

ارائه مکانیسمی کارآمد برای تعیین تابعی برای توزیع بارکاری در محاسبات مه با استفاده از سیستم‌های دسته‌بند یادگیر

مهدی عباسی، بهاره حمیدی محب *

گروه مهندسی کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران

مقاله پژوهشی

چکیده

تاریخ دریافت:

۱۴۰۲/۴/۸

تاریخ پذیرش:

۱۴۰۲/۱۲/۲۲

کلیدواژه‌ها:

اینترنت اشیا، پردازش مه، پردازش لبه، تخمین تابع، سیستم‌های دسته‌بند یادگیر

نویسنده مسئول:

b.hamidimoheb@eng.basu.ac.ir

با توسعه سریع اینترنت اشیا، الگوی محاسبات مه به‌عنوان یک راه‌حل جذاب برای پردازش داده‌ها در برنامه‌های اینترنت اشیا ارائه شده است. در محیط مه، برنامه‌های اینترنت اشیا توسط گره‌های محاسباتی میانی در مه و همچنین کارگزارهای فیزیکی در مراکز داده ابری اجرا می‌شوند. از این رو، مسائل مربوط به مدیریت منابع و مدیریت انرژی به‌عنوان یکی از مشکلات چالش‌برانگیز در محاسبات مه باید مورد توجه قرار گیرد. اخیراً پژوهش‌هایی برای ایجاد تعادل بین انرژی و هزینه در محاسبات مه صورت گرفته است. در این پژوهش ضمن بررسی این رویکردها روشی کارآمد برای تقریب تابع توزیع بار با دو روش مبتنی بر سیستم‌های یادگیر دسته‌بند به نام XCSF و BCM-XCSF در گره‌های پردازشی مه به‌منظور بهینه‌سازی هرچه بیشتر رویکردهای قبلی و مدیریت منابع پردازشی مه ارائه شده است. این دو روش در داشتن یک حافظه برای نگهداری بهترین دسته‌بندها باهم متفاوت هستند. نتایج شبیه‌سازی نشان می‌دهد دو روش XCS و BCM-XCS تفاوت توزیع بار مناسبی دارند. این دو روش سربار محاسباتی را کم می‌کنند. روش BCM-XCSF افزون بر این که حدود ۶۰ درصد تأخیر را کاهش می‌دهد مصرف انرژی را نیز بهبود می‌بخشد.



: 10.22034/ABMIR.2024.20271.1030



۱- مقدمه

در اینترنت اشیاء میلیاردها موجودیت فیزیکی برای جمع‌آوری و مبادله داده‌ها به‌منظور ارائه برنامه‌های کاربردی، به یکدیگر متصل می‌شوند [۱]. محاسبات ابری راه‌حلی برای پردازش داده‌های بزرگ ارائه می‌دهد. با ظهور اینترنت اشیاء، محاسبات ابر با چالش‌های رو به رشدی مانند تأخیرها، محدودیت پهنای باند، منابع محدود دستگاه‌ها، اهمیت ارتباط امن و بدون وقفه مواجه است [۲، ۳]. بر اساس تحقیقات، برای کاهش این مشکلات، به‌جای استفاده از پردازش‌های ابری دور و متمرکز، از منابع غیرمتمرکز در لبه‌های شبکه برای پردازش داده‌های نزدیک به کاربر می‌توان استفاده کرد. این موضوع به‌عنوان محاسبات مه‌آلود یا محاسبات لبه نامیده می‌شود [۴، ۵]. در محاسبات مه‌آلود، پردازش‌ها با تأخیر کمی انجام می‌شوند. از طرفی تأمین انرژی برای انجام این محاسبات سنگین دشوار است. انرژی تجدیدپذیر تا حدودی این دشواری را جبران می‌کند. همچنین، پیوستار محاسباتی در اینترنت اشیاء شامل لایه‌های اشیاء، لبه، مه و ابر است که هر یک ظرفیت‌های محاسباتی متفاوتی دارند. بنابراین، لازم است تصمیم بگیریم به هر لایه چه میزان بارکاری تخصیص یابد یا به هر بخش چه تعداد وظیفه اختصاص یابد. تصمیم‌گیری‌ها باهدف بهبود معیارهای تأخیر، پهنای باند و هزینه صورت می‌گیرد [۶]. نکته مهم در این تخصیص‌ها آن است که از منابع موجود در لبه شبکه حداکثر بهره‌وری به عمل آید. تحقیقات اخیر نشان داده است که از روش‌های یادگیری تقویتی می‌توان برای به حداقل رساندن زمان بیکاری گره‌های لبه شبکه و استفاده بهینه از منابع و خدمات مه استفاده کرد [۷].

در زمینه‌های یادگیری تقویتی، می‌توان از سیستم‌های دسته‌بندی‌کننده یادگیر استفاده کرد. در این سیستم‌ها، تقریبی از تابع بازخورد ارائه می‌شود که به سیستم اجازه می‌دهد برای هر ورودی، بهترین کنش با بالاترین بازخورد مثبت را انتخاب کند [۸]. سیستم‌های دسته‌بندی‌کننده یادگیر، سیستم‌هایی مبتنی بر قانون هستند. در این سیستم‌ها قوانین عمدتاً به فرم "IF condition THEN action" هستند. اصطلاح LCS^۱ نخست توسط هلند معرفی شد و در ابتدا توسعه‌ای برای الگوریتم‌های ژنتیکی بود [۹]. در دو دهه

اخیر LCS استفاده زیادی در حل مسائل واقعی چون داده‌کاوی داشته است [۱۰، ۱۱].

افزایش توان محاسباتی تجهیزات ابری درکنار محدودیت منابع در لبه‌های شبکه، انگیزه‌ای برای جستجو راه‌حل‌های جدید جهت به حداکثر رساندن بهره‌وری منابع این محاسبات شده است. این پژوهش درصدد یافتن راه‌حلی کارآمد برای توزیع بارهای کاری به نحوی مطلوب بین منابع موجود در لبه شبکه است ضمن کاهش مصرف انرژی تأخیر را هم کاهش دهد. به این منظور، در محاسبات مه‌آلود از سیستم دسته‌بندی‌کننده یادگیر برای کمک به تصمیم‌گیری ایستگاه پایه جهت تعیین میزان بار پردازشی در لبه شبکه استفاده می‌شود. در مدل‌سازی بارها در محاسبات مه، ورودی این سیستم (شرایط) وضعیت شبکه و خروجی (کنش) آن میزان بار پردازشی در لبه شبکه است. با استفاده از یک الگوریتم XCSF میزان بار مناسب به شکل یک تابع برای پردازش در لبه شبکه پیش‌بینی می‌شود. پس از یادگیری ضرایب این تابع، می‌توان از آن برای محاسبه بار توزیع بار در زمان‌های آتی استفاده کرد. در این صورت، پس از یادگیری تابع توزیع بار، به‌ازاء ورود بارکاری جدید به لبه شبکه، نیازی به اجرای الگوریتم یادگیری مذکور نخواهد بود و باهمان تابع یادگیری شده، مقدار بار توزیعی در لبه مشخص خواهد شد.

در ادامه این مقاله پژوهش‌های انجام‌شده در این حوزه، بررسی می‌شود. در بخش سوم روش پیشنهادی به‌صورت مفصل توضیح داده می‌شود. به این منظور، مدلی برای هزینه‌های سیستم بررسی می‌شود و مسئله توزیع بار در لبه شبکه مدل‌سازی و فرمول‌بندی می‌شود. در بخش چهارم، نحوه پیاده‌سازی و ارزیابی روش پیشنهادی توضیح داده می‌شود. درنهایت، بخش پایانی مقاله به نتیجه‌گیری و راهکارهای آتی می‌پردازد.

۲- مروری بر پژوهش‌های انجام‌شده

در سال‌های اخیر، به‌ویژه پس از ۲۰۱۶ میلادی، پژوهش‌های متعددی در زمینه توزیع بهینه بار محاسباتی در لبه‌های شبکه صورت گرفته است. در ادامه، تعدادی از کارهای انجام‌شده در

¹Learning Classifier System

انرژی تجدید پذیر به عنوان ورودی سیستم، میزان بهینه توزیع بارکاری در گره‌های لبه محاسبه می‌شد. همچنین از یک حافظه اضافی در سیستم XCS برای تسریع روند یادگیری استفاده شد. تأخیر در پردازش بارهای کاری و سطح باتری در XCS و XCS حافظه دار پیشنهادی در تحقیق مذکور، به طور قابل توجهی بهتر از هر روش اخیر بود. الگوریتم ارائه شده توانست تعادلی میان دو معیار انرژی مورد نیاز برای محاسبات مه‌آلود و تأخیر ایجاد کند و همچنین یک مدل جامع برای توزیع بار محاسباتی، بین گره‌های پردازش مه ارائه دهد؛ با این وجود، در هر بار ورود بار محاسباتی جدید به لایه مه، الگوریتم باید اجرا گردد که این مورد سربار محاسباتی قابل توجهی را به سیستم وارد می‌کند.

۳- روش پیشنهادی

فرض می‌کنیم که یک سیستم لبه از ایستگاه پایه و تعدادی از کارگزارهای لبه که از لحاظ فیزیکی نزدیک به هم قرار دارند، تشکیل شده است. هر کدام از کارگزارهای مستقر در لبه شبکه دارای ظرفیت محدود باتری می‌باشند. یک منبع تغذیه پشتیبان نیز در لبه شبکه مستقر شده تا در صورت کمبود منابع در کارگزارهای لبه، بارهای کاری را به ابر منتقل کند. این بار محاسباتی از سمت کاربران وارد لبه شبکه شده و به ایستگاه پایه ارائه می‌شود. در این سناریو، وظیفه ایستگاه پایه تصمیم‌گیری میزان بار پردازشی در لبه شبکه و انتقال مابقی آن به کارگزارهای ابر است.

۳-۱ مدل سازی

پیوستار پردازش بار از لبه تا ابر از چهار منظر مدل می‌شود که شامل: مدل بارکاری، مدل تأخیر، مدل مصرف انرژی و مدل باتری است. برای هر یک از این مدل‌ها فرمول‌هایی با توجه به مرجع [۱۲] ارائه می‌شود.

مدل بارکاری

بازه‌های زمانی در سناریوهای کاری سیستم با اعداد گسسته مانند $t = 0, 1, 2, 3, \dots$ نمایش داده می‌شود. در هر بازه زمانی باید تعداد کارگزارهای فعال در لبه شبکه تعیین شود. $L(t)$ نرخ رسیدن کل بارکاری در لحظه t به لبه شبکه است به طوری که رابطه (۳) برقرار باشد. همچنین، L_{max} حداکثر و L_{min} حداقل بار ورودی

حوزه بهبود مصرف انرژی و هزینه پردازش در محاسبات مه بررسی می‌شود.

اولین پژوهش‌ها در سال ۲۰۱۶ ارائه شد و امکان ترکیب انرژی‌های تجدید پذیر در محاسبات لبه‌های شبکه تلفن همراه را بررسی کردند. روش‌های ارائه شده عمدتاً برای بهبود مصرف انرژی بوده و عمدتاً کند بودند [۱۲].

در سال ۲۰۱۷ Jie Xu و همکارانش [۱۳] یک الگوریتم یادگیری تقویتی مبتنی بر تراکم طیفی انرژی ارائه دادند تا توزیع بار و تخصیص خودکار منابع در لبه شبکه به صورت بهینه انجام شود. الگوریتم فوق با وجود اینکه توانست مصرف انرژی را برای پردازش بارهای محاسباتی در لبه‌های شبکه بهبود بخشد، نمی‌توانست این بارهای کاری را به صورت متعادل در میان گره‌های پردازشی توزیع کند.

در سال ۲۰۱۸ Hang Wu و همکارانش [۱۴]، الگوریتمی را به نام GLOBE ارائه دادند. آن‌ها در این الگوریتم، دو مکانیسم تعادل بار جغرافیایی و کنترل ورودی بارها را برای بهینه‌سازی عملکرد ایستگاه‌های لبه شبکه، با هم ترکیب کردند. با اینکه این الگوریتم توزیع بارهای محاسباتی در میان گره‌های شبکه را بهبود بخشید، اما به حالت بهینه دست نمی‌یافت.

در سال ۲۰۱۸ Jiafu Wan و همکارانش [۱۵]، یک روش متعادل سازی توزیع بار محاسباتی برای بهبود مصرف انرژی ربات‌ها در محاسبات مه‌آلود را پیشنهاد کردند. نتایج کار ایشان نشان داد که در بستر ایجاد شده، نسبت به سایر ربات‌ها کارایی بهتری نسبت به سایر روش‌های زمان‌بندی داشته و بارهای کاری به صورت متوازن‌تری در میان آن‌ها توزیع شده است.

در سال ۲۰۱۸ Deze Zeng و همکارانش [۱۶]، مسئله بهره‌وری انرژی را به صورت یک مساله برنامه‌ریزی خطی ارائه دادند و ثابت کردند که یک مسئله NP-hard است. نتایج حاصل از شبیه‌سازی نشان داد که بهره‌وری انرژی در روش آن‌ها، نزدیک به راه‌حل بهینه است.

در سال ۲۰۲۰، مهدی عباسی و همکارانش [۱۷]، روشی پیشنهاد دادند که از سیستم دسته‌بند یادگیری به نام XCS استفاده می‌کرد. در این پژوهش، با تطبیق پارامترهایی شامل کل بارکاری و مقدار

ظرفیت باتری است. باتری‌ها توسط انرژی تجدید پذیر مانند باد و خورشید شارژ می‌شوند که با $g(t)$ نمایش داده می‌شود. در ابتدا میزان انرژی باتری صفر فرض شده است. در لبه شبکه ممکن است در هر بازه زمانی دو حالت برای باتری رخ بدهد:

اگر $b(t) > E_{op}(t)$ باشد، مقداری از بارهای کاری ($L_f(t)$) در لبه شبکه پردازش و مابقی به ابر ارسال می‌شود. در این حالت میزان باتری برای بازه زمانی بعدی از رابطه (۶) محاسبه می‌شود.

$$b(t+1) = b(t) + g(t) - E((L(t), L_f(t), S(t))) \quad (6)$$

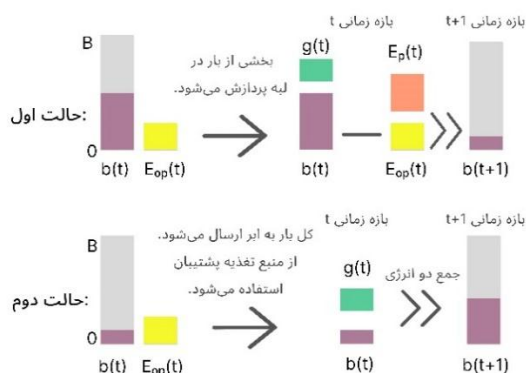
در این حالت هزینه‌ای که به سیستم تحمیل می‌شود هزینه استهلاک باتری خواهد بود که از رابطه **Error! Reference source not found.** به دست می‌آید.

$C_{bt}(t) = \beta \cdot \max\{E((L(t), L_f(t), S(t)) - g(t), 0)\} \quad (7)$
که $\beta < 0$ هزینه استهلاک یک واحد باتری است. حالت اول در شکل (۱) این موضوع را به تصویر می‌کشد.

اگر $b(t) \leq E_{op}(t)$ باشد، کل بارکاری به ابر ارسال خواهد شد تا باتری توسط منبع تغذیه پشتیبان شارژ شود. هزینه این عملیات از رابطه (۹) حاصل می‌شود که در آن، φ ضریب هزینه منبع تغذیه پشتیبان است. میزان انرژی باتری در بازه زمانی بعدی با فرمول (۹) محاسبه می‌شود. حالت دوم در شکل (۱) بیانگر این موضوع است.

$$C_{bak}(t) = \varphi \cdot E_{op}(t) \quad (8)$$

$$b(t+1) = b(t) + g(t) \quad (9)$$



امکان‌پذیر به لبه شبکه است. مقدار بارکاری که تصمیم گرفته شده به صورت محلی پردازش شود، با پارامتر $L_f(t)$ نمایش داده می‌شود. بین این سطوح بارکاری رابطه (۳) همواره برقرار است. میزان بار قابل پردازش در ابر $L_c(t)$ از رابطه (۳) به دست می‌آید. در این مدل‌سازی، $S(t)$ تعداد کارگزارهای فعال در لبه شبکه در نظر گرفته می‌شود که در طول بازه زمانی ثابت است.

$$L(t) \in [L_{min}, L_{max}] \quad (1)$$

$$L(t) \geq L_f(t) \quad (2)$$

$$L_c(t) = L(t) - L_f(t) \quad (3)$$

مصرف انرژی

در هر بازه زمانی، کل مصرف انرژی متشکل از دو نوع انرژی است:

- انرژی لازم برای عملیات پایه و انتقال بارها در لبه شبکه که با $E_{op}(t)$ نمایش داده می‌شود. میزان این انرژی فقط به میزان کل بار ورودی یعنی $L(t)$ وابسته است و از رابطه (۴) به دست می‌آید.

$$E_{op}(t) = E_{dyn}(t) + E_{sta}(t) \quad (4)$$

مقدار $E_{dyn}(t)$ انرژی مصرفی متغیر در لبه شبکه است که به میزان بارکاری ورودی بستگی دارد و در این سناریو به دلیل نزدیکی مراکز پردازش، صفر در نظر گرفته می‌شود. مقدار $E_{sta}(t)$ نیز انرژی مصرفی ثابت در لبه شبکه است.

- انرژی موردنیاز برای پردازش بارهای کاری در لبه شبکه با $E_p(t)$ نمایش داده می‌شود. این انرژی برحسب میزان بار قابل پردازش در لبه شبکه و تعداد کارگزارهای فعال محاسبه می‌شود.

طبق رابطه (۵) مصرف انرژی از جمع انرژی عملیاتی و انرژی لازم برای پردازش بارهای کاری به دست می‌آید.

$$E(L(t), L_f(t), S(t)) = E_{op}(t) + E_p(t) \quad (5)$$

باتری

مجموعه کل میزان باتری‌های موجود در لبه شبکه به شکل $b(t) \in [0, B]$ نمایش داده می‌شود. نماد B نشان‌دهنده حداکثر میزان

شکل (۱): دو حالت موجود در عملکرد باتری

تأخیر

در این سناریو سه مدل تأخیر داریم. نخستین تأخیر، تأخیر انتقال بارهای کاری در شبکه بی‌سیم $d_{trans}(t)$ است که برحسب مقدار بار قابل‌پردازش در لبه شبکه به دست می‌آید. مقدار این تأخیر در این سیستم به دلیل نزدیکی فیزیکی مراکز پردازش به مراکز تولید بارکاری صفر فرض می‌شود.

نوع دوم تأخیر، مربوطه به پردازش بارهای کاری به صوت محلی و در لبه شبکه است که با $d_{lo}(t)$ نمایان می‌شود. سبک محاسبه این تأخیر به مکانیسم مدیریت صف در لبه شبکه بستگی دارد. از آنجایی که در مدل موردنظر ما، از مکانیسم مدیریت صف $M/G/1$ استفاده می‌شود، این مقدار توسط رابطه (۱۰) به دست می‌آید. در رابطه مذکور α ظرفیت پردازش هر کارگزار است.

$$d_{lo}(t) = L_f(t) / (S(t) \cdot \alpha - L_f(t)) \quad (10)$$

تأخیر ارسال باقی‌مانده بارهای محاسباتی به ابر با $dc(t)$ نمایش داده می‌شود و مقدار آن وابسته به ازدحام شبکه $H(t)$ است. ازدحام شبکه مجموع تأخیر انتقال رفت و برگشت و تأخیر پردازش در ابر است. در نتیجه این تأخیر طبق رابطه (۱۱) محاسبه می‌شود.

$$d_c(t) = (L(t) - L_f(t)) H(t) \quad (11)$$

در نهایت هزینه تأخیر کل بارکاری که به سیستم وارد می‌شود بر اساس رابطه (۱۲)، از مجموع سه تأخیر فوق حاصل می‌شود.

$$d_{total}(H(t), L(t), L_f(t), S(t)) = d_{trans}(t) + d_{lo}(t) + d_c(t) \quad (12)$$

۲-۳ سیستم دسته‌بند یادگیر

در این بخش، ابتدا به مرور الگوریتم XCS و سپس XCSF می‌پردازیم و برای بهبود عملکرد آن‌ها یک حافظه به آن اضافه می‌کنیم.

۱-۲-۳ الگوریتم XCS

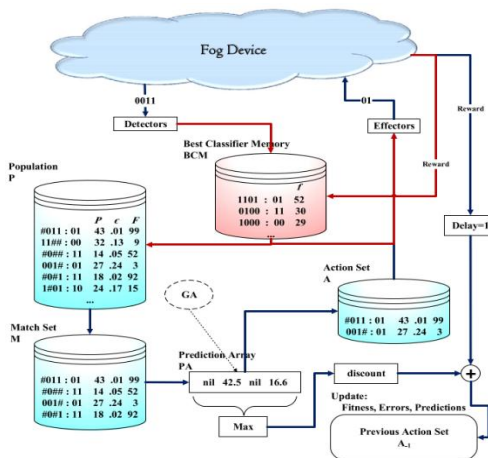
موارد زیر که از مقاله [۱۸] برگرفته شده است، شرح مختصری از XCS است. در XCS یک ورودی ارائه می‌شود؛ سیستم تصمیم می‌گیرد و محیط، مقداری به‌عنوان بازده ارائه می‌کند. [P] شامل تمام قوانین تکامل‌یافته است. این جمعیت را می‌توان از مجموعه‌ای از قوانین شناخته‌شده برای هر کلاس یا یک قانون کاملاً عمومی واحد به‌طور تصادفی تولید کرد. مجموعه [M] شامل مجموعه‌ای از قوانین فعال‌شده به‌وسیله نمونه ورودی است. زمانی که یک

کلاس پیش‌بینی‌شده از قوانین در [M] انتخاب شود، مجموعه عمل [A] شامل همه کنش‌های برگزیده کلاس قوانین [M] خواهد بود. از نظر ساختاری، هر قانون دسته‌بند Cj در جمعیت [P] دارای یک شرط، یک عمل و مجموعه‌ای از پارامترهای مرتبط است. شرط، رشته‌ای از $\{0,1,\#\}$ است. هر عمل در یک قانون دسته‌بند به‌صورت یک عدد صحیح نمایش داده می‌شود. سه پارامتر اصلی در به‌روزرسانی و انتخاب دسته‌بندها وجود دارند. (۱) پیش‌بینی بازده pj، که در صورت مطابقت Cj با انتخاب آن توسط سیستم، بازدهی را که سیستم دریافت خواهد کرد را تخمین می‌زند، (۲) خطای پیش‌بینی ε_j ، خطا را در pj با توجه به بازده‌های واقعی دریافتی تخمین می‌زند و (۳) قانون برازش Fj.

یک چرخه عملیاتی یا مرحله زمانی در این سیستم به این صورت است با دریافت اطلاعات اولیه از محیط در حافظه [P] جستجو می‌شود. آن دسته از دسته‌بندهایی که با ورودی مطابقت دارند در مجموعه مطابقت [M] ذخیره می‌شوند. طی محاسباتی آرایه پیش‌بینی که به تعداد اکشن‌ها خانه دارد، پر می‌شود. با استفاده از این آرایه بهترین عمل‌ها در مجموعه [A] قرار می‌گیرد و به سیستم اعمال می‌شود در قبال این عمل از محیط یک مقدار بازده دریافت می‌شود. در این بخش از طبق الگوریتم یادگیری تقویتی، پارامترهای دسته‌بندها به‌روزرسانی شده و در مراحل بعدی استفاده می‌شود.

در XCS در صورتی که ورودی با دسته‌بندی از مجموعه [P] منطبق شود، ولی مجموعه [A] کامل نباشد، سایر عمل‌ها به‌صورت تصادفی با عمل پوشش در قالب دسته‌بند جدید ایجاد شده و به مجموعه [P] اضافه می‌شوند. این دسته‌بندهای جدید می‌تواند در نتیجه ورودی‌های بعدی تأثیرگذار باشد. بدین جهت باهدف عدم تولید دسته‌بندهای تصادفی در صورت تطبیق ورودی با یک دسته‌بند موجود در [p] در مقاله [۱۷] به سیستم XCS یک حافظه جانبی اضافه شده است. این حافظه در Error! Reference source not found. به نام BCM نشان داده شده است. در این حافظه جدید، شرایط پیش‌آمده همراه با بهترین عمل مناسب برای آن و بهترین پاداشی که آن عمل از محیط دریافت کرده، نگهداری می‌شود و

وزن بردار دسته‌بند W محاسبه می‌شود. بر این اساس، در سیستم XCSF بشر بنیم، P تابعی از حالت فعلی s و عمل a است.



شکل (۲): XCS و BCM-XCS [۱۷]

در نتیجه، پیش‌بینی سیستم برای عمل a در حالت st به فرم

$P(st, a)$ به صورت (۱۴) تعریف می شود:

$$P(s_t, a) = \frac{\sum_{cl \in [M] | a \cdot cl, p(s_t) \cdot cl, F}}{\sum_{cl \in [M] | a \cdot cl, F}} \quad (18)$$

در رابطه فوق، cl یک دسته‌بند است، $a[M]$ نشان‌دهنده زیرمجموعه‌ای از دسته‌بندها در $[M]$ با عمل a است، F, cl برازش دسته‌بند cl است. (st) پیش‌بینی cl است که در حالت st محاسبه می‌شود. به‌طور خاص، هنگامی که تقریب خطی تکه‌ای در نظر گرفته می‌شود، (st) cl, p به‌صورت (۱۵) محاسبه می‌شود:

$$cl, p(s_t) = cl, w_0 \cdot x_0 + \sum_{i \geq 0} cl, w_i \cdot s_t(i) \quad (15)$$

در این رابطه، $cl.wi$ وزن wi از دسته‌بند و $x0$ یک ورودی ثابت است. مقادیر (st, a, P) آرایه پیش‌بینی را تشکیل می‌دهند. در مرحله بعد، XCSF یک عمل را برای انجام انتخاب می‌کند. دسته بندهای موجود در $[M]$ که عمل انتخاب‌شده در آن موجود است، در مجموعه عملکرد فعلی $[A]$ قرار می‌گیرند. عمل انتخاب‌شده به محیط ارسال می‌شود و یک پاداش P به سیستم برمی‌گردد [۲۰]. همان‌طور که در نسخه بهبودیافته XCS از یک حافظه برای نگهداری بهترین دسته‌بندها استفاده می‌شود، در XCSF هم می‌توان از این حافظه برای بهبود کارایی مدل یادگیر استفاده کرد که آن را BCM-XCSF می‌نامیم.

الگوریتم یادگیری دارای چنین حافظه‌ای الگوریتم BCM-XCS نامیده می‌شود.

٢-٢-٣ الگوریتم XCSF

تقریب توابع یک تکنیک مهم است که در بسیاری از حوزه‌های مختلف از جمله ریاضیات عددی، مهندسی و علوم اعصاب استفاده می‌شود. سیستم دسته‌بندی XCSF می‌تواند توابع چندبعدی پیچیده را، با استفاده از توابع ساده‌تر تقریب بزند. برای تقریب تابع، XCS از دو جنبه اصلاح شد. اولین مورد تطبیق الگوریتم برای ورودی عدد صحیح به جای بردارهای ورودی باینری بود. دومین تغییر پیش‌بینی‌های بازده الگوریتم را، مستقیماً در خروجی در دسترس قرار می‌دهد و سیستم را به یک عمل محدود می‌کند [۱۹].

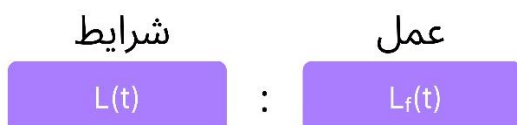
در شکل (۳) این مورد کاملاً مشخص است. برنامه حاصل XCSF نام خواهد داشت. درواقع، شرط دسته‌بند از رشته‌ای مانند $\{0, 1\}$ به ترکیبی از بازه‌ها مانند (li, ui) تغییر می‌کند. که در آن li حد پایین و ui حد بالای اعداد صحیح در بازه هستند. برای هر ورودی x مانند xi ، در صورتی که برای همه xi ها عبارت $li \leq xi \leq ui$ صدق کند دسته‌بند، آن را با بازه تطبیق می‌دهد.

خروجی الگوریتم XCSF یک تابع به شکل $f(x)$ است که طبق رابطه (۱۳) به دست می‌آید. در این رابطه، w بردار وزنی شامل $(w_0, w_1, \dots, w_n)$ است. N تعداد ورودی‌های الگوریتم است. x' بردار ورودی‌ها شامل $(x_0, x_1, \dots, x_n)$ است. بردار وزن w دارای $n + 1$ جزء در فضای حالت n بعدی است و یک وزن w_i برای هر بعد ورودی و وزن w_0 برای ورودی ثابت x_0 در نظر گرفته می‌شود.

$$f(x) = w \cdot x' \quad (13)$$

برای هر عمل ai در $[M]$ ، XCSF پیش‌بینی سیستم را محاسبه می‌کند. درواقع، بازدهی را که XCSF هنگام انجام عمل ai انتظار دارد، تخمین می‌زند. همانند XCS، در XCSF پیش‌بینی سیستم عمل a با میانگین وزنی برازش همه دسته‌بندهای منطبق که عمل a را مشخص می‌کند، محاسبه می‌شود. باین حال، برخلاف XCS، در XCSF پیش‌بینی دسته‌بند به‌عنوان تابعی از وضعیت فعلی st و

از XCSF و BCM-XCSF برای کمک به تصمیم‌گیری ایستگاه پایه جهت تعیین میزان بار پردازشی در لبه شبکه استفاده می‌شود. در مدل‌سازی بارها در پردازش لبه ورودی این سیستم (شرایط وضعیت شبکه و خروجی (عمل) آن میزان بار پردازشی در لبه شبکه است. شکل (۴) نمایی از شرایط دریافتی سیستم و عملی است که دسته‌بند تولید می‌کند.

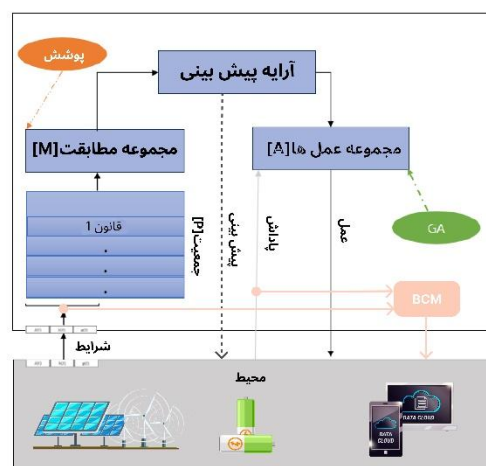


شکل (۴): نمونه‌ای از یک سطر دسته‌بند

شکل (۴) گواه بر این است که وضعیت محیط با یک پارامتر $L(t)$ قابل دریافت است. همچنین، انرژی تجدیدپذیر $g(t)$ و ازدحام شبکه $H(t)$ در هر بازه زمانی به عنوان وضعیت محیط داده می شود. وضعیت باتری $b(t)$ هم در لبه در سیستم معلوم است. زیرا در بازه زمانی قبل محاسبه شده است. در نهایت بر اساس این شرایط و با استفاده از الگوریتم ذکر شده میزان بار مناسب برای پردازش در لبه شبکه پیش بینی و یاد گرفته می شود. این میزان بار $L_f(t)$ به عنوان یک عمل به محیط اعمال می شود. در نهایت نتیجه این عمل به عنوان پاداش از جانب محیط به ما بازخورد داده می شود. این بازخورد همان هزینه ایست که به سیستم تحمیل شده و الگوریتم به دنبال کاهش این هزینه ها عمل خواهد کرد. هزینه ها توسط مدل سازی ارائه شده در بخش ۳-۱ محاسبه می شود.

با تطبیق این شرایط با وضعیت سیستم و قرار دادن سیستم دست‌بند یادگیر به‌عنوان عامل در الگوریتم یادگیر، عمل موردنظر تولید و به محیط اعمال می‌شود. پس از اعمال عمل، هزینه تحمیل‌شده به سیستم $c(t)$ تحت عنوان پاداش به عامل داده می‌شود. شکل (۳) این روند را به تصویر می‌کشد. همچنین در شکل (۳) حافظه BCM هم نمایش داده‌شده که برای استفاده از الگوریتم BCM-XCSF مورد استفاده قرار می‌گیرد.

در هر مرحله پس از پردازش بارها در ابر و لبه، هزینه تحمیل شده به سیستم را به عنوان بازده از محیط در نظر می گیریم که از فرمول (۱۶) محاسبه می شود:



شکل (۳): ساختار $XCSF$ و ارتباط آن با محیط

تفاوت الگوریتم XCS و الگوریتم XCSF

مهم‌ترین تفاوت XCS و XCSF تنها در خروجی است. بازخورد و ورودی‌های آن یکسان است. XCS مقدار ثابتی را به عنوان خروجی تولید می‌کند درحالی‌که در XCSF خروجی به شکل تابع $f(x)$ است. بخش وزنی این خروجی، w برداری است که در جریان الگوریتم یادگیری تقویتی به دست می‌آید. ورودی در محاسبات مه شامل میزان کل بار ورودی، ازدحام شبکه و میزان انرژی تجدید پذیر دریافت شده است. بنابراین بردار ورودی ما شامل این سه مورد است و در هر گره پردازشی مه مقدار بردار وزنی این تابع مشخص می‌شود.

۳-۳ فرموله سازی مسئله

در این قسمت، با استفاده از یادگیری تقویتی مسئله تخصیص بارها به ابر و لبه شبکه فرموله می‌شود. هدف این است که هزینه سیستم به حداقل برسد. در این یادگیری تقویتی یک عامل با استفاده از آزمون و خطا و تعامل با محیط و استفاده از بازخوردهای عمل‌ها مسئله را یاد می‌گیرد. در نتیجه از پاداش‌ها و تنبیه‌ها به عنوان معیاری برای شناسایی رفتار مثبت و منفی بهره برده می‌شود. عنصرهای اساسی یک مسئله یادگیری تقویتی از مورد های زیر تشکیل شده است:

۱- محیط: جهان فیزیکی عامل در آن عمل می‌کند. ۲- حالت: موقعیت فعلی عامل ۳- پاداش: بازخورد از محیط ۴- سیاست: روشی برای نگاشت حالت عامل به عمل

برای محاسبه میزان باری که در باید در لبه باید پردازش شود می‌توان استفاده نمود.

۴-۱ جزئیات پیاده‌سازی

روش پیشنهادی روی سیستمی با پردازنده ۵ هسته‌ای و فرکانس ۳۰/۲ GHz و حافظه ۸ گیگابایت با کتابخانه‌ای زبان C++ پیاده‌سازی شده است [۲۱]. در این شبیه‌سازی قسمت بازده الگوریتم XCSF فرمول‌های ارائه‌شده در بخش ۳-۱ محاسبه می‌شود و داده‌های مقاله مرجع که کل بار ورودی در هر لحظه هستند را به الگوریتم می‌دهیم. خروجی الگوریتم با توجه به بازده در هر لحظه، میزان بار قابل پردازش در لبه شبکه خواهد بود. در این بخش ابتدا مقادیر تنظیم‌شده برای متغیرهای سناریو پیشنهادی بیان‌شده و سپس نتایج حاصله ارزیابی می‌شود. این سناریو شامل ده هزار بازه زمانی بوده که هر بازه زمانی ۱۵ ثانیه است. تعداد بار ورودی در هر بازه زمانی $L(t)$ بین ۱۰ درخواست بر ثانیه تا ۱۳۰ درخواست بر ثانیه، به صورت تصادفی با تابع توزیع پواسون به ایستگاه پایه وارد می‌شود. $H(t)$ بین ۱۰ میلی‌ثانیه تا ۲۰ میلی‌ثانیه تغییر می‌کند. مقدار انرژی تجدید پذیر طبق [۱۲] از توزیع نرمال با پارامتر $N(520, 150)$ به سیستم وارد می‌شود. حداکثر ظرفیت باتری $B=2kWh$ در نظر گرفته شده که در $b(0)=0$ است. مصرف انرژی ثابت ایستگاه پایه $E_{sta}=300W$ است. حداکثر تعداد کارگزارها در لبه شبکه $S=10$ است که هر کدام از این کارگزارها در صورت فعال بودن $150W$ مصرف می‌کند. حداکثر نرخ پردازشی هر کارگزار ۲۰ درخواست بر ثانیه است. هزینه استهلاک هر واحد از باتری $\beta=0.01$ خواهد بود و ضریب هزینه هر واحد از منبع تغذیه پشتیبان $\varphi=0.15$ است. مقادیر پارامترها در الگوریتم XCSF به این صورت است که مقدار x_0 در این مسئله ۱ در نظر گرفته شده است. مقدار نرخ یادگیری ۰,۵ است.

۴-۲ مقایسه خروجی XCSF و XCS

همان‌طور که ذکر شد، الگوریتم XCSF تقریبی از وزن‌های تابع خطی توزیع بار به لبه شبکه به ما می‌دهد و ما می‌توانیم از این تابع برای تصمیم‌گیری در مورد ورودی‌ها در زمان‌های بعدی استفاده کنیم. بنابراین نیازی به اجرای الگوریتم برای تصمیم‌گیری برای بارهای کاری در اسلات‌های زمانی بعدی نخواهد بود. برای

$$\begin{aligned} & \text{if } (b(t) \leq \text{dop}(L(t))) \\ & c(t) = c_{\text{delay}}(H(t), L(t), 0, 0) + c_{\text{bak}}(L(t)) \\ & \text{else} \\ & c(t) = c_{\text{delay}}(H(t), L(t), Lf(t), S(t)) + c_{\text{battery}}(t) \end{aligned} \quad (16)$$

که در بخش اول اگر میزان باتری کمتر از انرژی مورد نیاز باشد $Lf(t)=S(t)=0$ خواهد شد و برای ارسال بارها به ابر از منبع تغذیه پشتیبان استفاده خواهد شد. در بخش دوم که باتری به اندازه کافی داریم بجای هزینه منبع تغذیه پشتیبان، هزینه استهلاک باتری اضافه خواهد شد.

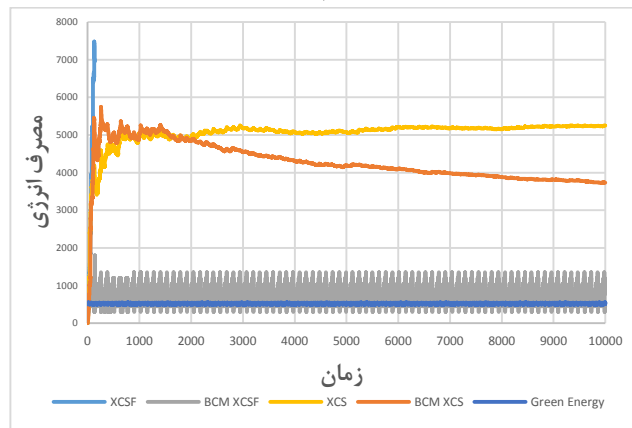
در نهایت الگوریتم XCSF طی یکبار اجرا وزن‌ها را مقداردهی می‌کند و یک تابع به فرم (۱۷) در اختیار ما قرار می‌دهد. پس از آن در زمان‌های بعدی که مقداری بار ورودی داریم، الگوریتم اجرا نخواهد شد و از تابع به دست آمده از الگوریتم XCSF برای تعیین میزان بار قابل پردازش در گره‌های مه استفاده خواهد شد. تا زمانی که درصد خطای کمی داشته باشیم، از این روند بهره‌وری می‌شود. این درصد خطا توسط میزان مصرف انرژی و تأخیر تعریف می‌شود. در هر مرحله که از این تابع به دست آمده استفاده می‌کنیم، میزان مصرف انرژی و تأخیر را باید زیر نظر بگیریم. در صورتی که الگوی مصرف انرژی تغییرات قابل توجهی را داشته باشد یا میزان تأخیر از مقدار متعارف بیشتر شود، مجدد الگوریتم اجرا می‌شود و تابع جدیدی برای میزان بار ورودی ارائه خواهد شد.

$$f(L(t)) = w_0 \cdot x_0 + w_1 \cdot L(t) \quad (17)$$

۴- پیاده‌سازی و ارزیابی

در این پژوهش هر دو الگوریتم پیشنهادی XCSF و BCM-XCSF را پیاده‌سازی می‌کنیم تا عملکرد آن را در لبه شبکه ارزیابی کنیم. حتی اگر خروجی‌های تابع‌های حاصل از این الگوریتم که همان مقدار بار قابل پردازش در لبه شبکه هستند، تا حدودی مشابه با خروجی‌های الگوریتم XCS و BCM-XCSF باشد ما نتایج مطلوب را گرفته‌ایم، زیرا سربار محاسباتی اجرای الگوریتم پس از مشخص شدن وزن‌های تابع با XCSF کاهش می‌یابد. پس از مشخص شدن ضرایب وزنی تابع توزیع بار در XCSF، از این تابع

است و رشد صعودی نخواهد داشت. پس می‌توان نتیجه گرفت که از نظر مصرف انرژی الگوریتم BCM XCSF عملکرد بهتری دارد.



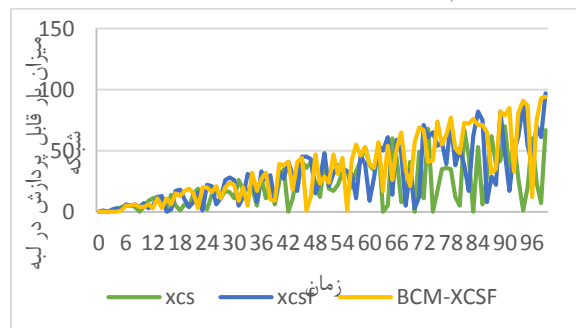
شکل (۶): میزان مصرف انرژی در ۱۰۰۰۰ بازه زمانی

نکته‌ای که در شکل (۶) به چشم می‌خورد، این است که مصرف انرژی مدل حافظه‌دار نسبت به مدل بدون حافظه در XCSF کاهش داشته و تقریباً انرژی تجدید پذیر مقداری انرژی مورد نیاز را در لبه شبکه تأمین می‌کند. از این موضوع می‌توان نتیجه گرفت که روش BCM-XCSF با ادامه یادگیری و جمع‌آوری بهترین دسته‌بندها در تلاش است، بارها به‌گونه‌ای توزیع شوند که انرژی مورد نیاز آن‌ها بیشتر از طریق انرژی تجدید پذیر تأمین شود تا مصرف باتری به حداقل برسد.

در ۱۰۰۰ بازه زمانی دو روش XCS و BCM-XCS روند تقریباً مشابهی را طی می‌کنند اما اگر در بازه زمانی ۱۰۰۰۰ آن‌ها را مقایسه کنیم، در بازه زمانی ۲۰۰۰ به بعد مصرف انرژی روش BCM-XCS نسبت به XCS پایه رو به کاهش می‌رود و به عدد ۴۰۰۰ در مصرف انرژی همگرا می‌شود، در حالی که در بازه ۱۰۰۰۰ روش XCS پایه به عدد ۵۰۰۰ در مصرف انرژی همگرا می‌شود.

روش BCM-XCSF حتی در ۱۰۰۰۰ بازه زمانی نیز عملکرد خود را حفظ می‌کند و مصرف انرژی آن از ۱۵۰۰ افزایش نمی‌یابد. لازم به ذکر است که در بازه زمانی ۱۰۰۰۰، مصرف انرژی روش XCSF مقدار مصرف انرژی به اندازه‌ای زیاد می‌شود که با نمودارهای دیگر قابل مقایسه نیست.

هر بار اجرای الگوریتم سرباری به هر گره در محیط مه اضافه می‌شود. در نتیجه در اسلات‌های زمانی پس از همگرایی سیستم به تابع توزیع بار، در صورتی که خروجی یعنی میزان بار قابل پردازش در گره‌های مه در الگوریتم XCSF با خروجی الگوریتم XCS مطابقت تقریبی داشته باشد، این سربار محاسباتی از سیستم حذف شده و XCSF بهتر از XCS عمل می‌کند. پس در این پژوهش، معیار بهبود XCSF به XCF، تطابق تقریبی خروجی‌های این دو الگوریتم است.



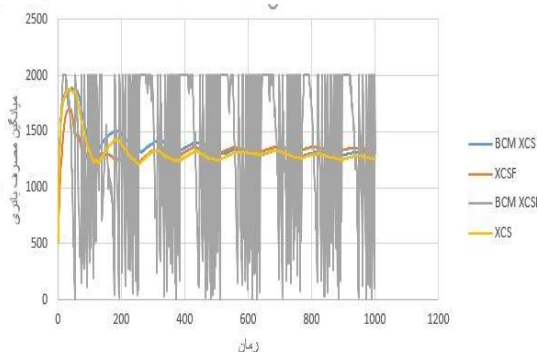
شکل (۵): میزان توزیع بار به ازای بار ورودی در ۱۰۰ بازه زمانی

در شکل (۵) می‌بینیم که نمودار خروجی‌های الگوریتم XCSF و خروجی‌های الگوریتم XCS باهم تطابق تقریبی دارند. به عبارات دیگر، می‌توان نتیجه گرفت که میانگین بار پردازشی در مه، در هر دو الگوریتم باهم تطابق دارد.

حتی پس از افزودن حافظه برای نگهداری بهترین دسته‌بندها در XCF نیز این تطابق تقریبی خروجی‌ها حفظ می‌شود. حال باید ببینیم که مقدار مصرف انرژی، تأخیر و مصرف باتری در هر الگوریتم چطور تغییر می‌کند. چراکه هدف ما ایجاد مصالحه بین سطح انرژی و تأخیر بوده است.

۴-۳ مقایسه مصرف انرژی

در شکل (۶) مصرف انرژی برای ۱۰۰۰ بازه زمانی نمودار رسم شده است. در ابتدا صعود هر سه الگوریتم XCS، BCM XCS و XCSF یکسان است. پس از بازه زمانی ۱۴۰ مصرف انرژی الگوریتم XCSF به صورت ناگهانی بسیار افزایش می‌یابد. اما همان‌طور که مشخص است مقدار میانگین مصرف انرژی در الگوریتم BCM XCSF در طول این بازه زمانی از ۲۰۰۰ کمتر



شکل (۸): میانگین مصرف باتری هر ۴ روش در ۱۰۰۰ بازه زمانی

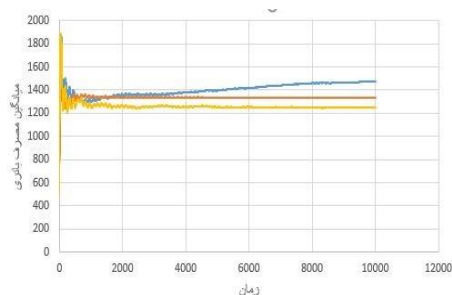
۴-۵ مقایسه تأخیر

در شکل (۹) نمودار میزان تأخیر هر الگوریتم تا بازه‌ی زمانی ۲۰۰ نمایش داده شده است. برای ملموس بودن و قابل مقایسه بودن نمودار هر کدام از روش‌ها، تسریع آن‌ها نسبت به الگوریتم XCS نرمال شده است. طبق تصویر می‌بینیم که تأخیر الگوریتم XCSF رو به کاهش است. تأخیر الگوریتم BCM-XCSF تا حدودی مشابه به تأخیر الگوریتم BCM-XCS است اما کمتر از آن است. در بازه‌های زمانی که مقدار بار ورودی برای پردازش کمتر است، عملکرد روش BCM-XCSF به عملکرد روش XCSF نزدیک تر است. هرچقدر این میزان بار ورودی یعنی $L(t)$ افزایش یابد، عملکرد BCM-XCSF به عملکرد BCM-XCS از لحاظ تأخیر نزدیک‌تر می‌شود. برای مثال، در ۱۰ بازه زمانی ابتدایی و همین‌طور بازه زمانی ۱۲۵ تا ۱۴۰ مقدار $L(t)$ زیر ۱۰ است. در این بازه زمانی، تأخیر XCSF و BCM-XCSF تا حدودی مشابه است و در برخی نقاط این دو نمودار به هم متصل می‌شوند و حدوداً تاخیرشان به ترتیب، ۴۰٪ و ۶۰٪ تأخیر XCS است.

۴-۶ مقایسه میزان باتری مصرف شده

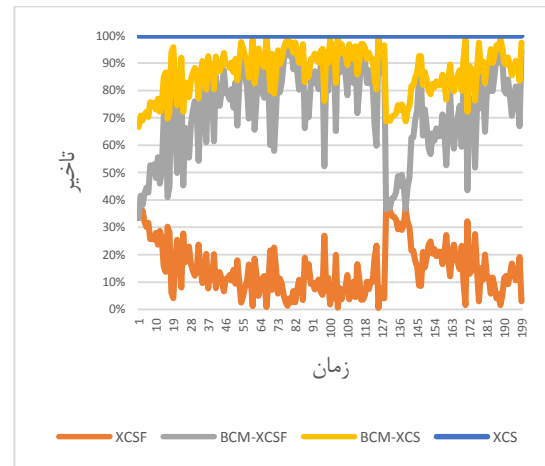
طبق نمودار به دست آمده در شکل (۶) در ابتدای بازه زمانی، باتری در هر دو الگوریتم در حال شارژ شدن است. در ابتدای بازه زمانی الگوریتم XCS مقدار شارژ بیشتری را در باتری نتیجه می‌دهد. پس از بازه زمانی ۲۵۰ مقدار شارژ در دو الگوریتم در حال همگرا شدن هستند. در الگوریتم XCSF بافاصله کمی از الگوریتم XCS شارژ بیشتری در باتری‌ها باقی می‌ماند. با در نظر گرفتن نمودار روش BCM-XCS، این بار سه روش XCS، XCSF و BCM-XCS را در ۱۰۰۰۰ بازه زمانی می‌بینیم. نتایج نشان می‌دهند که مقدار باتری BCM-XCS بیشتر از دو روش دیگر است. بنابراین، عملکرد روش XCSF در شارژ باتری از روش XCS پایه بهتر است.

در شکل (۷) می‌توان دید که BCM-XCSF به دلیل اینکه مصرف انرژی کمتری دارد و مقداری از این انرژی مصرفی توسط انرژی تجدید پذیر تأمین شده است در بیشتر اوقات میزان شارژ باتری بیشتری را نتیجه می‌دهد. در واقع، در این روش استفاده کمتری از باتری شده است و در بسیاری از بازه‌هایی زمانی باتری شارژ کامل یعنی ۲۰۰۰ را دارد.



شکل (۷): میانگین مصرف باتری در ۱۰۰۰۰ بازه زمانی

- applications," IEEE Communications Surveys & Tutorials, vol. 17, pp. 2347-2376, 2015.
- [4] C. -H. Hong and . B. Varghese, "Resource Management in Fog/Edge Computing: A Survey," arXiv preprint arXiv:1810.00305, 2018.
- [5] P. Zhang, . M. Zhou and . G. Fortino, "Security and trust issues in Fog computing: A survey," Future Generation Computer Systems, vol. 88, pp. 16-27, 2018.
- [6] W. Shi, J. Cao, Q. Zhang, Y. Li and . L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, pp. 637-646, 2016.
- [7] F. M. Talaat, . M. S. Saraya, . A. I. Saleh, H. A. Ali and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," Ambient Intell Human Comput, vol. 11, p. 4951-4966, 2020.
- [8] S. W. Wilson, "Classifiers that approximate functions," Natural Computing, vol. 1, p. 211-234, 2002.
- [9] J. Holland et al., "What Is a Learning Classifier System?," in Learning Classifier Systems, 2000.
- [10] B. Bartin, "Use of learning classifier systems in microscopic toll plaza simulation models," IET Intelligent Transport Systems, vol. 13, pp. 860-869, 2019.
- [11] M. R. Karlsen and S. Moschogiannis, "Evolution of control with learning classifier systems," Applied network science, vol. 3, p. 30, 2018.
- [12] J. Xu, . L. Chen and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," IEEE Transactions on Cognitive Communications and Networking, vol. 3, pp. 361-373, 2017.
- [13] D. Kanapram, . G. Lamanna and . M. Repetto, "Exploring the trade-off between performance and energy consumption in cloud infrastructures," in Second International Conference on Fog and Mobile Edge Computing (FMEC), 2017.
- [14] H. Wu, L. Chen, C. Shen, . W. Wen and . J. Xu, "Online geographical load balancing for energy-harvesting mobile edge computing," in 2018 IEEE International Conference on Communications (ICC), 2018.
- [15] J. Wan, B. Chen, S. Wang, M. Xia, D. Li and C. Li, "Fog Computing for Energy-aware Load Balancing and Scheduling in Smart Factory," IEEE Transactions on Industrial Informatics, 2018.
- [16] D. Zeng, L. Gu and H. Yao, "Towards energy efficient service composition in green energy powered Cyber-Physical Fog Systems," Future Generation Computer Systems, 2018.



شکل (۹): مقایسه درصد تأخیر الگوریتم‌ها در هر بازه زمانی

۵- نتیجه‌گیری

در این مقاله، از سیستم دسته‌بند یادگیر مبتنی بر XCS به نام XCSF برای تقریب تابع توزیع بار در محاسبات مه‌آلود استفاده شد. نتایج نشان داد که بار محاسباتی به‌صورت بهینه توسط این مدل یادگیری تقویتی بین مه و ابر توزیع می‌شود. مقایسه‌ها نشان می‌دهد که روش ارائه‌شده از نظر کاهش انرژی مصرفی و کاهش زمان پردازش از XCS پایه بهتر عمل می‌کند. در روش ارائه‌شده، در اکثر زمان‌ها بیشینه انرژی تجدید پذیر موجود برای محاسبات استفاده‌شده و انرژی باتری در بیشترین سطح شارژ ممکن باقی می‌ماند.

پایه‌سازی مدل‌های کوچک از دسته‌بند ارائه‌شده در این تحقیق قدم بعدی تحقیق خواهد بود. چنین مدل‌های کوچکی قابلیت اجرا در سیستم‌های لبه با منابع محدود را دارند.

References

- [1] J. Ni, K. Zhang, . X. Lin and . X. . S. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," IEEE Communications Surveys & Tutorials, vol. 20, pp. 601-628, 2017.
- [2] M. Chiang and . T. Zhang, "Fog and IoT: An overview of research opportunities," IEEE Internet of Things Journal, vol. 3, pp. 854-864, 2016.
- [3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, . M. Aledhari and . M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and



- [19] S. W. Wilson, "Classifiers that approximate functions," *Natural Computing*, vol. 1, p. 211–234, 2002.
- [20] P. Lanzi, "XCS with Computed Prediction for the Learning of Boolean Functions," in *IEEE Congress on Evolutionary Computation*, 2005.
- [21] fchauvel, "XCSF," [Online]. Available: https://github.com/fchauvel/XSCF.Optimization_of_Electrical_and_Electronic_Equipment (pp. 167–173).
- [17] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei and M. R. Khosravi, "Efficient resource management and workload allocation in fog–cloud computing paradigm in IoT using learning classifier systems," *Computer Communications*, vol. 153, pp. 217-228, 2020.
- [18] S. W. Wilson, "Get Real! XCS with Continuous-Valued Inputs," in *International Workshop on Learning Classifier Systems*, 2000.

An efficient mechanism to determine a function for load distribution in fog computing based on the use of learning classification systems

Mahdi Abbasi, Bahareh HamidiMoheb*

Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan, Iran

Article Information

Original Research Paper

Received:

2023 June 29

Accepted:

2024 March 12

Keywords:

Internet of things, fog computing processing, function estimation, learning classification systems

Corresponding Author*:

b.hamidimoheb@eng.basu.ac.ir

Abstract

With the rapid development of the Internet of Things, the fog computing model has been introduced as an attractive solution for processing data in IoT applications. In the fog environment, IoT applications are executed by intermediate computational nodes in the fog, as well as physical servers in cloud data centers. Therefore, issues related to resource management and energy management as challenging problems in fog computing need to be addressed. Recently, research has been conducted to create a balance between energy and cost in fog computing. In this study, while examining these approaches, an efficient method for approximating the load distribution function using two learning classification systems called XCSF and BCM-XCSF in fog processing nodes has been presented to optimize previous approaches and manage fog processing resources as much as possible. These two methods differ in having memory to store the best classifiers. Simulation results indicate that both XCS and BCM-XCS have appropriate load distribution. These two methods reduce computational overhead. Particularly, the BCM-XCSF method not only reduces delays by about 60%, but also improves energy consumption.



: 10.22034/ABMIR.2024.20271.1030

E-ISSN: [2821-2037](#) /© 2023. Published by Yazd University This is an open access article under the CC BY 4.0 License (<https://creativecommons.org/licenses/by/4.0/>).

