را بهینه می‌کند (که در افزایش پاداش‌های تجمعی قابل مشاهده است). در ابتدای فرآیند یادگیری، مقدار کم متغیر تنظیم دیمر، واریانس بالایی را برای همه الگوهای بارکاری نشان می‌دهد، اما پس از مدتی واریانس عملیات تطبیق به وضوح کمتر می‌شود.

برای الگوی بارکاری ثابت، پاداش پس از حدود ۱۹۵۰ چرخه یادگیری به مقدار ۴۷ صدم همگرا شده و مقدار متغیر دیمر در حدود نیم، تثبیت می‌شود که منجر به بالاترین نرخ توصیه بدون نقض آستانه تاخیر می‌شود. برای الگوی بارکاری خاموش-روشن، پاداش در طول زمان برای تنظیمات خاموش و همچنین روشن، افزایش می‌یابد. از تکرار دوم به بعد می‌توان همگرایی را مشاهده کرد. هنگام مقایسه یادگیری، برای دوره‌های خاموش و روشن به طور جداگانه، می‌توان مشاهده کرد که یادگیری تقویتی می‌تواند از دانش کسب شده در مورد بارهای کاری مشابه درطول زمان استفاده مجدد کند. الگوی بارکاری

1- Constant workload pattern.
2- On/Off workload pattern.
3- Cyclic workload pattern.
4- Mann

چرخه‌ای، مشابه بارکاری خاموش / روشن، نشان می‌دهد که چگونه یادگیری تقویتی می‌تواند به کمک پایگاه دانش از قبل به دست آمده تعمیم و گسترش یابد. مشاهدات سراسری از سیستم، بسیار با الگوی بارکاری خاموش / روشن قابل مقایسه است، با این تفاوت که میانگین پاداش آهسته‌تر افزایش و کاهش می‌یابد، زیرا بارکاری به آرامی تغییر می‌کند. برای بارکاری دنیای واقعی، الگوریتم می‌تواند از حالت‌های تجربه شده قبلی بیاموزد و با تنظیم مقدار کم برای متغیر دیمر، پاداش را تقریبا در همان سطح نگه دارد، حتی اگر بارکاری در طول زمان تغییر کند. مزیت مهم یادگیری این است که اگر بارکاری مشابه ای، دوباره تکرار شود، این رویکرد قادر است به سرعت عملیات تطبیقی موثر را پیدا کند.

شکل شماره (۴) نشان می‌دهد چگونه این نوع یادگیری به طور خودکار در محیط‌های غیر ثابت[1] عمل می‌کند. پس از چرخه یادگیری ۱۵۶۲ (رانش[2] اول)، منابع محاسباتی ماشین را به نصف کاهش دادیم. این بدان معنی است که برای همان مقدار کم دیمر، سیستم تأخیر بیشتری را تجربه می‌کند، زیرا منابع محاسباتی کمتری در دسترس است. الگوریتم یاد می‌گیرد که مقدار کم دیمر را به گونه‌ای کاهش دهد که آستانه تأخیر نقض نشود. پس از چرخه یادگیری ۳۱۲۵ (رانش دوم) منابع را یک و نیم برابر افزایش دادیم. مجددا مقادیر کم متغیر تنظیم تطبیق دیمر بر این اساس تنظیم می‌شوند. این نوع یادگیری قادر است این ویژگی غیرایستایی محیط را بدون انجام فرآیند نظارت صریح، تغییرات در منابع محاسباتی و بدون تغییر صریح نرخ اکتشاف، یاد بگیرد.




شکل (۴): رفتار الگوریتم یادگیری تقویتی مبتنی بر سیاست برخط درمحیط غیرثابت (آبی =بارکاری، سیاه =تأخیر؛ سبز =متغیر تطبیق دیمر؛ قرمز=پاداش)

## تهدیدات

برای مشاهده اینکه آیا یادگیری تقویتی مبتنی برسیاست نتایج مورد انتظار را برآورده می‌کند یا خیر، از یک شبکه پرسپترون چند لایه به عنوان یک شبکه عصبی ساده برای نشان دادن سیاست استفاده کردیم. برای حل مشکل کمی‌سازی خوب فضای حالات و تنظیم مناسب نرخ اکتشاف، از تنظیمات پیش فرض فراپارامتر استفاده کردیم. همچنین، به دلیل ماهیت تصادفی هر یک از آزمایش‌ها را چندین بار تکرار کردیم، تا اثرات تصادفی بودن را کاهش دهیم.

یادگیری تقویتی مبتنی بر سیاست پیشنهادی، می‌تواند فضای عمل بزرگی را در حین یادگیری برخط کنترل کند، مشروط بر اینکه فضای عملیات پیوسته باشد. با این حال، بر روی مجموعه‌ای از عملیات غیرپیوسته، یعنی گسسته، کار نمی‌کند و بنابراین نمی‌تواند به اعمالی که قبلاً دیده نشده بودند، گسترش و تعمیم یابد. به طور معمول، سیستم‌های اطلاعاتی خودتطبیقی دارای فضاهای عملیاتی گسسته بزرگی هستند، مانند سیستم‌های خودتطبیقی مبتنی بر ویژگی یا خودتطبیقی مبتنی بر معماری. به عنوان مثالی از سیستم خودتطبیقی مبتنی بر ویژگی با فضای

1- Non_Stationary
2- Drift 1

گسسته بزرگ، سیستمی را در نظر بگیرید که ۱۰ ویژگی اختیاری دارد که این ویژگی‌ها، می‌توانند به صورت پویا حین اجرا فعال و غیرفعال شوند. بنابراین فضای تطبیق سیستم یک فضای گسسته شامل ۱۰۲۴ عملیات تطبیقی است(یعنی دو به توان ده). این ۱۰۲۴عملیات تطبیقی را نمی‌توان به عنوان یک متغیر پیوسته نشان داد. در این حالت، یک راه حل می‌تواند جایگذاری عملیات گسسته در یک فضای پیوسته و استفاده از جستجوی نزدیک‌ترین همسایه برای یافتن نزدیک‌ترین عملیات گسسته باشد (دولاک ارنولد، و همکاران[1]، ۲۰۱۵). عملکرد یادگیری ماشین تا حدود زیادی به مقدار داده‌های موجود برای یادگیری بستگی دارد. هنگامی که از یادگیری تقویتی برای سیستم‌های خودتطبیقی استفاده می‌شود، ممکن است به چرخه‌های یادگیری زیادی نیاز باشد، تا فرآیند یادگیری همگرا شود (مصطفی و همکاران، ۲۰۱۴). در آزمایش‌های موجود، یادگیری حدود در ۲۰۰۰ چرخه یادگیری (با داده‌های ۲۵۶۰۰۰ مرحله زمان) برای همگرایی نیاز داشت. تا زمانی که یادگیری تقویتی همگرا نشود، سیستم به احتمال زیاد تطبیق‌های ناکارآمدی را اجرا می‌کند، زیرا هنوز مشاهده کافی وجود ندارد. تطبیق‌های ناکارآممکن است منجر به اثرات منفی شود، زیرا دریک سیستم زنده اجرا می‌شوند (فلوهو، و پورتر[2]، ۲۰۱۷) [۱۱]. برای سرعت بخشیدن به همگرایی، یافتن برآوردهای اولیه خوب برای دانش آموخته شده (ساتون و بارتو[3]، ۲۰۱۸؛ داتریل و همکاران، ۲۰۱۱). یا انجام یادگیری برون خط از طریق شبیه‌سازی سیستم می‌تواند استفاده شود (تسارو، جانگ، داس و بنانی[4]، ۲۰۰۷). در عین حال، یادگیری تقویتی برخط ممکن است برای سیستم‌هایی که در محیطی کار می‌کنند که اثر آزمایش و خطا یادگیری تقویتی، قابل تحمل نباشد (به عنوان مثال، اگر عملیات تطبیقی به محیط آسیب برساند)، قابل استفاده نباشد.

## ۳- نتیجه‌گیری

بیشتر مقالات موجود که از یادگیری تقویتی در سیستم‌های اطلاعاتی خودتطبیق استفاده می‌کنند، از یادگیری تقویتی مبتنی بر مقدار استفاده کردند. در این قسمت به چگونگی تعریف تابع مقدار یا همان تابع ارزش در کارهای گذشته پرداخته شد. عمویی و همکاران (۲۰۰۸) از SARSAبرای یادگیری عملیات تطبیقی مؤثر در یک برنامه وب خودنطبیق از تابع مقدار در قالب جدول جستجو استفاده کردند. آنها با تعریف تنها دو مقدار برای هر یک از متغیرهای حالت، حالت‌های محیط را به طور بنیادی کمی می‌کنند. بدین منظور، از آستانه‌های تعریف شده توسط کارشناسان حوزه استفاده می‌شود. هوانگ و همکاران (۲۰۱۱) از Q-Learningبرای بهینه‌سازی پویای تخصیص منابع در فرآیندهای تجاری استفاده کردند. علاوه بر بهینه‌سازی تخصیص منابع برای یک نمونه فرآیند، آنها با در نظر گرفتن هزینه‌های منابع سراسری هنگام به روزرسانی جدول ارزش، بهینه‌سازی را در بین نمونه‌های فرآیند توزیع می‌کنند. داتریل و همکاران (۲۰۱۱) ازQ- Learning برای مدیریت منابع ابری خودمختار استفاده کردند. آنها فرض می‌کنند که کران بالایی برای متغیرهای حالت وجود دارد. بو و همکاران (۲۰۱۲) از Q-Learning برای پیکربندی خودکار ماشین‌های مجازی ابری و برنامه‌های کاربردی استفاده کردند. برای تسهیل مقیاس‌پذیری، سه حالت برای گسسته‌سازی تعریف می‌کنند که نشان دهنده بازه‌های بالا، متوسط و پایین متغیر حالت مربوطه است.

عرب نژاد و همکاران (۲۰۱۷) از Q-Learning فازی و SARSA برای ایجاد ابری با خاصیت خود مقیاسی استفاده کردند. حالات محیطی کوانتیزه می‌شوند و در نتیجه به مجموعه‌های کوچکی از حالت‌های بیان شده در منطق فازی محدود می‌شوند. مزیت منطق فازی این است که بسیاری از حالت‌ها را می‌توان تنها با چند حالت فازی نشان داد. با این حال، رویکرد آنها هنوز نیازمند شناسایی عناصر فازی «گسسته» در مجموعه فازی است که یادگیری تقویتی بر اساس آن عمل می‌کند. آلینو و همکاران (۲۰۱۶) پیشنهاد استفاده از یادگیری تقویتی مبتنی بر ارزش را برای مونتاژ خدمات چند عاملی می‌دهند. عامل‌ها اطلاعات

1- Dulac-Arnold etal
2- Filho, & Porter
3- Sutton, & Barto
4- Tesauro, Jong, Das, & Bennani

نظارت بر وضعیت را به اشتراک می‌گذارند و از نمایش جدولی تابع مقدار استفاده می‌کند. سیلواندر(۲۰۱۹) پیشنهاد استفاده از Q- Learning را با تقریب تابعی از طریق یک شبکه عصبی عمیق[1] برای بهینه‌سازی فرآیندهای تجاری دارد. همه این رویکردها از الگوریتم حریصانه برای مکانیزم اکتشاف استفاده می‌کنند که نیاز به تنظیم دقیق نرخ اکتشاف به صورت دستی دارد. در مقابل، یادگیری ما نیازی به کنترل صریح نرخ اکتشاف ندارد، کاوش به طور خودکار از طریق انتخاب عملیات احتمالاتی انجام می‌شود.

مقاله حاضر به معرفی و ارزیابی یادگیری تقویتی برخط مبتنی بر سیاست برای تسهیل در مهندسی سیستم‌های خودتطبیقی پرداخت. رویکرد پیشنهادی با افزایش درجه اتوماسیون به مهندس سیستم کمک می‌کند. به طور مشخص، این روش نه به حالت‌های محیطی کمی‌سازی شده به صورت دستی نیاز دارد و نه به صورت دستی، نیاز به تعیین پارامترهای اکتشاف مناسب برای الگوریتم یادگیری تقویتی دارد. پیشنهاد می‌شود محققین در آینده به رویکردی برای مدیریت فضاهای عملیاتی گسسته بزرگ بپردازند، تا انواع بیشتری از سیستم‌های خودتطبیق را به تصویر بکشند. همچنین بررسی کنند، مدل‌های یادگیری عمیق، بازنمایی بهتری برای دانش آموخته شده، فراهم می‌کنند.

1- Deep Network Learning

## منابع

4-Aiello, M., Johnsen, E. B., Dustdar, S., & Georgievski, I. (2016). *Service-Oriented and Cloud Computing*. Springer International Publishing.
5-Amoui, M., Salehie, M., Mirarab, S., & Tahvildari, L. (2008, March). Adaptive action selection in autonomic software using reinforcement learning. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)* (pp. 175-181). IEEE.
6-Arabnejad, H., Pahl, C., Jamshidi, P., & Estrada, G. (2017, May). A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In *2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)* (pp. 64-73). IEEE.
7-Barrett, E., Howley, E., & Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and computation: practice and experience*, *25*(12), 1656-1674.
8-Bu, X., Rao, J., & Xu, C. Z. (2012). Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *IEEE transactions on parallel and distributed systems*, *24*(4), 681-690.
9-Chen, T., & Bahsoon, R. (2016). Self-adaptive and online qos modeling for cloud-based software services. *IEEE Transactions on Software Engineering*, *43*(5), 453-475.
10-De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., ... & Wuttke, J. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers* (pp. 1-32). Springer Berlin Heidelberg.
11-D'Ippolito, N., Braberman, V., Kramer, J., Magee, J., Sykes, D., & Uchitel, S. (2014, May). Hope for the best, prepare for the worst: multi-tier control for adaptive systems. In *Proceedings of the 36th International Conference on Software Engineering* (pp. 688-699).
12-[9[.Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., ... & Coppin, B. (2015). Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.
13-Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., & Truck, I. (2011, May). Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. In *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems* (pp. 67-74).
14-Filho, R. R., & Porter, B. (2017). Defining emergent software using continuous self-assembly, perception, and learning. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *12*(3), 1-25.
15-Huang, Z., van der Aalst, W. M., Lu, X., & Duan, H. (2011). Reinforcement learning based resource allocation in business process management. *Data & Knowledge Engineering*, *70*(1), 127-145.
16-Iglesia, D. G. D. L., & Weyns, D. (2015). MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *10*(3), 1-31.
17-Jamshidi, P., Cámara, J., Schmerl, B., Käestner, C., & Garlan, D. (2019, May). Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 39-50). IEEE.
18-Kappel, G., Maamar, Z., & Motahari-Nezhad, H. R. (Eds.). (2011). *Service Oriented Computing: 9th International Conference, ICSOC 2011, Paphos, Cyprus, December 5-8, 2011, Proceedings* (Vol. 7084). Springer.
19-Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, *36*(1), 41-50.
20-Klein, C., Maggio, M., Årzén, K. E., & Hernández-Rodriguez, F. (2014, May). Brownout: Building more robust cloud application

21-Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of grid computing*, *12*, 559-592.
22-Mann, Z. Á. (2017). Resource optimization across the cloud stack. *IEEE Transactions on Parallel and Distributed Systems*, *29*(1), 169-182.
23-Moustafa, A., & Zhang, M. (2014, June). Learning efficient compositions for QoS-aware service provisioning. In *2014 IEEE International Conference on Web Services* (pp. 185-192). IEEE.
24-Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, *30*.
25-Ramirez, A. J., Jensen, A. C., & Cheng, B. H. (2012, June). A taxonomy of uncertainty for dynamically adaptive systems. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 99-108). IEEE.
26-Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, *4*(2), 1-42.
27-Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXivpreprintarXiv:1707.06347*.
28-Silvander, J. (2019). Business process optimization with reinforcement learning. In *Business Modeling and Software Design: 9th International Symposium, BMSD 2019, Lisbon, Portugal, July 1–3, 2019, Proceedings 9* (pp. 203-212). Springer International Publishing.
29-Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
30-Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, *12*.
31-Tesauro, G., Jong, N. K., Das, R., & Bennani, M. N. (2007). On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, *10*, 287-299.