چهاردهمین کنفرانس ملی سالانه انجمن کامپیوتر ایران
دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران) ایران، تهران – ۲۰ و ۲۱ اسفندماه ۱۳۸۷

موازنه‌ی بار در گرید با استفاده از یادگیری تقویتی SARSA(λ)

مهرداد صنوبری وایقان
دانشگاه تربیت مدرس
senobari@modares.ac.ir

نصرا... مقدم چرکری
دانشگاه تربیت مدرس
charkari@modares.ac.ir

چکیده: یکی از مسائل مطرح در محیط‌های گرید، مدیریت بارِ منابع و موازنه‌ی بار در سیستم می‌باشد. ناهمگونی منابع، پویایی محیط گرید و تنوع کارهای ورودی به گرید، از جمله مسائلی هستند که توسعه‌ی الگوریتم‌های بهینه‌ی موازنه بار را با مشکل مواجه می‌سازند. در این مقاله با مدل کردنِ محیط گرید بعنوان یک محیط یادگیری تقویتی و اعمال الگوریتم یادگیری تقویتی SARSA(λ) در تصمیم‌گیری‌ها، روشی برای تخصیص کارها به منابع و موازنه‌ی بار ارائه داده‌ایم. در این روش تخصیص کارهای ورودی به منابع بگونه‌ای است که سطح مطلوبی از توازن بار در گرید حاصل می‌شود. روش ارائه شده، در یک محیط گرید واقعی مورد آزمایش قرار گرفته و نتایج تجربی نشان‌دهنده‌ی کارآیی مناسب و تطبیق‌پذیری این روش با شرایطِ پویای گرید می‌باشد.

واژه های کلیدی: یادگیری تقویتی، SARSA(λ)، گرید، توازن بار.

۱- مقدمه

الگوریتم‌های یادگیری تقویتی (RL) از ابتدا تاکنون، پیشرفت‌های بسیاری داشته‌اند اما بجز در مواردی خاص، برای حل مسائل کاربردی در محیط‌های با مقیاسِ بزرگ کمتر مورد توجه قرار گرفته‌اند [۱].

گرید، بستری سخت‌افزاری و نرم‌افزاری می‌باشد که شامل شبکه‌ای از منابع ناهمگن بوده و دسترسی ارزان و استوار به منابع فوق را فراهم می‌سازد [۲]. مدیریت منابع و تخصیص بهینه‌ی کارها به منابع، یکی از مسائل مطرح در حوزه‌ی گرید می‌باشد. بهمین‌منظور تلاش‌های نیز برای مدل‌سازی محیط گرید انجام شده است اما افزایش ابعاد گریدها، پارامترهای مؤثر در تخصیص کارها به منابع و موازنه‌ی بار را نیز افزایش داده است. از اینرو مدل‌سازی گرید بسیار پیچیده بوده و ممکن است توام با اشتباه باشد [۳]. این امر بویژه در گریدهای سراسری[1] [۴] که تنوع منابع و کارهای ورودی به گرید در آن‌ها بسیار گسترده است، تاثیر بیشتری دارد. طی سال‌های اخیر، تلاش‌هایی برای استفاده از یادگیری ماشین در الگوریتم‌های زمان‌بندی صورت گرفته است [۵-۷]. یادگیری تقویتی، گونه‌ای از یادگیری ماشین است که کاربرد بسیاری در توسعه‌ی عامل‌های خودمختار[2] دارد.

یکی از مزیت‌های برخی الگوریتم‌های یادگیری تقویتی، عدم نیازِ آنها به مدل صریحی از سیستم می‌باشد. همچنین، سیاست‌های یادگرفته شده توسط یادگیری تقویتی خود را با تغییرات صورت گرفته در محیط وفق می‌دهند [۸]. الگوریتم‌های تقویتی مسائل را توسط تجربه کردن محیط حل می‌کنند. در بسیاری از سناریوهای تخصیص منابع، تاثیر هر تصمیم‌گیری بر موازنه‌ی سیستم سریعاً مشخص نمی‌شود و بسته به نوعِ کارِ ورودی و حالت فعلی سیستم، مدت زمانی طول می‌کشد تا بتوان تاثیر آن را محاسبه کرد. با وجودیکه این ویژگی توسعه‌ی بسیاری از الگوریتم‌های زمان‌بندی در چنین محیطی را با اشکال مواجه می‌کند، اما یادگیری تقویتی از این امر مستثنی می‌باشد. «پاداش‌های معوق»[3] از جمله راه‌حل‌هایی است که یادگیری تقویتی برای مواجهه با چنین مشکلاتی مورد استفاده قرار می‌دهد.

این مقاله به ارائه رویکردی مبتنی بر یادگیری تقویتی در مساله‌ی تخصیص منابع و موازنه‌ی بار در گرید می‌پردازد. ایده‌ی اصلی برپایه‌ی این فرض بناشده است که می‌توان محیط گرید را بطور تقریبی مارکوف درنظر گرفت [۷]، که در نتیجه اعمالِ یادگیری تقویتی به آن میسر می‌شود. الگوریتمِ نهایی حاصل از این روش، یک سیاستِ تخصیص کار به منابع را می‌آموزد که هدف اصلی آن توازن بار در سیستم است.

این الگوریتم در یک محیط گرید که با استفاده از عامل‌ها پیاده‌سازی شده است، تست شده و نتایج مشاهده شده حاکی از کارآیی و تطبیق‌پذیری آن با شرایط پویای محیط گرید می‌باشد.

ادامه مقاله بصورت زیر سازمان‌دهی شده است: در بخش ۲ مروری بر کارهای مرتبط انجام شده است؛ در بخش ۳ مساله‌ی زمان‌بندی و موازنه‌ی بار در گرید، و همچنین یادگیری تقویتی و الگوریتم SARSA(λ) بررسی شده و در بخش ۴ به ارائه‌ی روش پیشنهادی پرداخته و نحوه‌ی کاربردِ الگوریتم یادگیری تقویتیِ SARSA(λ) در این محیط را بیان خواهیم کرد. جزییات پیاده‌سازی سیستم و نتایج آزمایشات در بخش ۵ شرح داده شده و بخش ۶ نیز نتیجه‌گیری مقاله خواهد بود.

[1] Global Grid
[2] Autonomic Agents
[3] Delayed Rewards

۲- کارهای مرتبط

Galstyan و همکاران در [۹] از یادگیری تقویتی برای تخصیص منابع استفاده کرده‌اند. سیستم آن‌ها شامل تعداد زیادی عامل‌های یادگیر است که تعدادی منبع را برای استفاده‌های محاسباتی بصورت اشتراکی دراختیار دارند. عامل‌های یادگیر بصورت جداگانه ویژگی‌های محیط را یاد می‌گیرند و با یکدیگر ارتباط ندارند. این عامل‌ها از نسخه‌ی ساده شده‌ای از Q-Learning برای یادگیری استفاده می‌کنند. فرض شده است که توان محاسباتی هر منبع و زمان مورد انتظار برای اجرای هر کار برروی یک منبع با توان محاسباتی واحد، از پیش مشخص است. بعلاوه در مدل آن‌ها فرض شده است که در هر لحظه فقط یک کار برروی هر منبع اجرا می‌شود. نتایج حاصله از اجرای روش نیز در یک محیط شبیه‌سازی شده ارائه شده است. نتایج مقاله‌ی فوق قابل توجه است، اما برخی پیش‌فرض‌های مقاله در محیط‌های گرید واقعی صادق نیست و همین امر امکان کاربرد روش آن‌ها در محیط‌های واقعی را دچار تردید می‌کند.

در [۷] نیز از یادگیری تقویتی برای حل مساله‌ی تخصیص سرور به برنامه‌های وب استفاده شده است. نوآوری مقاله‌ی فوق استفاده‌ی همزمان از یادگیری تقویتی غیربرخط و مدل‌های صف، بمنظور تسریع در فرآیند یادگیری می‌باشد. در ابتدای اجرای سیستم، تخصیص سرورها براساس الگوریتم‌های مبتنی بر مدل انجام می‌شود. سپس اطلاعات حاصله از آن تصمیم‌گیری‌ها برای آموزش بخشِ یادگیری تقویتی استفاده می‌شود. بدیت ترتیب، از تصمیم‌گیری‌های نادرست (و بعضاً پرهزینه) در ابتدای شروع بکار سیستم جلوگیری شده است.

رویکرد Gao و همکاران در [۵] استفاده از الگوریتم ژنتیک بمنظور پیش‌بینی زمان اجرای کارها در گریدِ سرویس بوده است. در گریدِ سرویس، یک یا چند منبع، سرویسی خاص را به کاربران ارائه می‌دهند. کاربران داده‌های ورودی خود را به گرید محول کرده و زمان‌بند با توجه به حجم داده‌های ورودی، یکی از منابع را برای اجرای سرویس انتخاب می‌کند. برای ایجاد توازن بار، در مقاله‌ی فوق پیشنهاد شده است که می‌توان مدل کارآیی محاسباتی هر منبع را حین اجرا یاد گرفت و براساس آن، زمان اجرای هر کارِ جدید را برای تمامی منابع محاسبه کرده و سپس با استفاده از الگوریتم ژنتیک حالت بهینه را برای کارهای ورودی بدست آورد. با وجود کارآیی مناسب، این روش فقط برای گریدهای سرویس مناسب است و نمی‌توان آن را در گریدهای عمومی هم مورد استفاده قرار داد.

Liu و همکاران نیز در [۱۰] به مساله توازن بار در گرید با منابع همگن پرداخته‌اند. این کار توسط مجموعه‌ای از مورچه‌ها انجام می‌شود. کارآیی و همگرایی این روش هم بصورت بیان رسمی و هم شبیه‌سازی در مقاله‌ی فوق به اثبات رسیده است.

پروژه NetSolve [۱۱] از الگوریتم‌های زمان‌بندی مختلف برای هر کدام از انواع کارها استفاده می‌کند. مدل بار و کارآیی کارها بصورت تجربی مدل شده و زمان اتمام هر کار بوسیله آن تخمین زده می‌شود. با دردست داشتن زمان اتمام کارها و متوسط نرخ پاسخگویی منابع، تخصیص کارها انجام می‌شود.

Buyya و همکاران نیز در [۱۲] رویکردی مبتنی بر اقتصاد خرد برای مدیریت منابع و زمان‌بندی در گرید ارائه کرده‌اند.

۳- موازنه‌ی بار و یادگیری تقویتی

هدف این مقاله ارائه روشی برای کاربرد یادگیری تقویتی در مساله‌ی موازنه‌ی بار در گرید می‌باشد. لذا در این بخش ابتدا خلاصه‌ای از مساله‌ی زمان‌بندی و موازنه‌ی بار در گرید مطرح شده و سپس مروری بر یادگیری تقویتی و الگوریتم SARSA(λ) انجام خواهد شد.

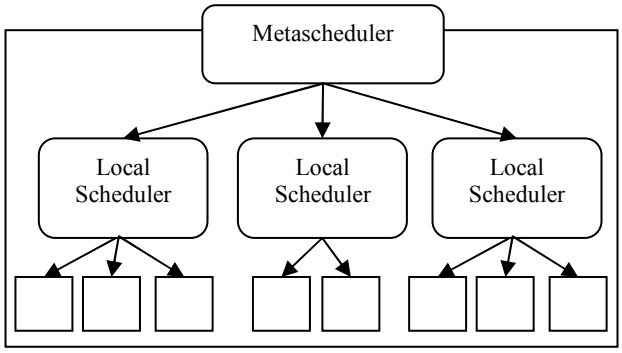
۳-۱ زمان‌بندی و موازنه‌ی بار در گرید

گرید شامل تعدادی منابع توزیع شده می‌باشد که هر کدام از این منابع ممکن است تحت سیاست‌های متفاوتی به اشتراک گذاشته شوند. مدیریت منابع در گرید شامل کشف و ارائه‌ی منابع، و همچنین زمان‌بندی، تحویل کار به منابع و نظارت می‌باشد.

منظور از زمان‌بندی در گرید، یک نگاشت از کارها به منابع می‌باشد. این نگاشت متاثر از معیارهای مختلفی می‌باشد [۱۳]. موازنه‌ی بار منابع، یکی از معیارهایی است که بخصوص در گریدهای محاسباتی و سراسری مورد توجه می‌باشد. افزایش گذردهی سیستم، کمینه نمودن زمان اجرای کارها، تحمل خرابی و بهره‌برداری بهینه از منابع، از دیگر اهداف زمان‌بندها بشمار می‌روند. بدیهی است که دستیابی به تمامی اهداف فوق در یک زمان‌بند امکان‌پذیر نمی‌باشد، چرا که برخی از آن‌ها در تقابل با یدیگر می‌باشند.

تاکنون ساختارهای متفاوتی برای زمان‌بندهای گرید ارائه شده است: «متمرکز»، «توزیع‌شده» و «سلسله مراتبی» [۳]. ساختار ۲ سطحی (شکل ۱) که در دسته‌ی سلسله مراتبی قرار می‌گیرد در بسیاری از گریدها دیده می‌شود، منجمله در Globus که یکی از اصلی‌ترین بسترهای گرید می‌باشد [۲]. در سطح اولِ این ساختار، «فرا-زمان‌بند»[4] قرار دارد. ورود کارها به گرید از طریق فرا-زمان‌بند صورت می‌گیرد. فرا-زمان‌بند به تعدادی «زمان‌بند محلی» دسترسی دارد و کارها را به آن‌ها اختصاص می‌دهد. هر کدام از زمان‌بندهای محلی می‌توانند راهبرد متفاوتی برای اختصاص منابع درنظر گرفته باشند. از آن‌جا که زمان‌بندهای محلی به اطلاعاتِ دقیق‌تری از وضعیت منابع تحت کنترل خود دسترسی دارند، زمان‌بندی وظایف در این سطح می‌تواند با دقت بالاتری انجام پذیرد.

[4] Metascheduler



شکل ۱- ساختار ۲ سطحی زمان‌بند در گرید

در این مقاله، زمان‌بندهای سطح دوم (زمان‌بندهای محلی) موردتوجه قرار گرفته‌اند. ایده‌ی این مقاله قابل اعمال به فرا-زمان‌بند نیز می‌باشد که این مساله در کارهای آتی مورد ارزیابی قرار خواهد گرفت.

۲-۳ یادگیری تقویتی

یادگیری تقویتی (RL) روشی است که یک عامل توسط آن می‌تواند کنش‌های بهینه با محیط را یاد بگیرد. این یادگیری از طریق کاوش در محیط، آزمون و خطا، و استفاده از پاداش‌های دریافتی توسط محیط انجام می‌گیرد [۸]. در حالت کلی هر عامل پس از دریافتن «حالت» محیط، یک «کنش» انجام می‌دهد و در قبال آن، از محیط پاداش دریافت می‌کند. نتایج آزمون و خطا «تجربه» نام دارد و توسط ۴تایی «حالت فعلی، کنش، پاداش، حالت بعدی» بیان می‌شود. پاداش، میزان سودمندی هر کنش در موقعیت‌های مختلف را بیان می‌کند. هدف نهایی هر عامل در یادگیری تقویتی، بیشینه نمودن مجموع پاداش‌هایی است که از محیط دریافت می‌کند.

یادگیری تقویتی، در پی یافتن سیاستی برای تصمیم‌گیری‌های متوالی در محیط برای رسیدن به یک هدفِ خاص است. برخلاف برخی روش‌های یادگیریِ ماشین، یادگیری تقویتی عملیات تصمیم‌گیری را سریع انجام می‌دهد اما شرط همگرایی آن تجربه کردن تمامی حالاتِ محیط بصورت دوره‌ای است. از این‌رو، در محیط‌های با فضای حالت بزرگ استفاده از تقریب‌گر تابع در فرآیند RL اجتناب ناپذیر است[۱۴].

فرض کنیم $X = \{ x_1, x_2, x_3, \ldots, x_n \}$ مجموعه‌ی حالت‌هایی است که یک محیط می‌تواند در آن قرار گیرد، و $A = \{a_1, a_2, \ldots, a_n\}$ مجموعه‌ی کنش‌های قابل انجام توسط عاملِ یادگیر باشد. توسط RL، عامل یادگیر تلاش می‌کند تا سیاست بهینه‌ی $\pi^*(x) \in A$ را به ازای تمامی مقادیر ممکنِ x پیدا کند. سیاست $\pi^*(x)$ مجموع پاداش‌های تخفیف یافته‌ی[5] موردانتظار عامل در طول زمان را بیشینه می‌کند. تابع کنش-مقدار[6] برای سیاست بهینه π^* را می‌توان بصورت زیر تعریف کرد:

$$Q^*(s,a) = E\{ r(x,a) + \gamma V^*(x_{t+1}) \mid x_t = x, a_t = a\} \quad (۱)$$

که در آن $V^*(x_{t+1})$ مقدار سیاست بهینه برای حالت x_{t+1} است. Q-Learning و SARSA از جمله الگوریتم‌های یادگیری تقویتی هستند که برپایه تابع کنش-مقدار (تابع Q) عمل می‌کنند. در مواردی که تعداد حالات محیط و کنش‌ها محدود باشد، می‌توان از جدول برای نگهداری تابع Q استفاده کرد. اما همانگونه که اشاره شد، در محیط‌هایی با فضای حالت پیوسته یا بسیار بزرگ بایستی از یک تقریب‌گر تابع مانند شبکه عصبی استفاده کرد.

تحت برخی شرایط الگوریتم‌های یادگیری تقویتی همگرا به سیاست کنترلی بهینه می‌باشند. این همگرایی در هنگام استفاده از تقریب‌گر تابع به اثبات نرسیده است اما بصورت تجربی همگرایی آن مورد قبول واقع شده است [۱، ۸، ۱۴].

۱-۲-۳ روش یادگیری تقویتی SARSA(λ)

SARSA(λ) یک شیوه‌ی یادگیری تقویتی است که بصورت برخط عمل می‌کند، بدین معنی که از داده‌های تجربه با محیط، در همان لحظه استفاده کرده و سیاست فعلی خود را با آن تطبیق می‌دهد. این روش یادگیری تقویتی در حل مسائل کنترلی کاربرد بسیاری دارد [۱]. شبه کد آن در شکل ۲ آورده شده است [۸]:

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$ for all s, a.
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q
  Repeat (for each step of episode):
    Take action a, observe reward r, s'
    Choose a' from s' using policy derived from Q
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s, a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
      $e(s,a) \leftarrow \gamma \lambda e(s,a)$
    $s \leftarrow s'; a \leftarrow a';$
  until s is terminal

شکل ۲- شبه کد الگوریتم SARSA(λ)

[5] Expected Discounted Reward
[6] Action-Value

در کد فوق γ ضریب تخفیف[7] بوده و α نیز نرخ یادگیری می‌باشد. مقادیر $e(s,a)$ نیز Eligibility Trace نام دارند و بمنظور تعیین میزان انتساب از پاداشِ فعلی به کنش‌هایی است که قبلاً انتخاب شده‌اند [۸]. پارامتر λ درصد این انتساب را مشخص می‌کند.

در شبه کد شکل ۲، بروز رسانی تابع Q در الگوریتم SARSA(λ)، توسط فرمول زیر انجام می‌شود:

$$Q_{t+1} = Q_t(s,a) + \alpha\, \delta_t\, e_t(s,a), \quad \text{به‌ازای تمامی مقادیر } a \text{ و } s \qquad (۲)$$

که در آن:

$$\delta_t = r_{t+1} + \gamma\, Q_t(s_{t+1},a_{t+1}) - Q_t(s_t, a_t)$$

و $e_t(s,a)$ نیز به ازای تمامی مقادیر s و a برابر است با:

$$\begin{cases} \gamma\lambda\, e_{t-1}(s,a) + 1 & \text{اگر } s=s' \text{ و } a=a' \\ \gamma\, e_{t-1}(s,a) & \text{در غیر این‌صورت} \end{cases}$$

۴- روش پیشنهادی

با توجه به مباحث طرح شده، هدف از بکارگیری یادگیری تقویتی در این مقاله تخصیص عادلانه‌ی کارها به منابع در جهت دستیابی به توازن بار در سیستم است. در ادامه‌ی این بخش به ارائه‌ی روش پیشنهادی خواهیم پرداخت.

۴-۱ گرید بعنوان محیط یادگیری تقویتی

می‌توان محیط گرید را بعنوان یک محیط یادگیری تقویتی درنظر گرفت که هدف اصلیِ عامل یادگیر در آن، رسیدن به سطح مطلوبی از توازن بار میان گره‌ها است. با این قیاس، «محیط» متشکل از پارامترهایی همچون «تعداد منابع»، «میزانِ بارکاری منابع»، «در دسترس بودن هرکدام از منابع»، و «تعداد کارهای ورودی به سیستم» است. بر همین اساس، «کنش» عامل در هر لحظه بصورت انتخاب یکی از منابع برای اجرای کارِ بعدی می‌باشد. «پاداش» را نیز می‌توان تابعی از سطح موازنه‌ی بار در کل سیستم تعریف کرد.

با فرض حضور n منبع در سیستم، حالت سیستم در زمان t را می‌توان بصورت یک n+1 تایی تعریف کرد:

$$s_t = \langle l_1, l_2, .. l_m, j \rangle \qquad (۳)$$

که در آن l_i بیانگر میزان بارِکاری منبع i-ام بوده و j نیز تعداد کارهای تخصیص نیافته در سیستم است. ممکن است منابع در برخی از زمان‌ها دردسترس نباشند (مانند مشکلات سخت‌افزاری یا موارد مشابه)، در این حالت میزان بارِکاری آن منبع، ۱ درنظر گرفته می‌شود.

می‌توان زمان‌بند را بصورت عاملی فرض کرد که در محیطی با فضای حالتِ بیان شده، به یادگیری مشغول است. ورود هر کارِ جدید به سیستم باعث می‌شود تا عامل یک کنش در محیط انجام دهد. کنش در این محیط، انتخاب یکی از منابع و تخصیص کار جدید به آن منبع می‌باشد. پاداش این تخصیص هم در هنگام بروز رسانی حالت محیط مشخص خواهد شد. در هنگامی که منابع میزان بارِ خود را به عامل زمان‌بند گزارش می‌دهند، عامل زمان‌بند حالت جدید و حالت فعلی را با یکدیگر مقایسه می‌کند. در صورتیکه حالت فعلی سیستم به توازن بار نزدیکتر باشد، پاداش به عامل تعلق خواهد گرفت و در غیر این‌صورت هیچ پاداشی به آن تعلق نمی‌گیرد.

۴-۲ شرایط محیط

برای مدل کردن محیط گرید بعنوان یک محیط RL، فرضیات و شرایط زیر در نظر گرفته شده است:

(۱) در مورد کارهای ورودی به سیستم فرض شده است که هیچ اطلاع قبلی از نوع و زمان اجرای کارها دردسترس نبوده و کارها نیز مستقل از هم می‌باشند؛ این فرض در بسیاری از گریدها و بویژه در گریدهای سراسری برقرار است.

(۲) از تاخیر شبکه چشم‌پوشی شده است. از آن‌جاییکه عامل زمان‌بند در سطح محلی قرار دارد، می‌توان از تاخیر شبکه صرفنظر کرد.

(۳) مقدار بار هر منبع، تابع متغیرهای بسیاری است. در این مقاله بارِ هر منبع را بصورت تابعی از درصد مصرف پردازنده و حافظه در نظر گرفته‌ایم. درنظر نگرفتن مقدار مصرفی پهنای باند، متاثر از پیش‌فرضِ مستقل بودن کارهای ورودی بوده است. در صورتیکه کارهای ورودی از هم مستقل باشند، میزان استفاده از پهنای باند شبکه بصورت قابل توجهی کاهش یافته و قابل چشم‌پوشی می‌گردد [۱۵].

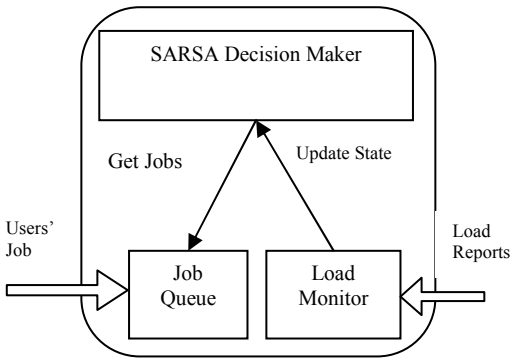
(۴) فرض می‌شود که منابع ممکن است بصورت تصادفی از سیستم خارج شده و پس از مدتی دوباره به سیستم متصل شوند.

۴-۳ عامل زمان‌بند

همانگونه که ذکر شد، تصمیمات تخصیص منابع به کارها، توسط عامل زمان‌بند انجام می‌شود. کارهای ورودی به سیستم، توسط عامل زمان‌بند دریافت شده و تصمیم‌گیری براساس شرایط محیط انجام می‌شود. ساختار عامل زمان‌بند از ۳ واحد اصلی تشکیل شده است:

- واحد نظارتِ بار
- واحد تصمیم گیرنده‌ی SARSA
- صف کارها

[7] Discount factor



شکل ۳ - ساختار عامل زمان‌بند و نحوه‌ی ارتباط واحدهای آن

بمنظور تطبیق الگوریتم SARSA(λ) با ساختار عامل زمان‌بند و رخدادهای سیستم، آن را به ۳ مدول مستقل تقسیم نموده‌ایم که جزییات آن در شکل ۴ آورده شده است.

RL_Initialization:

1. Initialize **Q(s,a)** arbitrarily
2. e(s,a) = 0 for all s, a
3. Initialize ***s***
4. Choose ***a*** randomly

Job_Receive:

1. **s** ← Determine current state
2. **A(s)** ← Create action List
3. Choose **a** from **A(s)** using policy derived from **Q**
4. e(s, a') = e(s, a') + 1
5. Apply action **a'**

Load_Report:

1. **s'** ← Determine new state
2. **r** ← Calculate reward
3. **δ** ← r + γ*Q(s', a') – Q(s, a)
4. e(s, a) ← e(s,a) + 1
5. **for all s_i, a_i**
   $Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\, \delta e(s_i, a_i)$
   $e(s_i, a_{ij}) \leftarrow \gamma*\lambda*e(s_i, a_i)$
6. s ← s' ; a ← a';

شکل ۴- مدول‌های الگوریتم SARSA(λ) که در عامل زمان‌بند پیاده‌سازی شده است.

واحد نظارتِ بار پس از دریافت هر گزارش از منابع، حالت جدید سیستم را به واحد تصمیم‌گیرنده اطلاع می‌دهد. واحد تصمیم‌گیرنده، پاداش آخرین تصمیم‌گیری را براساس رابطه‌ی زیر محاسبه می‌کند:

$$y = \mathrm{std}(l_1, l_2, \ldots, l_n)$$
$$Y = 0.05 \qquad (۴)$$

$$r_t = \begin{cases} 1 & \text{اگر } y < Y \\ 0 & \text{در غیر اینصورت} \end{cases}$$

که در آن Y حد مطلوب سیستم برای موازنه‌ی بار بوده و y نیز برابر با انحراف معیار بار منابع می‌باشد. سپس مقدار تابع Q براساس زیربرنامه Load_Report بروز رسانی می‌شود.

هر کارِ جدید پس از ورود به سیستم، در صف کارها قرار می‌گیرد. واحد تصمیم‌گیرنده، کارها را بترتیب ورود آن‌ها (FCFS) پردازش می‌کند. اگر کارِ جدیدی در صف وجود داشت، ابتدا یک لیست از کنش‌های ممکن ایجاد می‌شود. این لیست شامل تمامی منابعی است که ازدیاد بار ندارند. همواره یک کنش ویژه نیز به لیست اضافه می‌شود که «کنش تاخیر» نام دارد و هدف از آن، به تاخیر انداختن اجرای کارِ جدید است. با فرض اینکه $L(a_i)$ میزان بار فعلی گره a_i ، Delay_Action کنش تاخیر و Load_Threshold نیز حد آستانه سیستم برای ازدیاد بار باشد، فهرست کنش‌ها بصورت زیر خواهد بود:

$$\text{Actions} = \{a_1, a_2, a_3, \ldots, \textit{Delay\_Action}\} \qquad (۵)$$
$$L(a_i) <= \text{Load\_Threshold} \quad \text{که در آن:}$$

پس از ایجاد لیست، یک کنش با استفاده از روش ε-greedy انتخاب می‌شود (زیربرنامه‌ی Job_Receive در شکل ۴). در روش ε-greedy، با احتمال ε یک کنش بصورت اتفاقی انتخاب می‌شود و درغیر اینصورت کنشی که بیشترین مقدار Q برای حالتِ فعلی سیستم را دارد انتخاب می‌شود. این روش در یادگیری تقویتی بمنظور کاوش در فضای حالت محیط استفاده می‌شود [۸].

پس از انتخاب کنش، در صورتیکه «کنش تاخیر» انتخاب شده باشد هیچ کاری از ابتدای صف برداشته نمی‌شود، و درصورتیکه کنش‌های دیگر انتخاب شوند، یک کارِ جدید از صف برداشته شده و به منبعِ متناظر برای اجرا فرستاده می‌شود.

۵- پیاده‌سازی و نتایج

محیط گرید که در آزمایشات مورد استفاده قرار گرفته است، یک محیط مبتنی بر عامل می‌باشد که توسط بستر برنامه‌نویسی JADE [۱۶] و زبان برنامه‌نویسی Java توسعه یافته است. محیط تست شامل ۸ منبع و ۱ عامل زمان‌بند بوده است. بر روی هر کدام از منابع، یک عامل اجرا می‌شود که وظیفه‌ی ارتباط با عامل زمان‌بند و اجرای کارهای محول شده را بر عهده دارد. عاملِ منبع، میزان بار آن را طی بازه‌های از پیش تعیین شده به عامل زمان‌بند گزارش می‌دهد. همانگونه که قبلاً نیز اشاره شد، منابع ممکن است بصورت تصادفی از سیستم خارج شده و پس از مدتی دوباره به سیستم متصل شوند.

در پیاده‌سازی عامل زمان‌بند، با توجه به فضای حالت بزرگ مساله و عدم امکان نگهداری تابع Q(s,a) بصورت جدول در حافظه، از یک شبکه عصبی پیشخور با یک لایه‌ی پنهان بعنوان تقریب‌گر استفاده شده است. تعداد گره‌های لایه پنهان بصورت تجربی و درحین آزمایشات، ۳۰ انتخاب شده است. α برابر با ۳×۱۰^{-۲} و ε نیز برابر با ۵×۱۰^{-۲} انتخاب شده است.

محیط هر تست شامل تعدادی منبع بوده که از لحاظ حافظه و پردازنده متفاوت می‌باشند. منابع، میزان بارِ خود را در بازه‌های ۱۲۰ ثانیه به عامل گزارش می‌دهند. میزان بار، مضربی از ۰.۰۵ بوده و در بازه‌ی [۰،۱] است.

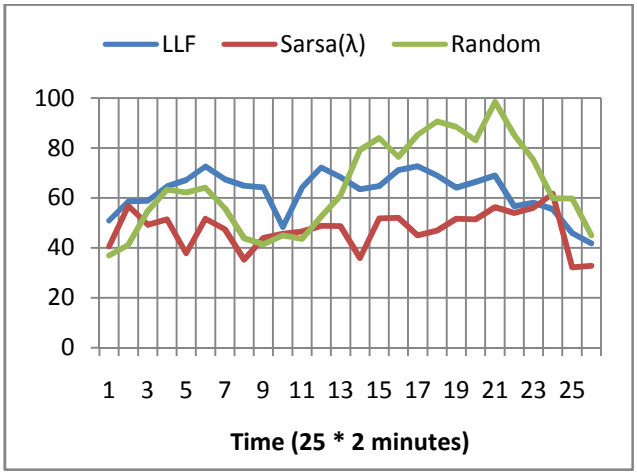
زمان ورود کارها به سیستم، بصورت تصادفی توسط توزیع نمایی انتخاب می‌شود. ویژگی‌های هر کار نیز بصورت تصادفی در هنگام اجرای آن مشخص می‌شود. این ویژگیها شامل «نوع» (پردازنده متمرکز یا حافظه متمرکز)، «میزان بار»، و «زمان اجرا»ی آن کار می‌باشد. زمان اجرای کار بصورت تصادفی از بازه‌ی [۱،۱۵] دقیقه انتخاب شده است.

در ادامه‌ی این بخش نتایج حاصل از ارزیابی روش پیشنهادی ارائه خواهد شد. این ارزیابی‌ها شامل (۱) مقایسه کارآیی روش با الگوریتم‌های زمان‌بندی دیگر و (۲) مقیاس‌پذیری روش در برابر تعداد منابع می‌باشد.

۵-۱ ارزیابی کارآیی

بمنظور تست کارآیی روش، ۲ الگوریتم برای مقایسه انتخاب شده است: (۱) الگوریتم تصادفی؛ که تخصیص کارهای جدید به منابع را بصورت تصادفی انجام می‌دهد، (۲) الگوریتم Least Loaded First (LLF): این الگوریتم در هنگام ورود کارِ جدید به سیستم، منبعی را به آن اختصاص می‌دهد که کمترین بارِ کاری را دارد. هر دو الگوریتم بدون صفِ کارها پیاده‌سازی شده‌اند و بلافاصله پس از ورودِ هر کار به سیستم، آن را به یک منبع تخصیص می‌دهند.

تعداد منابع این مورد استفاده در این تست، ۸ بوده است و عامل زمان‌بندِ SARSA نیز پس از یادگیری مورد آزمایش قرار گرفته است. آموزش عامل زمان‌بند طی ۴۸ ساعت اجرای سیستم انجام شده است. متوسط بارِ منابع طی اجرا برای هر ۳ الگوریتم در شکل ۵ نمایش داده شده است:



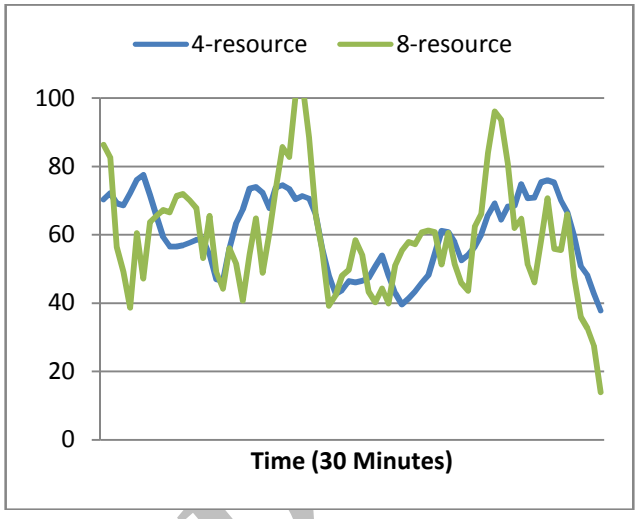
شکل ۵- مقایسه کارآیی روش پیشنهادی: متوسط بارِ کاری منابع در یک بازه‌ی زمانی

با وجودیکه بنظر می‌رسد الگوریتم LLF بایستی بهترین کارآیی را داشته باشد، اما همانطور که در شکل ۵ مشاهده می‌شود، روش SARSA از کارآیی نسبی بهتری برخوردار است. این امر از آنجا ناشی می‌شود که: (۱) الگوریتم LLF پیاده‌سازی شده در این تست، امکان سربارِ منابع را درنظر نمی‌گیرد (بعلت عدم وجود صف کارها). اما الگوریتم SARSA با استفاده از «کنش تاخیر» و قرار دادن کارها در صف انتظار، احتمال ایجاد سربار در منابع را کمتر می‌کند. (۲) کارآیی الگوریتم LLF نسبت مستقیمی با دقتِ اطلاع آن از بارِ فعلی منابع دارد. از آن‌جایی‌که بازه‌ی زمانی گزارش بارِ منابع در این تست ۱۲۰ ثانیه درنظر گرفته شده است، دقتِ الگوریتم نیز LLF کاهش یافته است. (۳) الگوریتم LLF ناهمگن بودن منابع را درنظر نمی‌گیرد، در حالیکه الگوریتم SARSA ویژگی‌های منابع را بصورت ضمنی در طی فرآیند تخصیص منابع یادگرفته است.

۵-۲ ارزیابی مقیاس‌پذیری

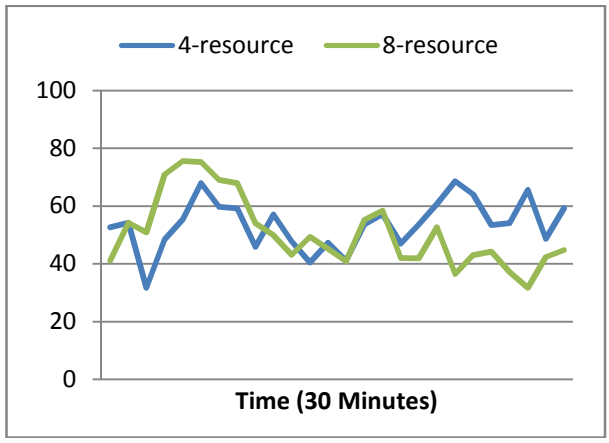
کارآیی الگوریتم‌های یادگیری تقویتی نسبت مستقیمی با مدت زمان اجرای آن (و در واقع، تعداد تعامل‌های عامل با محیط) دارد. بنابراین یکی از آزمایشات انجام شده در این مقاله، تغییر تعداد منابع از ۴ به ۸ و بررسی تاثیر آن بر یادگیری بوده است.

در این آزمایش، ابتدا تعداد منابع را ۴ درنظر گرفتیم و یک سناریوی تصادفی از کارها را به آن اعمال کردیم. سپس همین آزمایش با ۸ منبع انجام شد. چنین تغییری، بزرگ شدن فضای حالت و همچنین تعدادِ کنش‌های ممکن در سیستم را به همراه دارد که تاثیر مستقیمی بر زمان یادگیری و همگرایی سیستم دارد.



شکل ۶ – انحراف معیار بار منابع یک بازه‌ی زمانی پس از ۱۸ ساعت اجرا

همانطور که انتظار می‌رود و در شکل ۶ نیز دیده می‌شود، اغتشاش بیشتری در انحراف معیار بارِ منابع هنگامی‌که ۸ منبع در سیستم حضور دارد ، مشاهده می‌شود. در حالتی که تنها ۴ منبع در سیستم وجود دارد، فضای حالت شامل ۵ متغیر می‌باشد. با افزایش تعداد منابع به ۸ منبع، تعداد متغیرهای فضای حالت نیز به ۹ افزایش می‌یابد. در نتیجه مدت زمان و تعداد تجربه‌های لازم برای یادگیری شرایط محیط نیز افزایش یافته است.



شکل ۷ – انحراف معیار بار منابع در یک بازه‌ی زمانی پس از ۴۸ ساعت اجرا

در شکل ۷ مشاهده می‌شود که پس از مدت زمان مناسب، الگوریتم برای ۸ منبع به پایداری نسبی رسیده و کارآیی آن قابل مقایسه با نتایج حاصله از اجرای روش با ۴ منبع است. این پایداری بصورت تجربی مشاهده شده و همانطور که در بخش ۳-۲ نیز بیان شد، تاکنون اثباتی در زمینه‌ی همگرایی الگوریتم‌های تقویتی درهنگام استفاده از تقریب‌گر تابع ارائه نشده است اما بصورت تجربی همگرایی آن مورد قبول است.

۶- نتیجه گیری

در این مقاله روشی برای کاربرد یادگیری تقویتی بمنظور زمان‌بندی با هدف توازن بار در گرید ارائه دادیم. عامل زمان‌بند در این سیستم با استفاده از روش SARSA(λ) ویژگی‌های محیط را یادگرفته و تخصیص کارهای جدید را بر مبنای آن انجام می‌دهد. همانگونه که مشاهده شد، این عمل بدون ساختن مدل صریحی از محیط گرید انجام می‌شود.

از آن‌جاییکه الگوریتم SARSA(λ) بصورت عمومی بیان شده است، تغییراتی برای تطبیق با شرایط محیط در آن انجام شد. ساختار عامل زمان‌بند نیز موردبحث قرار گرفت. جزییات پیاده‌سازی سیستم و ارتباطات اجزای آن نیز تشریح شد. در مرحله‌ی تست، روش پیشنهادی از ۲ بُعد مورد بررسی قرار گرفت. ابتدا کارآیی سیستم در برابر الگوریتم‌های تخصیص تصادفی و LLF تست شد و نتیجه‌ی آن ارائه شد. آزمایش بعدی، مقیاس‌پذیری روش دربرابر تعداد منابع بود که همانگونه که نتایج بیانگر آن بودند؛ روش پیشنهادی، به شرط گذراندن تعداد تجربه‌های کافی در محیط و زمان یادگیری طولانی‌تر، در برابر تعداد منابع مقیاس‌پذیر است. بصورت کلی نتایج تجربی حاکی از موفقیت روشِ پیشنهادی دارد.

شرایط و مدت زمان لازم برای همگرایی روش، از جمله مواردی است که بعنوان مسائل آتی این تحقیق موردبررسی قرار خواهند گرفت.

مراجع

[1] Stone, P., R.S. Sutton, and G. Kuhlmann, *Reinforcement Learning for RoboCup Soccer Keepaway.* Adaptive Behavior, 2005. **13**(3): p. 165-188.
[2] Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* Int. J. High Perform. Comput. Appl., 2001. **15**(3): p. 200-222.
[3] Rajkumar Buyya, K.K.a.M.M., *A taxonomy and survey of grid resource management systems for distributed computing.* Softw. Pract. Exper., 2002. **32**(2): p. 135-164.
[4] Buyya, R., et al., *Utility Computing and Global Grids.* Arxiv preprint cs/0605056, 2006.
[5] Gao, Y., H. Rong, and J.Z. Huang, *Adaptive grid job scheduling with genetic algorithms.* Future Generation Computer Systems, 2005. **21**(1): p. 151-161.
[6] Kesselman, I.F.a.N.R.J.a.C., *Brain Meets Brawn: Why Grid and Agents Need Each Other.* Autonomous Agents and Multiagent Systems, International Joint Conference on, 2004. **1**: p. 8-15.
[7] Tesauro, G.J., N.K. Das, R. Bennani, M.N. *A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation*. in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*. 2006.
[8] Sutton, R.S. and A.G. Barto, *Reinforcement learning: an introduction.* IEEE Trans Neural Netw, 1998. **9**(5): p. 1054.

[13] Dong, F., *A Taxonomy Of Task Scheduling Algorithms In The Grid*. Parallel Processing Letters (PPL), 2007. **17**(4): p. 439 - 454

[14] Kalyanakrishnan, S. and P. Stone, *Batch reinforcement learning in a complex domain*, in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. 2007, ACM: Honolulu, Hawaii.

[15] Yu, J. and R. Buyya, *A Taxonomy of Workflow Management Systems for Grid Computing.* Journal of Grid Computing, 2005. **3**(3): p. 171-200.

[16] Szymczak, M., et al., *Resource Management in an Agent-based Virtual Organization.* Web Intelligence and Intelligent Agent Technology, International Conference on, 2007: p. 458-462.

[9] Galstyan, A., K. Czajkowski, and K. Lerman, *Resource Allocation in the Grid with Learning Agents.* Journal of Grid Computing, 2005. **3**(1): p. 91-100.

[10] Jin, X. and Y. Wang, *Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization.* IEEE Trans. Parallel Distrib. Syst., 2005. **16**(7): p. 586-598.

[11] Seymour, K., et al., *NetSolve: Grid Enabling Scientific Computing Environments.* Grid Computing and New Frontiers of High Performance Processing, 2005.

[12] Buyya, R., et al., *Economic models for resource management and scheduling in Grid computing.* Concurrency and Computation: Practice and Experience, 2002. **14**(13-15): p. 1507-1542.