مقاله پژوهشی

توسعه منطق خودتطبیقی سیستم‌های خودتطبیق به کمک یادگیری تقویتی عمیق

Doi: 10.30508/kdip.2023.383007.1060

کاظم نیک فرجام[1]

۱-مربی و عضو هیات علمی دانشگاه آزاد اسلامی واحد بیرجند، بیرجند، ایران

تاریخ دریافت: ۱۴۰۱/۱۱/۰۴
تاریخ پذیرش: ۱۴۰۲/۰۵/۱۷
صفحه: ۰۰ - ۰۰

چکیده

یک سیستم خودتطبیق می‌تواند ساختار و رفتار خود را در زمان اجرا، بر اساس درکش از محیط و از خودش و نیازمندی‌هایش، اصلاح کند. یکی از عناصر کلیدی در توسعه این سیستم‌ها، منطق خودتطبیقی آن است که زمان و نحوه تطبیق سیستم را رمزگذاری می‌کند. هنگام توسعه منطق تطبیق، مهندسان با چالش عدم قطعیت زمان طراحی مواجه هستند. برای تعریف زمان تطبیق سیستم، باید تمام حالات محیطی بالقوه را پیش‌بینی کنند. پیش‌بینی تمام تغییرات محیطی بالقوه اغلب به دلیل اطلاعات ناقص در زمان طراحی، غیرممکن است. یادگیری تقویتی برخط، با یادگیری اثربخشی عملیات تطبیق، از طریق تعامل سیستم با محیط در زمان اجرا، مشکل عدم قطعیت زمان طراحی را برطرف، و توسعه منطق خودتطبیقی را به طور خودکار درمی‌آورد. عناصر یادگیری تقویتی، در حلقه MAPE-K سیستم‌های خودتطبیق ادغام می‌شود. روش‌های یادگیری تقویتی برخط موجود در سیستم‌های خودتطبیق، دانش آموخته شده را در قالب تابع ارزش نمایش می‌دهند و دو نقص دارند که درجه خودکارسازی و توسعه را محدود می‌کند: نیازمند تنظیم دقیق نرخ اکتشاف به صورت دستی هستند. از سوی دیگر برای تقویت توسعه‌پذیری، ممکن است نیاز به کمی‌سازی حالت‌های محیط به صورت دستی باشد. در این مقاله برای خودکارسازی فعالیت‌های فوق از یادگیری تقویتی عمیق، استفاده شد. در این یادگیری، دانش در قالب یک شبکه عصبی، در وزن‌های شبکه عصبی پنهان است. نتایج آزمایش‌ها، از سرعت همگرایی بالای یادگیری حکایت دارد.

واژگان کلیدی: یادگیری تقویتی عمیق، عدم قطعیت، منطق خود تطبیق، سیستم خودتطبیق.

۱- مقدمه

خودتطبیقی، باعث تسهیل در توسعه سیستم شده و سیستم را قادر می‌سازد، نیازمندی‌های کیفی خود را در محیط‌های پویا حفظ کند و در زمان اجرا به شکلی انعطاف‌پذیری عمل نماید (اشوف و زیسمن، ۲۰۱۱؛ صالحی و تحویلداری، ۲۰۰۸). سیستم خودتطبیق می‌تواند ساختار، پارامترها و رفتار خود را در زمان اجرا بر اساس درک خود از محیط، و از خود و نیازمندی‌هایش، تغییر دهد.

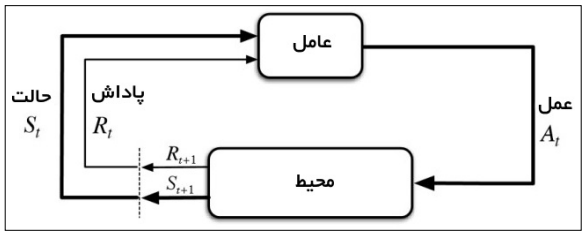
به عنوان مثال یک فروشگاه وب برخط خودتطبیق را در نظر بگیرید که مهندسان حفظ کارایی سیستم را به عنوان هدف یادگیری بیان کرده‌اند. بنابرین باید کارایی سیستم تحت بارهای کاری[1] در حال تغییر در زمان اجرا، حفظ شود. سیستم در مواجهه با افزایش ناگهانی بارکاری، به صورت برخط، با غیرفعال کردن ویژگی‌های اختیاری، خود را با شرایط جدید بوجود آمده، تطبیق می‌دهد. یادگیری تقویتی برخط، عملیات توسعه منطق خودتطبیقی (که قبلا دستی توسط مهندس انجام می‌شود)، را به طور خودکار انجام می‌دهد. به عنوان مثال؛ موتور توصیه سیستم را که یک ویژگی اختیاری است، در هنگام بارکاری زیاد (درخواست‌های زیاد کاربران)، غیرفعال می‌کند، تا منابع کمتری استفاده شده و بتواند به همه درخواست‌های کاربران با حداقل تاخیر پاسخ دهد (کلین و ماگو، ۲۰۱۴). عدم قطعیت زمان طراحی یک چالش مهم در توسعه یک سیستم خودتطبیق است. تعریف اینکه چگونه سیستم باید هنگام مواجهه با یک وضعیت محیطی جدید سازگار شود، نیازمند درک تأثیر دقیق عملیات تطبیقی است و احتمال دارد، در زمان طراحی شناخته شده، نباشد. یکی از روش‌های نوظهور برای رسیدگی به عدم قطعیت زمان طراحی در سیستم‌های خودتطبیق، استفاده از یادگیری تقویتی برخط است (آمویی، صالحی و تحویلداری، ۲۰۰۸؛ مصطفی و ژانگ، ۲۰۱۳؛ عرب نژاد، ۲۰۱۷). با استفاده از این نوع یادگیری، سیستم خود تطبیق می‌تواند از داده‌های عملیاتی واقعی و نتایج بازخوردی که فقط در زمان اجرا در دسترس است، بیاموزد.

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 نموده، همچنین تعیین کند در پاسخ به یک تغییر محیطی، معین کدام عمل تطبیقی اجرا شود. توسعه منطق خودتطبیق مستلزم درک درستی از شرایط سیستم و محیط آن دارد و اینکه عملیات تطبیق بر کیفیت سیستم چه تأثیری می‌گذارند (پیلیتو و همکاران، ۲۰۱۴؛ چن و باشمن، ۲۰۱۷). نگرانی مهم دیگر، پیش‌بینی تغییرات بالقوه محیطی بوده که سیستم ممکن است در زمان اجرا با آن مواجه شود. باید تعریف شود که چگونه سیستم می‌تواند خود را در پاسخ به این تغییرات محیطی تطبیق دهد. با این حال، به دلیل عدم قطعیت‌های موجود در محیط و سیستم در حین اجرا، پیش‌بینی تمام تغییرات محیطی بالقوه در زمان طراحی در بیشتر موارد غیرممکن است (رامیرز و جانسون، ۲۰۱۲). علاوه بر این، در حالی که اثر عملیات تطبیق بر روی سیستم ممکن است شناخته شده باشد، اما پیش‌بینی دقیق اثر عملیات تطبیق به دلیل ساده کردن فرضیات ساخته شده در زمان طراحی دشوار است (جمشیدی و کامارا، ۲۰۱۹).

یادگیری ماشین علم طراحی ماشین‌هایی است که با استفاده از داده‌هایی که به آن‌ها داده می‌شود (نمونه‌ها) و

1- Work load

تجربیات خودشان عمل کنند، بدون آنکه همه اقدامات با بهره‌گیری از برنامه‌نویسی به آن‌ها دیکته شود. الگوریتم‌های یادگیری ماشین سه دسته: یادگیری با نظارت[1]، یادگیری نظارت نشده[2] و یادگیری تقویتی[3] است.



شکل (۱): عناصر یک مساله یادگیری تقویتی (ساتن، ۲۰۰۰)

در شکل شماره (۱)، عناصر مساله یادگیری ارایه شده است، که شامل؛ محیط[4]: جهان فیزیکی عامل؛ حالت[5]: موقعیت کنونی عامل؛ پاداش[6]: بازخورد از محیط؛ سیاست[7]: روشی برای نگاشت حالت عامل به عمل و ارزش[8]: پاداش آینده که عامل با اقدام به یک عمل در یک حالت خاص به آن دست می‌یابد.

تاکنون یادگیری ماشین در سیستم‌های خودتطبیق در موارد زیر استفاده شده است (غیبی امید، دنی، ونیز، و کوین، ۲۰۲۱).

- به روزرسانی و تغییر قوانین و سیاست‌های تطبیق[9]: به روزرسانی یا تغییر قوانین تطبیق برای پشتیبانی از سیستم در هنگام مواجهه با شرایط عملیاتی متغیر. به طور مثال، سیستم توسط یک یادگیرنده تقویتی به طور مداوم مورد بررسی قرار می‌گیرد، تا سیاست‌های تطبیقی را برای مقابله با حجم کاری در حال تغییر در سیستم به روز کند.
- پیش‌بینی و تجزیه و تحلیل استفاده از منابع[10]: بررسی وضعیت منابعی که توسط سیستم استفاده می‌شوند و بر تصمیم‌گیری سیستم خودتطبیق تأثیر می‌گذارند. به عنوان مثال، یادگیرنده‌ای که در حین اجرا مصرف انرژی باتری‌ها، مصرف CPU .. را پیش‌بینی می‌کند.
- به روز نگه داشتن مدل‌های زمان اجرا[11]: پشتیبانی از سیستم با به روز نگه داشتن مدل‌های زمان اجرا. به عنوان مثال به روز نگه داشتن مدل کارایی و مدل قابلیت اطمینان سیستم خودتطبیق.
- کاهش فضاهای تطبیق بزرگ[12]: پشتیبانی از سیستم با کاهش تعداد زیادی از گزینه های تطبیق (فضای تطبیق بزرگ) به طوری که سیستم بتواند سریعتر تصمیمات تطبیقی کارآمدتری اتخاذ کند. مثلا استفاده از یک یادگیرنده برای پیش‌بینی ویژگی‌های کیفی گزینه‌های تطبیق برای انتخاب گزینه‌های مناسب و تسریع در تجزیه و تحلیل سیستم.
- تشخیص و پیش‌بینی ناهنجاری‌ها[13]: شناسایی یا پیش‌بینی ناهنجاری‌ها در رفتار سیستم یا محیط آن که مربوط به تطبیق است. مثلا، یک یادگیرنده جریان غیرعادی ترافیک را در یک سیستم مدیریت ترافیک خودتطبیق تشخیص می‌دهد و تهدیدات سایبری را در یک شبکه ارتباطی شناسایی می‌کند.
- جمع‌آوری دانش قبلی که در دسترس نیست[14]: جمع آوری دانش اولیه ناشناخته زمان اجرا برای پشتیبانی از تطبیق. مثلا یادگیرنده یک مدل برای محاسبه سودمندی پیکربندی‌های مختلف سیستم (بعد از انجام عملیات تطبیق) می‌سازد یا یادگیرنده‌ای که سیاست‌های مدیریتی را بدون هیچ دانش قبلی شناسایی می‌کند.
- همچنین یادگیری ماشین در سیستم های خودتطبیق اهداف زیر را دنبال می‌کند (غیبی امید و همکاران، ۲۰۲۱)

1- Supervised learning.
2- Unsupervised learning.
3- Reinforcement Learning.
4- Environment
5- State
6- Reward
7- Policy
8- Value
9- Update/Change adaptation rules/policies
10- Predict/Analyze resource usage
11- Keep runtime models up-to-date
12- Reduce large adaptation space
13- Detect/Predict anomalies
14- Collect unavailable prior knowledge

۱-بهبود ویژگی‌های کیفی سیستم[1]: بهینه‌سازی و حفظ ویژگی‌های کیفی سیستم خودتطبیق در حین اجرا. مثلا پایین نگه داشتن زمان پاسخ و بالا نگه داشتن قابلیت اطمینان سیستم خودتطبیق تحت تغییر بارکاری[2] و وقوع رویدادهای غیرمنتظره .

۲-حفظ تعادل بین ویژگیهای کیفی سیستم و منابع[3]: به طور مثال، نگهداری تأخیر در یک محدوده مشخص و در عین حال حداقل استفاده از منبع محاسباتی مورد نیاز (مانند CPU) برای دستیابی به آن تاخیر.

۳-تعادل بین ویژگی‌های کیفی سیستم و هزینه[4]: به طور مثال، پایین نگه داشتن نرخ شکست یک گردش کار مبتنی بر سرویس زیر یک آستانه مشخص در حالی که هزینه (عملیاتی، مالی و ...) استفاده از خدمات به حداقل برسد.

۴-بهبود در تخصیص منابع سیستم[5]: بهبود (بهینه‌سازی، مدیریت و ..) در تخصیص منابع سیستم. به عنوان مثال، مدیریت استفاده از CPU و حافظه سیستم تحت بارهای کاری نامشخص زمان اجرا در سیستم خودتطبیق.

۵-دفاع در برابر تهدیدات سایبری[6]: خودکارسازی دفاع سایبری در سیستم خودتطبیق با شناسایی و مدیریت تهدیدات (از قبیل؛ نفوذها، ناهنجاری‌ها و غیره).

یادگیری تقویتی برخط می‌تواند اثربخشی عملیات تطبیق را از طریق تعامل با محیط بیاموزد، سیستم به طور خودکار منطق خود تطبیقی را به کمک یادگیری تقویتی در زمان اجرا یاد می‌گیرد و نیازی به تنظیم دستی توسط مهندس ندارد. به جای اینکه مهندسان مجبور باشند منطق خودتطبیقی، و مسئله یادگیری را به شکلی اعلانی، برحسب اهداف یادگیری که سیستم باید به آن دست یابد؛ بیان کنند. سیستم به طور خودکار با کمک یادگیری تقویتی برخط این عملیات را می‌آموزد.

با وجود پتانسیل‌های یادگیری تقویتی، پیاده‌سازی آن می‌تواند دشوار باشد و به همین علت کاربردهای آن هنوز محدود مانده است. یکی از موانع پیاده‌سازی این نوع یادگیری، لزوم و تکیه این روش برای جستجوی اطلاعات (اکتشاف[7]) در محیط است. برای حل مساله یادگیری تقویتی، باید یک مساله مهم‌تر یعنی؛ توازن بین جست‌وجو (اکتشاف) و استخراج (بهره‌برداری) را مورد بررسی قرار داد. جست‌وجو (اکتشاف)، به معنای یافتن اطلاعات بیشتر درباره محیط و استخراج، به معنای بهره‌برداری از اطلاعات شناخته شده قبلی برای بیشینه‌سازی پاداش است.

هدف عامل یادگیرنده تقویتی بیشینه‌سازی پاداش انباره‌ای (تجمعی) است. بنابراین به نظر می‌رسد، بهتر باشد از اطلاعات شناخته شده قبلی بهره‌برداری کنیم، تا مجموع پاداش‌ها افزایش یابد. با این حال می‌توان در یک تله افتاد، ممکن است اطلاعات ناشناخته‌ای در محیط وجود داشته باشند، که پاداش بسیار بالایی دارند. اما تاکنون استخراج نشده‌اند. مهندسان باید قوانینی را تعیین کنند تا به برقراری توازن و مدیریت این دو مساله کمک کند. از طرفی یادگیری تقویتی تمرکز زیادی روی کارایی زنده سیستم دارد، و زمان برای اکتشاف محدود است. به این دلیل هم نیازمند پیداکردن یک تعادل مناسب بین اکتشاف چیزهای جدید و بهره‌برداری از دانش اندوخته شده داریم. به این کار موازنه اکتشاف و استخراج[8] گفته می‌شود. مشکل اصلی اکتشاف این است که تعداد حالات، ممکن است بسیار زیاد یا حتی نامتناهی باشد. از طرفی ممکن است تعداد خروجی‌ها بسیار زیاد و متنوع باشند که در این حالت نیازمند نمونه برداری بسیار گسترده‌ای برای تخمین خروجی نهایی سیستم هستیم. بنابراین در مسئله یادگیری تقویتی، نیازمند یک راهکار هوشمندانه برای اکتشاف هستیم. تصمیم‌گیری‌های تصادفی بدون استفاده از یک توزیع احتمال برآورد شده، معمولاً کارایی بسیار ضعیفی دارد. در دنیای واقعی، کار آسانی نیست که به طور پیوسته، بهترین عملیات

1- Improve quality
2- Work Load
3- Balance qualities with resources
4- Balance qualities with cost
5- Improve resource allocation
6- Protect against cyber threats
7- Explore
8- Exploration vs Exploitation trade-off

انتخاب و انجام شوند؛ چراکه محیط پیوسته در حال تغییر است. یکی دیگر از مشکلاتی که سر راه یادگیری تقویتی وجود دارد، زمان و منابع محاسباتی که لازم است تا اطمینان حاصل کنیم، یادگیری به درستی انجام شده است. هر چه محیط بزرگ‌تر باشد، به زمان و منابع بیشتری برای فرآیند آموزش الگوریتم نیاز است.

روش‌های یادگیری تقویتی برخط موجود برای توسعه سیستم‌های خودتطبیقی اغلب از جدول جستجو برای نشان دادن دانش آموخته شده، استفاده می‌کنند. لذا مهندسان سیستم را ملزم می‌کنند تا به صورت دستی حالات محیط را (برای توسعه‌پذیری در صورتی که محیط دارای تعداد حالت‌های زیادی است)، کمی‌سازی (گسسته‌سازی) کنند. این فعالیت دستی ممکن است گران و به طور بالقوه غیرقابل اعتماد باشد (جمشیدی و کامارا، ۲۰۱۹). ایده اصلی در این مقاله این است که فعالیت‌های دستی فوق‌الذکر به طور خودکار با استفاده از یادگیری تقویتی مبتنی بر سیاست[1] (یا خط‌مشی)، به عنوان یک نوع متفاوت از یادگیری تقویتی برای توسعه سیستم‌های خودتطبیق انجام شوند (نچمن و نوروزی، ۲۰۱۷؛ ساتن، ۲۰۰۰). همچنین موازنه بین اکتشاف و استخراج به طور خودکار در سیستم انجام شود. در یادگیری تقویتی مبتنی بر سیاست، به جای جدول، برای نگهداری دانش‌آموخته شده، از یک شبکه عصبی مصنوعی استفاده شده و به کمک یک یادگیرنده تقویتی مبتنی بر سیاست دانش در حین اجرا و گام به گام آموخته شده و در قالب وزن‌های شبکه عصبی نمایش می‌یابد (نوروزی و نچمن، ۲۰۱۷). رویکرد پیشنهادی از لحاظ مفهومی، رسمی و فنی، یادگیری تقویتی مبتنی بر سیاست را در مدل مرجع سیستم‌های خودتطبیق شناخته شده، ادغام و یکپارچه می‌کند. این کار، استفاده از یادگیری تقویتی برخط را برای توسعه سیستم‌های خودتطبیق ساده می‌کند، زیرا نیاز به کمی کردن حالت‌های محیطی به صورت دستی ندارد. همچنین 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

همانطور که ذکر شد، یادگیری تقویتی به صورت موثر کمک می‌کند تا مهندسی منطق خودتطبیقی سیستم، به طور خودکار انجام شود. به طورکلی، یادگیری تقویتی از طریق تعاملات عامل با محیط خود، اثربخشی اقدامات عامل را می‌آموزد (ساتن و بارتو، ۲۰۱۸).

در این نوع یادگیری عامل درمرحله زمانی t، عملی را در حالت محیطی Stانجام می‌دهد. در نتیجه، حالت سیستم در مرحله زمانی t+1 به St+1 تبدیل می‌شود و عامل برای اجرای عمل، پاداش rt+1 را دریافت می‌کند. هدف یادگیری تقویتی، بهینه‌سازی پاداش‌های تجمعی است. هنگام بکارگیری یادگیری تقویتی برای سیستم‌های خودتطبیق، «عمل» به معنای عمل تطبیقی مشخص (مانند اصلاح ساختار، پارامترها یا رفتار سیستم)، و «عامل» نقش منطق خودتطبیقی را بر عهده می‌گیرد. محیط، سیستمی که در زمان اجرا باید تطبیق داده شود را شامل می‌شود.

بطور کلی سه رویکرد به یادگیری تقویتی وجود دارد:

الف) رویکرد ارزش محور[2]: در این رویکرد، هدف بهینه‌سازی تابع ارزش است. تابع ارزش، تابعی است که پاداش بیشینه آینده را مشخص می‌کند که عامل در هر حالت دریافت می‌کند. ارزش هر حالت برابر است با ارزش کل پاداشی که عامل می‌تواند انتظار داشته باشد در آینده با آغاز از آن حالت جمع‌آوری کند. عامل از این تابع ارزش برای انتخاب آنکه کدام حالت در هر گام انتخاب شود، استفاده می‌کند. عامل حالتی با بیشترین ارزش را انتخاب می‌کند.

ب) رویکرد سیاست محور[3]: در یادگیری تقویتی سیاست‌محور، هدف، بهینه‌سازی تابع سیاست بدون استفاده از تابع ارزش است. سیاست، چیزی است که رفتار عامل را در یک زمان داده شده، تعیین می‌کند. عامل یک تابع سیاست را می‌آموزد. این امر به او کمک می‌کند تا هر حالت را به بهترین عمل ممکن نگاشت کند. دو دسته سیاست وجود دارد. سیاست قطعی که برای یک

1- Policy based Reinforcment Learning
2- Value based
3- Policy based

حالت داده شده همیشه عمل مشابهی را باز می‌گرداند. و سیاست تصادفی که برای هر یک از اعمال یک توزیع احتمالی در نظر می‌گیرد. همان طور که مشهود است، سیاست مستقیما بهترین عمل برای هر حالت را بیان می‌کند.

ج) رویکرد مدل محور[1]: در یادگیری تقویتی مدل محور، ابتدا محیط مدل می‌شود. این یعنی مدلی از رفتار محیط ساخته می‌شود. مساله مهم در این رویکرد نیاز به مدلی متفاوت برای ارائه هر محیط است.

رویکردهای موجود که از یادگیری تقویتی برای ساخت سیستم‌های خودتطبیق استفاده می‌کنند، اکثرا از یادگیری تقویتی مبتنی بر ارزش (یا مقدار) استفاده می‌کنند. یادگیری تقویتی مبتنی بر مقدار، از یک تابع مقدار برای نمایش دانش‌آموخته شده، استفاده می‌کند. همیشه عملی که بالاترین ارزش را در یک حالت معین دارد، انتخاب 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 (ساتن و بارتو، ۲۰۱۸). اما 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[2]

اغلب رویکردهای موجود، مقادیر تابع ارزش را در یک جدول جستجو، ذخیره می‌کنند. اگر چه راه حل جدولی پیاده‌سازی ساده‌ای داشته و به خوبی قابل درک است (کلین و ماگو، 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 (کلین و ماگو، ۲۰۱۴؛ مصطفی و ژانگ ۲۰۱۴).

دوم اینکه، اندازه جدول جستجو مستقیماً به تعداد حالت‌های محیطی که باید ذخیره شوند، بستگی دارد و اندازه جدول با افزایش تعداد متغیرهای حالت به طور تصاعدی افزایش می‌یابد. و جهت نگهداری، نیازمند حافظه بالایی است. در نتیجه، راه حل جدولی از مقیاس‌پذیری ضعیفی رنج می‌برد. فرآیند یادگیری برای اینکه بتواند به طور مؤثر یاد بگیرد، باید برای همه ورودی‌های جدول، تمام داده‌های لازم را جمع‌آوری کند (کلین ماگو، ۲۰۱۴؛ مصطفی و ژانگ، ۲۰۱۴).

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 مهندس سیستم انجام شود. بنابراین ممکن است گران و به طور بالقوه غیرقابل اعتماد (خطای انسانی) باشد. همان طور که در مطالب قبلی اشاره شد، یادگیری تقویتی در حین اجرا و به صورت تدریجی میزان اثربخشی اقدامات عامل را از طریق تعامل عامل با محیط یاد می‌گیرد و بدین ترتیب منطق خودتطبیقی سیستم به طور خودکار حین اجرا ساخته می‌شود (ساتن و بارتو، ۲۰۱۸).

حالت‌های محیط می‌توانند به صورت درشت دانه[3]، یا خیلی ریزدانه[4] کمی‌سازی شوند، و ممکن است به دلیل اندازه بسیار بزرگ جدول جستجو منجر به مقیاس پذیری ضعیف شود. دانستن وسعت فضای حالت به این معنی است که توسعه دهندگان در زمان طراحی می‌توانند تمام تغییرات محیطی بالقوه‌ای که سیستم ممکن است در زمان اجرا با آن مواجه شود، را پیش‌بینی کنند. از طرفی نیز احتمال دارد، وسعت فضای حالات (یعنی مجموعه همه حالت‌ها)، به دلیل عدم قطعیت در زمان طراحی، ناشناخته بماند. از این رو تعیین کران‌های پایین و بالای حالت‌های گسسته امکان‌پذیر نیست.

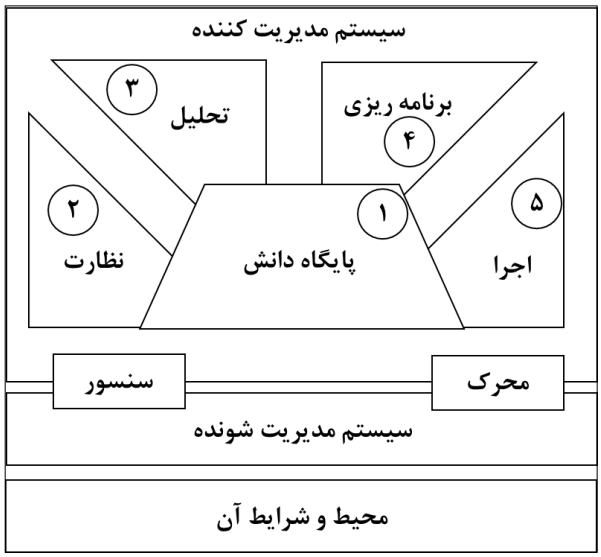
1- Model based
2- Value Function as Lookup Table
3- Coarse-grained
4- Fine-grained

روش دوم: تقریب زدن تابع ارزش[1]

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 فضای حالت بزرگ را با تعمیم وضعیت‌های نادیده فراهم می‌کند. با وجود چنین تقریبی، یادگیری تقویتی مبتنی بر مقدار همچنان با معضل بهره برداری- اکتشاف مواجه است (ساتن و بارتو، ۲۰۱۸). برای بهینه‌سازی پاداش‌ها، عملیاتی انتخاب می‌شوند که اثربخشی خود را قبلا نشان داده و درون پایگاه دانش ذخیره شده‌اند)بهره‌برداری(در عین حال، برای کشف چنین عملیاتی در وهله اول، باید عملیاتی را انتخاب کرد که قبلاً انتخاب نشده‌اند (اکتشاف). یک راه حل معمول برای معضل اکتشاف-بهره‌برداری، روش حریصانه[2] است. این روش در طول فرآیند یادگیری، 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 (عرب نژاد و جمشیدی، ۲۰۱۷). به عنوان مثال، مهندس ممکن است مکانیزمی را اجرا کند که در طول زمان اجرا مقدار اپسیلن کاهش یافته و در نتیجه میزان اکتشاف کاهش یابد 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[3]، (یعنی محیط هایی که پویا هستند و در آن نتایج عملیات تطبیقی در طول زمان تغییر می‌کند)، افزایش داد؟ (ساتن و بارتو ۲۰۱۸). بنابراین 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. برای غلبه بر این چالش‌ها از یادگیری تقویتی مبتنی بر سیاست در مدل حلقه بازخورد سیستم خودتطبیق استفاده می‌شود.

رویکرد یادگیری تقویتی برخط مبتنی بر سیاست

رویکرد پیشنهادی، مطابق شکل شماره (۲)، یادگیری تقویتی را در مدل MAPE-K[4] که یک مدل مرجع شناخته شده برای سیستم‌های خودتطبیق[5] است، ادغام می‌کند. (فیلو و پورتر، ۲۰۱۷؛ لموس و همکاران، ۲۰۱۳).



شکل (۲): مدل MAPE-K برای سیستم خود تطبیق (آمویی و همکاران، ۲۰۰۸)

همان طور که در شکل شماره (۲) نشان داده شده، این مدل، ساختار مفهومی یک سیستم خودتطبیق را به دو عنصر اصلی تقسیم می‌کند: منطق سیستم (یا سیستم مدیریت شونده[6]) و منطق خودتطبیق (یا سیستم مدیریت کننده[7]). منطق خودتطبیقی به چهار فعالیت مفهومی اصلی تقسیم شده: (نظارت یا مانیتور)، تحلیل، برنامه‌ریزی و اجرا) که از یک پایگاه دانش مشترک استفاده می‌کنند. پایگاه دانش شامل؛ اطلاعات مربوط به سیستم مدیریت شده و محیط آن است (مانند؛ مدل‌های زمان اجرای کد شده). این چهار فعالیت شامل؛ نظارت بر

1- Value function Approximation.
2- e-Greedy
3- Non-stationary
4- Monitor-Analys-Planning-Execute-Knowleage
5- SelfAdaptive Systems
6- Maneged system
7- Maneging system

منطق سیستم و محیط از طریق حس‌گرها[1]، تجزیه و تحلیل داده‌های نظارت شده در مرحله قبل برای تعیین نیاز به تطبیق سیستم، برنامه‌ریزی عملیات تطبیق، و در نهایت اجرای عملیات تطبیقی از طریق محرک‌ها[2]، که در نهایت باعث اصلاح منطق سیستم در زمان اجرا، می‌شود.

ایده اساسی پشت یادگیری تقویتی مبتنی بر سیاست، بهینه سازی سیاست انتخاب عملیات تصادفی پارامتری شده[3] است (ناچمن و نوروزی، ۲۰۱۲؛ ساتن ۲۰۰۰). سیاست انتخاب عمل، وضعیت ها را به یک توزیع احتمال در فضای عملیات (یعنی مجموعه عملیات تطبیقی ممکن) نگاشت می‌کند.

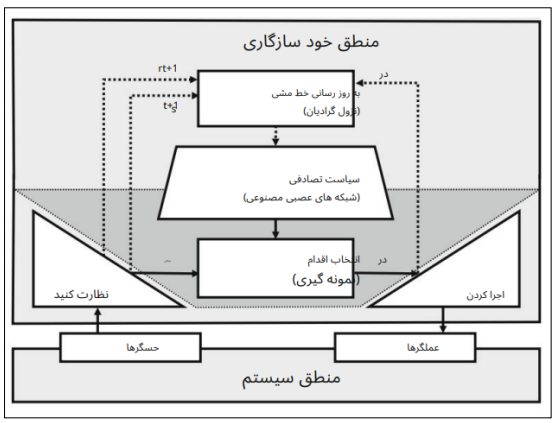
بدین ترتیب عملیات با نمونه‌گیری از این توزیع احتمال انتخاب می‌شوند. همچنین از یک چرخه یادگیری با تعداد دفعات معین از قبل تعیین شده، برای به روزرسانی سیاست استفاده می‌شود. در پایان هر چرخه یادگیری، وضعیت و پاداش‌های دریافتی، بروز می‌شوند، طوری که توزیع احتمال حاصل به سمتی تغییر کند، که احتمال انتخاب عملیاتی را افزایش می‌دهد که منجر به پاداش تجمعی بالاتری می‌شوند. بیشتر مقالات موجود که از یادگیری تقویتی در سیستم‌های خودتطبیق استفاده می‌کنند، از یادگیری تقویتی مبتنی بر مقدار استفاده کرده‌اند. در این قسمت به چگونگی تعریف تابع مقدار یا همان تابع ارزش در کارهای گذشته می‌پردازیم. (آمویی و همکاران، ۲۰۰۸) از الگوریتم SARSA برای یادگیری عملیات تطبیقی مؤثر در یک برنامه وب خودتطبیق و 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 مقادیر آستانه‌ای تعریف شده توسط کارشناسان حوزه استفاده می‌شود. هوانگ و همکاران از Q-Learning برای بهینه سازی پویای تخصیص منابع در فرآیندهای تجاری استفاده کردند (کپارت و چیس، ۲۰۰۳) 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. دوتریل و همکاران از Q-Learning برای مدیریت منابع ابری خودمختار (دوتریل و همکاران، ۲۰۱۱) استفاده کردند. آنها فرض می‌کنند که کران بالایی برای متغیرهای حالت وجود دارد. بو و همکاران از Q-Learning برای پیکربندی خودکار ماشین‌های مجازی ابری و برنامه های کاربردی استفاده کردند (بو، رایو و ایکسو، ۲۰۱۳) برای تسهیل مقیاس‌پذیری، سه حالت برای گسسته‌سازی تعریف می‌کنند، که نشان دهنده بازه های بالا، متوسط و پایین متغیر حالت مربوطه است. عرب نژاد و همکارانش از Q-Learning فازی و SARSA برای ایجاد ابری با خاصیت خود مقیاسی استفاده می‌کند (عرب نژاد و همکاران، ۲۰۱۷) حالات محیطی کمی‌سازی 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. کاپوروسیو و همکاران پیشنهاد استفاده از یادگیری تقویتی مبتنی بر ارزش را برای مونتاژ خدمات چند عاملی می‌دهند (کاپورسیو و آنجلو، ۲۰۱۶). عامل‌ها اطلاعات نظارت بر وضعیت را به اشتراک می‌گذارند و از نمایش جدولی تابع مقدار استفاده می‌کند. سیلواندر پیشنهاد استفاده از Q-Learning با تقریب تابعی از طریق یک شبکه عصبی عمیق[4] برای بهینه‌سازی فرآیندهای تجاری دارد(سیلواندر، ۲۰۱۹). همه این رویکردها از الگوریتم حریصانه برای مکانیزم اکتشاف استفاده می کنند 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.

مروری مفهومی بر معماری روش پیشنهادی

شکل شماره (۳)، معماری مفهومی رویکرد پیشنهادی را نشان داده و مشخص می‌کند، چگونه عناصر یادگیری

1- Sensors
2- Effectors
3- Parametrizaed stochastic action selection policy
4- Deep Network Learning

تقویتی مبتنی بر سیاست در حلقه MAPE-k ادغام می‌شوند. ناحیه خاکستری تیره رنگ شکل شماره (۳) 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 بر عهده دارد.



شکل (۳): معماری مفهومی رویکرد یادگیری مبتنی بر سیاست

در زمان اجرا، منطق خودتطبیق (سیستم مدیریت کننده) از این سیاست برای انتخاب (از طریق نمونه گیری) یک عمل تطبیق، بر اساس وضعیت فعلی تعیین شده توسط فعالیت نظارت استفاده می‌کند. انتخاب عملیات، تعیین می‌کند آیا نیازی به اجرا (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 (ساتن، ۲۰۱۸؛ ساتن آلسدر، ۲۰۰۰) انجام شده، و سیاست، در قالب یک شبکه عصبی مصنوعی نشان داده می‌شود.

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

فرمول‌بندی رویکرد:

مسایل یادگیری تقویتی معمولا در قالب فرایند تصمیم مارکوف[1] MDP=(S,A,T,R) فرمول بندی می‌شوند 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 بار کاری محیط یا کارایی سیستم).

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

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

رابطه ۱ $T: S*A*S \to [0,1] = Pr(St+1|St, at).$

تابع پاداش[2] R مشخص کننده پاداش عددی که سیستم در هر حالت که قرار گرفت از محیط دریافت می کند.

رابطه ۲ $R: S \to IR+.$

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

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

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

احتمال انجام عمل تطبیقی a در حالت 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، 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 طور مثال حتی اگر بارکاری دقیق یک برنامه وب را ندانیم (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

1- Markov decision process
2- Reward function

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 بر سیاست، 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 و همکاران، ۲۰۱۷؛ وینز و ایگلسیا، ۲۰۱۵).

الگوریتم بهینه‌سازی سیاست پروگزیمال (PPO) از محبوب‌ترین الگوریتم‌های یادگیری تقویتی توسط تیم OpenAI در سال ۲۰۱۷ معرفی شد. این الگوریتم به جمع‌آوری دسته کوچکی از تجربیات در تعامل با محیط پرداخته و از آن دسته برای به روزرسانی سیاست تصمیم‌گیری استفاده می‌کند. الگوریتم PPO به جای تخصیص مقادیر به جفت‌های حالت - عمل، فضای سیاست‌ها را جستجو می‌کند. همچنین با محدود کردن تغییری که در سیاست خود در هر مرحله ایجاد می‌کند، به ثبات آموزش و بهینه سازی شبکه عصبی کمک می‌کند. رایج‌ترین پیاده‌سازی PPO از طریق مدل عامل منتقد است که از دو شبکه عصبی استفاده می‌کند، یکی وظیفه عامل (بازیگر) و دیگری محاسبه پاداش‌ها (منتقد) را انجام می‌دهد.

هدف مورد استفاده برای بهینه‌سازی شبکه عصبی با رابطه ۵ از احتمال انجام عمل تحت سیاست فعلی تقسیم بر احتمال انجام عمل تحت سیاست قبلی استفاده می‌کند.

$$r(\theta) = \Pi_{\theta(at|st)} / \Pi_{\theta old(at|st)}$$ رابطه ۵

این $r(\theta)$ زمانی بزرگ‌تر از ۱ خواهد بود که احتمال این عمل برای سیاست فعلی بیشتر از سیاست قدیمی باشد. زمانی که احتمال این عمل برای سیاست فعلی کمتر از سیاست قبلی باشد، بین ۰ و ۱ خواهد بود. برخلاف روش‌های گرادیان سیاست، PPO امکان می‌دهد توسط چند عامل یادگیرنده به طور موازی چندین دوره از شیب صعود را روی نمونه‌های مشاهده شده، اجرا کنیم؛ بدون اینکه به‌روزرسانی‌های سیاست بسیار مخربی ایجاد شود. این ویژگی این امکان را می‌دهد که اطلاعات بیشتری از محیط جمع آوری کرد و ناکارآمدی نمونه را کاهش داد. الگوریتم PPO با استفاده از یک تابع برش[3] از به روزرسانی سیاست های 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 کرده است. برای نمایش مدل‌های بازیگر - منتقد، این الگوریتم از یک شبکه پرسپترون چند لایه با دو لایه پنهان ۶۴ نورونی استفاده می‌کند (تعداد نرون‌ها در لایه‌ی ورودی و خروجی شبکه به تعداد متغیرهای عملیات و حالت سیستم بستگی دارد).

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

شکل شماره (۴): شبه کد الگوریتم ppo سبک عامل منتقد (اسچالمن و همکاران، ۲۰۱۷)

1- Actor-Critic.
2- Proxcimal Policy Optimization
3- Clipping

پیاده سازی 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 می‌باشد. 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 گسسته‌سازی فضای حالت. به ویژه، باید مراقب بود که مقایسه ناعادلانه‌ای انجام نشود. مقایسه ممکن است به شدت تحت تأثیر گسسته‌سازی یا به اصطلاح کوانتیزاسیون انتخابی قرار گیرد. گسسته‌سازی بسیار ریزدانه، ممکن است به این معنی باشد که یادگیری مبتنی بر ارزش همگرایی بسیار کندی خواهد داشت. از طرفی گسسته‌سازی خیلی درشت ممکن است به این معنی باشد که یادگیری مبتنی بر ارزش قادر به تمایز و تفکیک بین حالت‌های مختلف سیستم نیست و در نتیجه نمی‌تواند پاداش‌های تجمعی را بهینه کند. برای آزمایش از یک سیستم وب خودتطبیق حراجی برخط RUBiS-Brownout استفاده می‌کنیم (کلین و ماگیو، ۲۰۱۴). هنگامی که کاربر، مورد خاصی را از بین کالاهای حراجی درخواست می‌کند، موتور توصیه برنامه، لیستی از موارد توصیه شده را بر اساس حراجی‌های 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 تطبیق (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 توصیه، تجربه کاربر را افزایش می‌دهد، اما در عین حال برای اجرای این توصیه و راهنمایی، نیاز به منابع، افزایش یافته و در نتیجه ممکن است تأخیر را افزایش دهد و زمان انتظار کاربر برای پاسخ گرفتن به درخواست‌هایش زیاد شود، باعث نارضایتی کاربر گردد. همان طور که در جدول شماره (۱) نشان داده شده، مساله یادگیری را در قالب فرآیند تصمیم 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 باشد. هدف سیستم به حداکثر رساندن پاداش تجمعی است. فرض می‌کنیم که رضایت کاربر برای تأخیرهای بالاتر از آستانه λmax (که اینجا ۱۰ میلی ثانیه در نظر گرفته شده) کاهش می‌یابد، بنابراین تأخیرهای بالاتر از λmax جریمه می‌شوند.

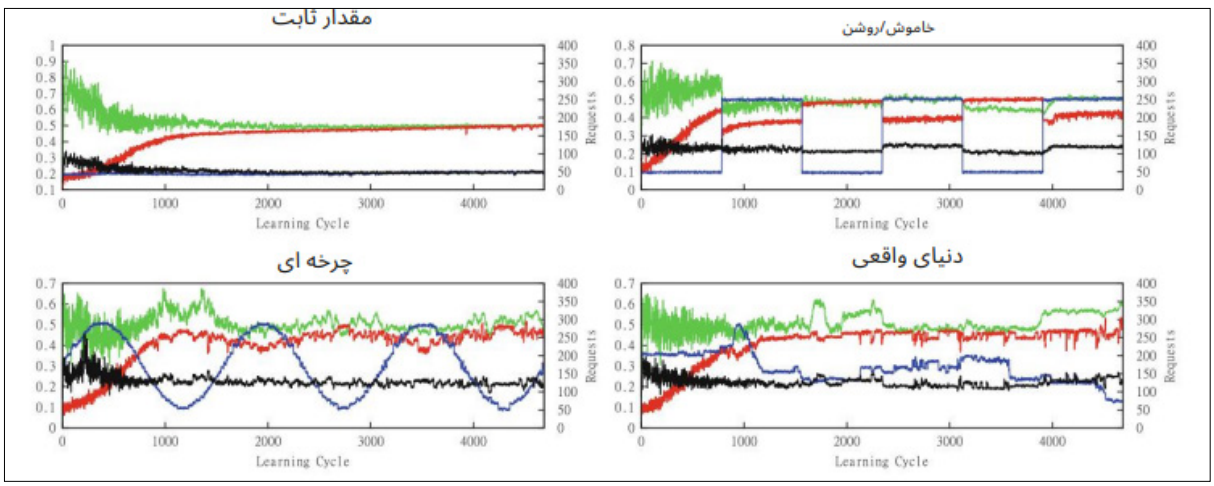
جدول (۱): یادگیری تقویتی برخط در برنامه وب خود تطبیق

تعداد درخواست های کاربر در لحظه t نرخ توصیه در زمان t تاخیر در زمان t	Ut ϵ IN+: αt ϵ[0,1] λt ϵIR+	State St=(Ut,αt,λt)
متغیر تطبیقی (دیمر)	A= δ ϵ [0,1]	Action αt ϵ A
λmax =10 ms	At ϵ[0,1] λt ϵIR+ f(λt) = 1 if λt < λma f(λt) = 0 if λt >2*λmax else f(λt) = - λt / λmax +2	Reward r(t)=αt* f(λt)

1- Dimmer value
2- Reward Function

مطابق جدول شماره (۱)، حالت سیستم به صورت سه‌گانه‌ای شامل تعداد درخواستهای کاربران، نرخ توصیه و میزان تاخیر در لحظه t تعریف می‌شود. نرخ توصیه همان مقدار متغیر تطبیق است که نسبت عکس با تاخیر دارد. اگر تاخیر در سیستم افزایش یابد، باید نرخ توصیه را کم کنیم و برعکس. بنابراین مقدار پاداش در هر لحظه بر اساس این دو ویژگی که اهداف تطبیقی سیستم است، تعیین می‌شود. مقدار پاداش در زمان t برابر با حاصل ضرب مقدار متغیر تطبیق که همان نرخ توصیه است (عددی بین صفر و یک). در تابعی از تاخیر سیستم در زمان t است. اگر تاخیر سیستم زیر ده میلی ثانیه (آستانه تحمل تاخیر) باشد، پاداش دریافتی در زمان t برابر است با مقدار متغیر تطبیق یا همان نرخ توصیه و چنانچه تاخیر بالاتر از ۲۰ میلی ثانیه باشد، پاداش دریافتی از محیط صفر است (زیرا خروجی تابع تاخیر صفر است). چانچه تاخیر از ده به سمت بیست میلی ثانیه حرکت کند میزان پاداش هم به صورت خطی کم می‌شود. یادگیرنده در چرخه سیستم خودتطبیقی یاد می‌گیرد، که در دراز مدت از این حالات (تاخیر بالای ده میلی ثانیه) پرهیز کند. هدف یادگیرنده‌ی تقویتی بیشینه‌کردن پاداش تجمعی است. 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 (کلین و ماگیو، ۲۰۱۴) 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 (لوربد، میگایل و لوزانو، ۲۰۱۴) استفاده کردیم: الگوی بارکاری ثابت[1] (تعداد ثابت درخواست ها)، الگوی خاموش / روشن[2] (منعکس کننده پردازش دسته ای و دوره ای)، و الگوی چرخه ای[3] (بارکاری درطی دوره افزایش و کاهش می یابد). همچنین بخشی از بار کاری ردیابی شده در دنیای واقعی را هم در نظر می گیریم. در این آزمایش از ۴۶۰۰ چرخه یادگیری برای هر کدام از الگوهای بار کاری، استفاده کردیم. هر چرخه یادگیری از داده‌های پایش شده ۱۲۸ مرحله زمانی متوالی قبل استفاده می‌کند نمودارهای شکل ۴ وضعیت، عملیات و پاداش برای هر چرخه یادگیری را به طور میانگین برای ۱۲۸ مشاهده از مراحل زمانی متوالی نشان را می دهند (مان، ۲۰۱۸).

شکل شماره (۵)، نتایج را برای انواع الگوهای بارکاری نشان می‌دهد. الگوی چرخه‌ای، الگوی (خاموش / روشن)، الگوی دنیای واقعی و الگوی ثابت.



شکل (۵): رفتار یادگیری تقویتی عمیق مبتنی بر سیاست در سیستم وب خودتطبیق (رنگ بارکاری=سیاه تاخیر=سیاه متغیرتطبیق یا دیمر=سبز پاداش=فرمز)

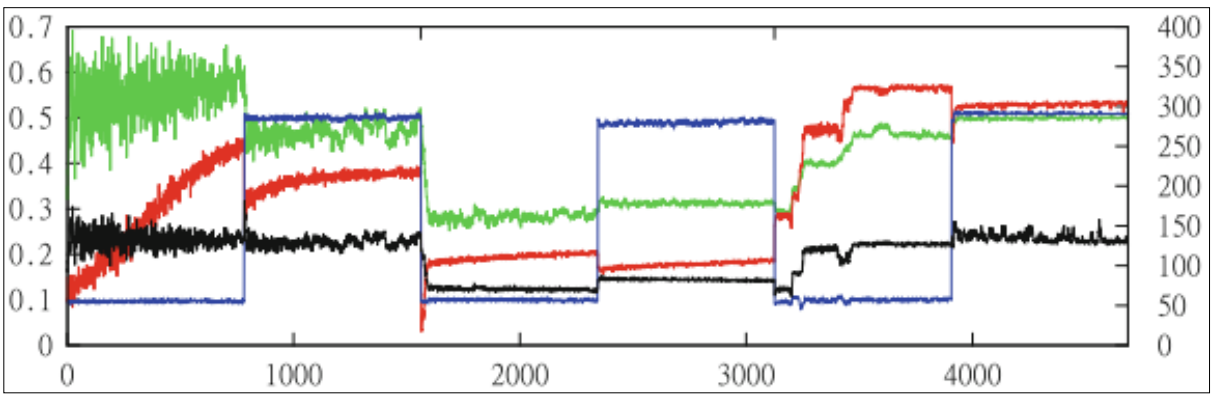
1- Constant workload pattern
2- On/Off workload pattern
3- Cyclic workload pattern

نتایج نشان می‌دهد که چگونه این نوع یادگیری تقویتی عامل منتقد مبتنی بر سیاست، سیستم را قادر می سازد تا به مرور زمان خود را تطبیق دهد. سیستم به طور خودکار متغیر تنظیمی برای تطبیق (دیمر) را، بسته به نوع الگوی بارکاری، تنظیم می‌کند، و در نتیجه تعادل بین تاخیر و افزایش تجربه کاربر با نرخ توصیه و پیشنهادات بیشتر ایجاد می‌کند (که در افزایش پاداش‌های تجمعی قابل مشاهده است). در ابتدای فرآیند یادگیری، مقدار کم متغیر تنظیم، 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 تطبیق در حدود نیم، تثبیت می‌شود و این حالت 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 در طول زمان استفاده مجدد کند. الگوی بارکاری چرخه‌ای، مشابه بارکاری خاموش / روشن، نشان می‌دهد، 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 تطبیق، پاداش را تقریبا در همان سطح نگه دارد، حتی اگر بارکاری در طول زمان تغییر کند. مزیت مهم این یادگیری این است که اگر بار کاری مشابه‌ای، دوباره تکرار شود، این رویکرد قادر است به سرعت عملیات تطبیقی موثر را پیدا کند.

شکل شماره (۶) 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، مقدار متغیر تطبیق (دیمر) را به گونه‌ای کاهش دهد که آستانه تأخیر (تاخیر از ۱۰ میلی ثانیه بیشتر نشود) نقض نشود. پس از چرخه یادگیری ۳۱۲۵ (رانش دوم) منابع را یک و نیم برابر افزایش دادیم. مجددا مقدار متغیر تطبیق (دیمر) بر این اساس تنظیم می‌شوند. این نوع یادگیری قادر است ویژگی غیرایستایی محیط را بدون انجام فرآیند نظارت صریح، تغییرات در منابع محاسباتی و بدون تغییر صریح نرخ اکتشاف، یاد بگیرد.



شکل (۶): رفتارالگوریتم یادگیری تقویتی عمیق مبتنی بر سیاست بر خط در محیط غیرثابت (آبی = بارکاری، سیاه = تأخیر، سبز = متغیر تطبیق دیمر، قرمز=پاداش)

1- Non_Stationary
2- Drift 1

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 برای فراپارامترهای شبکه استفاده شد. 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، 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. بنابراین نمی‌تواند به عملیاتی که قبلاً دیده نشده بودند، گسترش و تعمیم یابد. به طور معمول، سیستم‌های اطلاعاتی خودتطبیقی دارای فضاهای عملیاتی گسسته بزرگی هستند، مانند سیستم‌های خودتطبیق مبتنی بر ویژگی یا مبتنی بر معماری. به عنوان مثالی از سیستم خودتطبیق مبتنی بر ویژگی با فضای گسسته بزرگ، سیستمی را در نظر بگیرید که ۱۰ ویژگی اختیاری دارد. 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 ۱۰۲۴، عملیات 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 (دالاس، ایوان، سانگ و کوپینگ، ۲۰۱۵). عملکرد یادگیری ماشین تا حدود زیادی به مقدار داده موجود برای یادگیری بستگی دارد. هنگامی که از یادگیری تقویتی برای سیستم های خودتطبیقی استفاده میشود، ممکن است به چرخه های یادگیری زیادی نیاز باشد تا فرآیند یادگیری همگرا شود (مصطفی و ژانگ، ۲۰۱۴).

در آزمایش‌های انجام شده، یادگیری حدودا به ۲۰۰۰ چرخه یادگیری (با داده‌های ۲۵۶۰۰۰ مرحله زمانی (برای همگرایی نیاز داشت. تا وقتی یادگیری تقویتی همگرا نشود، سیستم به احتمال زیاد تطبیق‌های ناکارآمدی را اجرا می‌کند، زیرا هنوز مشاهدات کافی وجود ندارد. تطبیق‌های ناکارآممکن است منجر به اثرات منفی شود، زیرا در یک سیستم زنده اجرا می‌شوند (فیلهو و پورتر ۲۰۱۷). برای سرعت بخشیدن به همگرایی، یافتن برآوردهای اولیه خوب برای دانش آموخته شده (ساتن و بارتو، ۲۰۱۸؛ دوتریل ملخوف، ۲۰۱۱) یا انجام یادگیری برون خط از طریق شبیه‌سازی سیستم، می‌تواند استفاده شود (تزار، جانگ، داس و بنانی، ۲۰۰۷). در عین حال، یادگیری تقویتی برخط ممکن است برای سیستم‌هایی که در محیطی کار می‌کنند تا اثرات آزمایش و خطا یادگیری تقویتی، قابل تحمل نباشد (به عنوان مثال، اگر عملیات تطبیقی به محیط آسیب برساند)، قابل استفاده نباشد.

۳- نتیجه‌گیری

یادگیری تقویتی مبتنی بر ارزش در سیستم‌های خودتطبیق بکار گرفته می‌شدند این مقاله به معرفی و ارزیابی یادگیری تقویتی برخط مبتنی بر سیاست برای تسهیل در مهندسی سیستم‌های خودتطبیقی پرداخت. برای تسهیل یادگیری برخط تقویتی، به الگوریتمی نیاز است که بتواند به طور مداوم، سیاست را بدون انتظار برای نتیجه نهایی، یعنی بدون انتظار برای رسیدن به حالت پایانی، به روز کند.

الگوریتم‌های منتقد بازیگر یک نوع بدون مدل از الگوریتم‌های یادگیری تقویتی مبتنی بر سیاست هستند که در آن دانش به طور مداوم بدون انتظار برای نتیجه نهایی به‌روزرسانی می‌شود. این نوع یادگیرنده را در مدل چرخه بازخورد سیستم‌های خودتطبیق ترکیب شده که نقش تحلیلی را برای سیستم ایفا می‌کند. با توجه به تنوع داده‌ها در انواع الگوهای بار کاری (که لازمه هر الگوریتم تقویتی، حجم و تنوع داده‌های زیاد در حالات مختلف است)، می‌توان به دقت بالایی در تطبیق سیستم نسبت به رویکردهای مبتنی بر ارزش رسید. با طراحی یک تابع پاداش خوب که تمام ویژگی‌های هدف را دربردارد، توازن بالایی بین نرخ توصیه (بر اساس حجم درخواست‌ها) و میزان تاخیر برقرار کرد. رویکرد پیشنهادی این مقاله می‌تواند فضای عملیات بزرگ پیوسته را در حین یادگیری برخط کنترل کند (چون از شبکه عصبی به

جای راه جدولی بهره می‌برد). این رویکرد نه به حالت‌های محیطی گسسته‌شده به صورت دستی نیاز دارد، و نه به صورت دستی نیاز به تعیین نرخ اکتشاف مناسب همانند الگوریتم یادگیری تقویتی ارزش‌محور دارد. به عنوان کار آینده، این رویکرد می‌تواند برای مدیریت فضاهای عملیاتی گسسته بزرگ گسترش خواهد یافت. تا انواع بیشتری از سیستم‌های خودتطبیق را به تصویر کشید.

منابع

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.

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.

Aschoff, R., & Zisman, A. (2011). QoS-driven proactive adaptation of service composition. In *Service-Oriented Computing: 9th International Conference, ICSOC 2011, Paphos, Cyprus, December 5-8, 2011 Proceedings 9* (pp. 421-435). Springer Berlin Heidelberg.

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.

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.

Caporuscio, M., D'Angelo, M., Grassi, V., & Mirandola, R. (2016). Reinforcement learning techniques for decentralized self-adaptive service assembly. In *Service-Oriented and Cloud Computing: 5th IFIP WG 2.14 European Conference, ESOCC 2016, Vienna, Austria, September 5-7, 2016, Proceedings 5* (pp. 53-68). Springer International Publishing.

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.

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).

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*.

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).
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.
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.
Gheibi, O., Weyns, D., & Quin, F. (2021). Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 15*(3), 1-37.
Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41-50.
Klein, C., Maggio, M., Årzén, K. E., & Hernández-Rodriguez, F. (2014, May). Brownout: Building more robust cloud applications. In *Proceedings of the 36th International Conference on Software Engineering* (pp. 700-711).
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.
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.
Mann, Z. Á. (2017). Resource optimization across the cloud stack. *IEEE Transactions on Parallel and Distributed Systems, 29*(1), 169-182.
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.
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*.
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.
Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS), 4*(2), 1-42.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
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.
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*.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
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.
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.