مجله محاسبات نرم، جلد ۱۱، شماره ۲ (پاییز و زمستان ۱۴۰۱)، صفحه ۷۲ـ۸۷
https://doi.org/10.22052/scj.2023.248643.1115

مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir

دانشگاه کاشان
University of Kashan

تامین پویای سرویس در محیط مه مبتنی بر اتوماتای یادگیر و الگوریتم ژنتیک چندهدفه✧

سید مهدی جامعی[1*]، استادیار

[1] گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، واحد شهر قدس، دانشگاه آزاد اسلامی، تهران، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت ۱۳ آذر ماه ۱۴۰۱
پذیرش ۲۱ فروردین ماه ۱۴۰۲

کلمات کلیدی:
اینترنت اشیا
محاسبات مه
محاسبات ابری
تامین پویای سرویس
الگوریتم ژنتیک
ماشین یادگیر

چکیده

امروزه تعداد برنامه‌های کاربردی که نیاز به زمان پاسخ‌دهی کمی دارند، روز به روز در حال افزایش است و بکارگیری محیط مه اخیرا مورد توجه زیادی قرار گرفته است. با توجه به پویایی استفاده از منابع در اکثر برنامه‌های اینترنت اشیا، نمی‌توان مکان ثابتی برای قرارگیری و اجرای سرویس‌ها در محیط مه در نظر گرفت و بنابراین باید سرویس‌ها در محیط مه به صورت پویا قرار داده شوند. مساله قرار دادن سرویس‌های مورد نیاز اینترنت اشیا در دستگاه‌های مه با محدودیت منابع، به عنوان یک مساله NP-hard شناخته می‌شود. در این مقاله، روشی پویا مبتنی بر الگوریتم ژنتیک چندهدفه با رتبه‌بندی نامغلوب جهت حل این مساله ارائه می‌گردد. در روش پیشنهادی، از اتوماتای یادگیر، جهت بهبود رفتار ژنتیکی و تنظیم پویای نرخ جهش و تقاطع استفاده می‌شود. روش پیشنهادی با استفاده از نرم‌افزار iFogsim شبیه‌سازی شده و نتایج شبیه‌سازی نشان می‌دهد روش پیشنهادی با در نظر گرفتن همزمان سه معیار تاخیر سرویس، هزینه و انرژی مصرفی، کارایی بهتری را نسبت به الگوریتم‌های مورد مقایسه دارد. از نظر هزینه، در مقایسه با دو روش CSA و LRFC به‌ترتیب ۱۱ و ۲۱ درصد کاهش داشته است. همچنین روش پیشنهادی از نظر میانگین تاخیر سرویس‌دهی نسبت به دو روش CSA و HAFA به‌ترتیب ۷ و ۱۵ درصد کاهش داشته است. از نظر انرژی مصرفی نیز روش پیشنهادی نسبت به روش‌های دیگر بهبود حداقل ۸ درصدی را نشان می‌دهد.

۱. مقدمه

پیشرفت‌های اخیر در زمینه‌های اینترنت اشیا[1] منجر به افزایش تعداد روزافزون برنامه‌های پیچیده شده است. این برنامه‌ها حساس به تاخیر بوده و نیازمند پردازش در زمان واقعی[2] هستند. اطمینان از کیفیت سرویس[3] برای چنین برنامه‌های حساس به تاخیری امری ضروری است [۱]. این برنامه‌ها نمی‌توانند تاخیر زیاد و غیرقابل پیش‌بینی ابر را زمانی که منابع ابری دور از محل تولید داده‌های برنامه مستقر می‌شوند، تحمل کنند. سرورهای متمرکز ابری قادر نیستند تا جریان داده‌ها را با سرعت و به صورت بلادرنگ رسیدگی کنند و این نوع فرآیند پردازش متمرکز باعث بروز تاخیر پاسخ‌دهی می‌گردد. به منظور غلبه بر محدودیت‌های رایانش ابر، مفهوم رایانش مه یا لبه پیشنهاد شد [۲]. رایانش مه یک معماری است که در آن تعداد زیادی دستگاه‌های ناهمگن، از طریق یک شبکه، باهم در ارتباط هستند

✧ نوع مقاله: پژوهشی
* نویسنده مسئول
پست(های) الکترونیک: sm.jameii@iau.ac.ir (جامعی)

[1] Internet of Things (IOT)
[2] Real-Time
[3] Quality of Service (QOS)

نحوه ارجاع به مقاله: جامعی، سید مهدی، «تامین پویای سرویس در محیط مه مبتنی بر اتوماتای یادگیر و الگوریتم ژنتیک چندهدفه»، مجله محاسبات نرم، جلد ۱۱، شماره ۲، ص۷۲ـ۸۷، پاییز و زمستان ۱۴۰۱.

و می‌توانند با تاخیر و ترافیک کم، پاسخگوی نیاز کاربران خود باشند. رایانش مه در مقایسه با محاسبات ابر دارای تاخیر کم، آگاهی مکانی، قابلیت تحرک و جابجایی، توزیع جغرافیایی گسترده، تعداد گره‌های زیاد، ناهمگن بودن گره‌ها و قابلیت اجرای نرم‌افزارهای کاربردی بلادرنگ می‌باشد. محاسبات مه مدلی است که در آن سیستم سعی می‌کند پردازش داده‌ها را از سرورهای ابری به دستگاه‌های نزدیک اینترنت اشیا انتقال دهد تا تاخیر پردازش را کاهش دهد. محاسبات مه می‌تواند به طور موثر، نیازهای برنامه‌های کاربردی بلادرنگ و حساس به تاخیر را برآورده و مشکلات پهنای باند شبکه را برطرف سازد [۳].

مکان‌های استقرار سرویس‌های اینترنت اشیا روی منابع موجود در محیط مه تاثیر مهمی بر زمان پاسخ کلی یک برنامه کاربردی می‌گذارند. به دلیل ماهیت پویای محیط مه و تغییرات دائمی رفتار برنامه‌های اینترنت اشیا در طول زمان، چالش بزرگ‌تر این است که چگونه این سرویس‌ها را به صورت پویا در بین منابع مه مستقر کنیم [۴]. بنابراین مساله تامین پویای سرویس به صورت بهینه در رایانش مه با هدف کاهش زمان سرویس‌دهی، کاهش مصرف انرژی و کاهش هزینه، انگیزه انجام این تحقیق است. این مساله به عنوان یک مساله NP-hard شناخته می‌شود که برای حل آن باید اهداف مختلفی به طور همزمان مورد توجه قرار گیرند. در واقع با یک مساله بهینه‌سازی چندهدفه مواجه هستیم. این مساله هنوز به عنوان یک مساله باز شناخته شده و محققان به دنبال یافتن الگوریتم‌های جدید بهینه برای حل این مساله هستند. همان‌طور که در جدول (۱) نیز ذکر شده است، در برخی از روش‌های موجود، به بعضی از اهداف از جمله تاخیر سرویس‌دهی، هزینه سرویس‌دهی و انرژی توجه شده ولی توجه همزمان به این اهداف در قالب یک مساله بهینه‌سازی می‌تواند منجر به روشی کارآمد جهت استقرار سرویس‌ها گردد. همچنین در روش‌های مبتنی بر الگوریتم‌های فراابتکاری برای استقرار سرویس‌ها در محیط مه، از تنظیم پویای عملگرهای ژنتیکی استفاده نشده است، لذا در این مقاله، روشی پویا مبتنی بر الگوریتم ژنتیک چندهدفه با رتبه‌بندی نامغلوب[1] جهت حل این مساله ارائه می‌گردد که از اتوماتای یادگیر، جهت بهبود رفتار ژنتیکی و تنظیم پویای نرخ جهش و نرخ تقاطع استفاده می‌کند. دلیل استفاده از الگوریتم ژنتیک چندهدفه با رتبه‌بندی نامغلوب کارایی بالای این الگوریتم در مواجهه با مساله‌های بهینه‌سازی چندهدفه و حل آنهاست که در سال‌های اخیر بسیار مورد توجه قرار گرفته است [۵]. در نسخه پایه این الگوریتم، مقدار ثابتی برای نرخ تقاطع و نرخ جهش در نظر گرفته می‌شد و نمی‌توان از قبل مشخص نمود کدام نرخ تقاطع یا جهش برای مساله مورد نظر مناسب است. در حالی که مقادیر نرخ تقاطع و نرخ جهش می‌توانند روی کارایی الگوریتم بهینه‌سازی تاثیر زیادی داشته باشند. هرچه مقدار نرخ تقاطع بزرگ‌تر باشد، الگوریتم راحت‌تر می‌تواند راه‌حل‌های جدید را تشخیص دهد و سرعت همگرایی بیشتر می‌گردد. در مقابل، مقدار خیلی کوچک برای این نرخ منجر به گیر افتادن فرآیند جستجو در بهینه محلی[2] می‌شود. عملگر جهش نیز برای حفظ تنوع راه‌حل‌ها در جمعیت و جلوگیری از همگرایی زودرس[3] و گریز از بهینه محلی مفید است. اگر مقدار این نرخ خیلی بزرگ باشد، جنبه تصادفی بودن جستجو خیلی زیاد می‌شود. بنابراین باید توازن بین نرخ جهش (مرور فضای جستجو[4]) و نرخ تقاطع (کشف بهترین راه‌حل[5]) به طور مناسب کنترل شود. لذا برای انجام این کار، از اتوماتای یادگیر در روش پیشنهادی استفاده شده است که به طور پویا مقادیر نرخ تقاطع و نرخ جهش را بدون نیاز به کنترل خارجی تنظیم می‌کند. لذا می‌توان نوآوری‌های این مقاله را به شکل زیر ذکر کرد:

- مدل‌سازی مساله تامین پویای سرویس در محیط مه به صورت چندهدفه،
- ارائه روشی مبتنی بر الگوریتم ژنتیک چندهدفه جهت حل مساله تامین پویای سرویس،
- بکارگیری اتوماتای یادگیر جهت تنظیم پویای پارامترهای الگوریتم ژنتیک چندهدفه،

[1] Non-dominated Sorting Genetic Algorithm (NSGA-II)

[2] Local Optima
[3] Premature Convergence
[4] Exploring the search pace
[5] Exploiting the best solution

- ارزیابی روش پیشنهادی جهت نشان دادن کارایی آن به لحاظ زمان سرویس‌دهی، هزینه و مصرف انرژی.

ادامه مقاله به صورت زیر سازماندهی شده است: در بخش دوم، کارهای گذشته مرتبط با مساله مرور خواهند شد. سپس در بخش سوم، مدل‌سازی و فرموله‌سازی مساله آورده شده است. روش پیشنهادی مبتنی بر الگوریتم NSGA-II جهت حل مساله در بخش چهارم پیشنهاد خواهد شد و پیچیدگی زمان آن مورد تحلیل قرار می‌گیرد. در بخش پنجم، روش پیشنهادی شبیه‌سازی شده و به تحلیل نتایج و ارزیابی آن می‌پردازیم و در نهایت در بخش ششم نتیجه‌گیری آورده شده است.

۲. کارهای مرتبط

در این بخش کارهای مرتبط با تامین و استقرار سرویس در محیط مورد بررسی قرار می‌گیرند. تعدادی از مقالات مرتبط، شبیه روش پیشنهادی این مقاله، مساله استقرارسرویس را به صورت یک مساله بهینه‌سازی چندهدفه در نظر گرفته و تلاش کرده‌اند با استفاده از روش‌های فراابتکاری و همچنین روش‌های غیرفراابتکاری این مساله را حل کنند. بنابراین در ادامه، کارهای مرتبط در دو زیربخش شامل روش‌های فراابتکاری و روش‌های غیرفراابتکاری تشریح می‌شوند.

۲,۱. روش‌های فراابتکاری

در مقاله [۶]، یک چارچوب محاسباتی مبتنی بر محیط ابر-مه برای مدیریت درخواست‌های سرویس و مکان‌یابی سرویس‌ها روی گره‌های مه ارائه شده است. در چارچوب پیشنهادی، مساله استقرار سرویس به عنوان یک مساله بهینه‌سازی چندهدفه مدل‌سازی شده است که ناهمگونی منابع و برنامه‌های کاربردی را بر اساس الزامات کیفیت سرویس در نظر گرفته است. برای حل این مساله، یک الگوریتم تکاملی بر اساس جستجوی فاخته ارائه شده و معیارهایی از قبیل بهره‌وری منابع مه، مصرف انرژی و تاخیر سرویس در آن در نظر گرفته شده است.

در مقاله [۷]، روشی برای استقرار برنامه‌های اینترنت اشیا در زیرساخت مه پیشنهاد شده است. روش پیشنهادی الزامات کیفیت سرویس اینترنت اشیا را برای تعیین یک طرح استقرار در نظر می‌گیرد و مبتنی بر الگوریتم فراابتکاری بهینه‌سازی نهنگ است. روش پیشنهادی از توان عملیاتی و مصرف انرژی به عنوان توابع هدف برای یافتن طرح استقرار خدمات اینترنت اشیا مطلوب استفاده کرد. نتایج شبیه‌سازی نشان می‌دهد که این روش، مصرف منابع و درصد پذیرش خدمات را افزایش داده و تاخیر سرویس و مصرف انرژی را در مقایسه با سایر روش‌های مبتنی بر فراابتکاری کاهش می‌دهد.

در مقاله [۸]، مساله قرار دادن سرویس اینترنت اشیا در محیط محاسبات مه را مد نظر قرار داده و یک الگوریتم ژنتیک موازی بهبود یافته به نام IPGA-SPP ارائه نموده است. از آنجایی که الگوریتم ژنتیک ممکن است در بهینه محلی به دام افتد، روش این مقاله، آن را به صورت موازی با یک حافظه مشترک همراه با چندین عملگر نخبه پیکربندی کرده است. این روش، توزیع منابع را برای متعادل‌سازی بار در نظر می‌گیرد و اجرای سرویس را برای کاهش تاخیر اولویت‌بندی می‌کند. همچنین، با حفظ مجموعه‌ای از راه‌حل‌های پارتو با ایجاد مصالحه بین تاخیر سرویس، هزینه خدمات، استفاده از منابع و زمان سرویس، مساله را به عنوان یک مساله چندهدفه حل می‌کند.

در مقاله [۹]، مساله استقرار سرویس اینترنت اشیا روی گره‌های مه برای به حداقل رساندن هزینه‌های سرویس و اطمینان از کیفیت خدمات به عنوان یک مساله NP-hard در نظر گرفته شده است. برای حل این مساله، الگوریتم ترکیبی فراابتکاری به نام MGAPSO با ترکیب الگوریتم ژنتیک و الگوریتم بهینه‌سازی گروهی ذرات مبتنی بر نخبه‌گرایی ارائه شده است. آزمایش‌هایی روی چارچوب محاسباتی مه دو سطحی انجام شده که نتایج آزمایشات نشان می‌دهد که الگوریتم ترکیبی پیشنهادی این مقاله، زمان و هزینه خدمات را به حداقل می‌رساند.

در مقاله [۱۰]، یک الگوریتم قرار دادن سرویس مه مبتنی بر ویژگی شبکه‌های پیچیده به نام FSPCN ارائه شده که ایده آن، گروه‌بندی گره‌های مه در جوامع متعادل قبل از قرار دادن سرویس، بر اساس ساختار شبکه و گره‌ها و ویژگی‌های پیوندها می‌باشد. همچنین یک معیار فاصله همسایگی تعریف شده است

که بر اساس تعداد همسایه‌های مشترک در بین جوامع محاسبه می‌شود تا جوامع را اولویت‌بندی کند. این کار، میانگین تعداد پرش‌ها از درخواست گره‌ها به خدمات درخواستی را بهبود می‌بخشد و تاخیر و ترافیک را در شبکه کاهش می‌دهد.

در مقاله [۱۱]، یک الگوریتم استقرار سرویس مبتنی بر الگوریتم ژنتیک در محیط‌های ابر–مه ارائه شده که تمرکز آن به حداقل رساندن تاخیر است. برای این کار، روشی مبتنی بر جریمه معرفی شده است که منجر به کاهش مصرف انرژی و کاهش هزینه شده است.

۲،۲. روش‌های غیرفراابتکاری

در مقاله [۱۲]، با در نظر گرفتن یک محیط مبتنی بر محاسبات ابر–مه، دو مدل بر اساس میانگین متحرک یکپارچه اتورگرسیو[1] و حافظه کوتاه‌مدت بلندمدت[2]، به منظور قرارگیری سرویس چندرسانه‌ای که از پیش‌بینی ترافیک داده آگاه است، پیشنهاد شده است. هدف اصلی، انتخاب حداقل تعداد گره‌ها با توجه به ظرفیت سخت‌افزاری آنها برای ارائه سرویس‌های چندرسانه‌ای به‌نحوی است که تاخیر برای سرویس‌دهی به تمام درخواست‌ها به حداقل برسد و هزینه سرویس‌ها کاهش یابد.

در مقاله [۱۳]، روشی توزیع‌شده، مقیاس‌پذیر و با سربار پایین، برای انتخاب گره‌های مه جهت میزبانی سرویس‌ها پیشنهاد شده است. در این روش، معیارهایی از قبیل هزینه‌های محاسباتی و ارتباطی، درصد کارهای اجرا شده، زمان پاسخ و سربار در نظر گرفته شده است.

در مقاله [۱۴]، یک مکانیزم مبتنی بر اتوماتای یادگیر برای تعیین تصمیم‌های ارائه سرویس برای استقرار یا انتشار برنامه‌های اینترنت اشیا در زیرساخت مه ناهمگن و پویا ارائه شده است. علاوه بر این، یک مدیر ارائه سرویس پویا مستقل (DSPM[3]) طراحی شده است که از یک حلقه کنترل خود مدیریتی برای ارائه برنامه‌های اینترنت اشیا در زیرساخت مه پیروی می‌کند. نتایج شبیه‌سازی به دست آمده با استفاده از ردیابی ترافیک مصنوعی و دنیای واقعی نشان می‌دهد که این الگوریتم تاخیر سرویس، هزینه و نقض تاخیر سرویس را در مقایسه با سایر الگوریتم‌ها کاهش می‌دهد.

در مقاله [۱۵]، روشی جهت زمان‌بندی برنامه‌های اینترنت اشیا آگاه از کیفیت سرویس مبتنی بر میکروسرویس برای قرارگیری در محیط‌های مه پیشنهاد شده است. هدف اصلی این روش، توجه به میزان بودجه و توان عملیاتی و در عین حال تمرکز بر استفاده از منابع مه محدود است.

در مقاله [۱۶]، روشی مبتنی بر الگوریتم مدل توسعه منبع باز[4] جهت استقرار سرویس‌های اینترنت اشیا روی گره‌های مه ارائه شده است. در این روش، معیارهای هزینه سرویس و مصرف انرژی به عنوان توابع هدف در نظر گرفته شده است. راه‌حل پیشنهادی مقاله، مصرف منابع و نسبت پذیرش سرویس‌ها را افزایش داده و تاخیر سرویس و مصرف انرژی را در مقایسه با سایر مکانیسم‌های مبتنی بر فراابتکاری کاهش می‌دهد.

در مقاله [۱۷]، یک رویکرد ابتکاری برای قرار دادن سرویس‌های اینترنت اشیا بر اساس اولویت شبکه و مصرف انرژی در محیط محاسباتی ابر و مه پیشنهاد شده است. نویسندگان این مقاله، برنامه‌های اینترنت اشیا را با حداکثر دو سرویس در نظر گرفتند و روی مه و ابر قرار دادند.

در مقاله [۱۸]، روشی جهت سرویس‌دهی به برنامه‌های کاربردی اینترنت اشیا ارائه شده که در آن، زمان سرویس‌دهی، ازدحام شبکه و مصرف انرژی به عنوان هدف در نظر گرفته شده‌اند. در این مقاله دو استراتژی تخصیص منبع به نام‌های اولین برازش[5] و برازش تصادفی[6] را برای حل مشکل قرار دادن سرویس‌ها در محیط مه–ابر پیشنهاد داده است.

نویسندگان مقاله [۱۹]، روشی هوشمند جهت زمانبندی کارها در محیط مه–ابر برای بهبود کیفیت سرویس از نظر تاخیر و مصرف انرژی ارائه نمودند. در این مقاله یک لایه هوشمند بین اینترنت اشیا و ابر برای پردازش داده‌ها با استفاده از سیاست‌های مبتنی بر یادگیری معرفی شده است.

---

[1] ARIMA
[2] LSTM
[3] Dynamic Service Provisioning Manager
[4] ODMA
[5] First-Fit
[6] Random Fit

در مقاله [۲۰]، روشی برای قرار دادن برنامه‌های کاربردی اینترنت اشیا در محیط مه ارائه شده است که از الگوریتم A3C به عنوان یک رویکرد جدید یادگیری تقویتی عمیق برای حل مساله استقرار سرویس استفاده کرده است. این روش برای قرار دادن سرویس‌های اینترنت اشیا، اهداف به حداقل رساندن هزینه و تأخیر را مد نظر قرار داده و همچنین محدودیت‌های مهلت و منابع را در نظر گرفته است. با توجه به این اهداف، A3C به دنبال به حداکثر رساندن پاداش تجمعی بلند مدت برای بهبود کیفیت خدمات است. نتایج شبیه‌سازی نشان می‌دهد که این روش هزینه و تاخیر را در مقایسه با روش‌هایی مانند DDQL و IMPALA بهبود می‌بخشد.

در مقاله [۲۱]، یک رویکرد تطبیقی برای قرار دادن داده‌های اینترنت اشیا با چندین نسخه در زیرساخت مه با هدف کاهش تاخیر پیشنهاد شده است. برای مقابله با تغییرات برنامه اینترنت اشیا، یک الگوریتم حریصانه به نام قرار دادن تکرارهای چندگانه داده تطبیقی برنامه‌های کاربردی اینترنت اشیا ایجاد شده است که هدف آن، به حداقل رساندن تاخیر نوشتن، خواندن و مهاجرت با استفاده از تعدادی تکرار داده است. کارایی این رویکرد در کاهش ذخیره‌سازی و تاخیر دسترسی با کپی‌های کمتر داده با استفاده از شبیه‌سازی نشان داده شده است.

برای جمع‌بندی، در جدول (۱) خلاصه‌ای از روش‌های بررسی شده، ذکر شده است.

جدول (۱): خلاصه کارهای مرتبط

مرجع	روش	شبیه‌ساز	معیارهای کارایی	مزایا	معایب
[۶]	الگوریتم تکاملی مبتنی بر جستجوی فاخته	Matlab	بهره‌وری منابع مه، مصرف انرژی، تاخیر سرویس‌دهی	ارائه یک میان‌افزار کنترلی برای مدیریت سرویس‌ها و توجه به ناهمگونی منابع و سرویس‌ها	عدم تحلیل سرعت همگرایی، عدم توجه به مقیاس‌پذیری
[۷]	الگوریتم فراابتکاری بهینه‌سازی نهنگ	iFogSim	تاخیر سرویس‌دهی، هزینه سرویس‌دهی	توجه به مصرف منابع و درصد پذیرش خدمات	در نظر گرفتن یک تابع شایستگی به صورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب
[۸]	الگوریتم ژنتیک موازی بهبود یافته	iFogSim	تاخیر، هزینه منابع	توجه به نخبه‌گرایی در الگوریتم و پیشگیری از بهینه محلی	در نظر گرفتن یک تابع شایستگی به صورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب
[۹]	ترکیب الگوریتم ژنتیک و الگوریتم بهینه‌سازی گروهی ذرات	Python	هزینه سرویس‌دهی، زمان سرویس‌دهی	توجه به الزامات کیفیت سرویس، توجه به مقیاس‌پذیری	در نظر گرفتن یک تابع شایستگی به صورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب
[۱۰]	گروه‌بندی گره‌های مه در جوامع متعادل بر اساس ساختار شبکه و وضعیت گره‌ها و پیوندها	YAFS simulator	تاخیر سرویس‌دهی، مقیاس پذیری	توجه به توازن بار در دسته‌بندی گره‌های مه، اولویت‌بندی جوامع گره‌ها، کاهش ترافیک شبکه	عدم توجه به مصرف انرژی، عدم بهینه-سازی جوامع با تغییر محیط
[۱۱]	الگوریتم ژنتیک	iFogSim	مصرف انرژی، هزینه سرویس‌دهی	بهبود بکارگیری منابع شبکه	در نظر گرفتن یک تابع شایستگی به صورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب

ادامه جدول (۱): خلاصه کارهای مرتبط

مرجع	روش	شبیه‌ساز	معیارهای کارایی	مزایا	معایب
[۱۲]	مدل ARIMA و شبکه LSTM	iFogSim	هزینه سرویس‌دهی، تاخیر سرویس‌دهی	انتخاب گره‌های مه نزدیک‌تر به کاربر، کاهش درخواست‌های ارسالی	عدم تحلیل انرژی مصرفی، عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب
[۱۳]	معماری سلسله مراتبی و خوشه‌بندی شده برای گره‌های مه	PFogSim (extended from CloudSim)	هزینه‌های محاسباتی، هزینه‌های ارتباطی، زمان پاسخ و سربار	توجه به سربار، توجه به مقیاس‌پذیری	عدم توجه به معیار مصرف انرژی، عدم توجه به توازن بار
[۱۴]	مدیر ارائه سرویس پویا مستقل مبتنی بر اتوماتای یادگیر	iFogsim	تاخیر سرویس، هزینه، نقض تاخیر سرویس	توجه به پویایی درخواست سرویس-ها و بکارگیری اتوماتای یادگیر برای مدیریت پویایی	عدم توجه به معیار مصرف انرژی، عدم توجه به بحث جبهه پارتو و جواب‌های نامغلوب
[۱۵]	میکروسرویس‌ها به صورت گراف جهت‌دار بدون‌دور	iFogsim	هزینه، توان عملیاتی	توجه به کیفیت سرویس، مقیاس‌پذیری	عدم توجه به معیار مصرف انرژی، عدم توجه به پویایی سرویس‌ها
[۱۶]	الگوریتم مدل توسعه منبع باز (ODMA)	iFogsim	تاخیر سرویس‌دهی، مصرف انرژی	توجه به نسبت پذیرش سرویس‌ها	عدم تحلیل پیچیدگی زمانی، عدم توجه به مقیاس‌پذیری
[۱۷]	دسته‌بندی سرویس‌ها در دو دسته عادی و حیاتی	Java	مصرف انرژی، تاخیر شبکه	توجه به اولویت شبکه	عدم توجه به معیار هزینه، عدم توجه به چندین سرویس وابسته
[۱۸]	روش اولین برازش و برازش تصادفی جهت اختصاص منابع	Omnetpp	زمان سرویس‌دهی، ازدحام شبکه، مصرف انرژی	توجه به پهنای باند شبکه، توجه به احتمال قطعی سرویس‌ها	عدم توجه به معیار هزینه، عدم تحلیل پیچیدگی زمانی
[۱۹]	لایه هوشمند بین اینترنت اشیا و ابر برای پردازش داده‌ها	CloudSim, iFogsim	تاخیر، مصرف انرژی	زمانبند تطبیقی و یادگیرنده برای سرویس‌های حساس به تاخیر	عدم توجه به معیار هزینه، عدم تحلیل پیچیدگی زمانی
[۲۰]	روش جدید یادگیری تقویتی (A3C)	OpenAI Gym	هزینه، تاخیر	توجه به مهلت سرویس و محدودیت منابع	عدم توجه به معیار انرژی، زمان پاسخ بالا برای بعضی از سرویس‌ها
[۲۱]	قرار دادن تکرارهای چندگانه داده تطبیقی برنامه‌های کاربردی اینترنت اشیا	iFogSim	تاخیر دسترسی به داده‌ها	کاهش تاخیر خواندن و نوشتن داده‌ها، کاهش تعداد مهاجرت‌ها	عدم توجه به معیار انرژی، عدم توجه به معیار هزینه

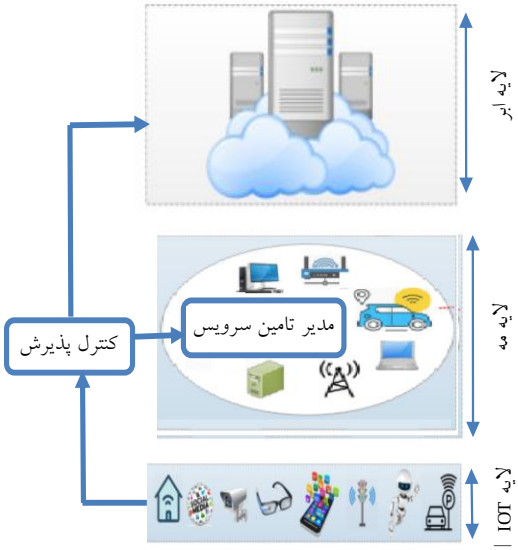
۳. مدل‌سازی و فرموله‌سازی مساله

در روش پیشنهادی، مطابق مرجع [۲۲] یک معماری سه لایه را برای اشیا، مه و ابر در نظر می‌گیریم. در پایین‌ترین لایه که لایه اینترنت اشیا نام دارد، تمامی حسگرها و اشیا هوشمند نظیر تبلت، موبایل و ... قرار دارند. در لایه مه که لایه میانی می‌باشد، دستگاه‌هایی مانند روتر و دروازه‌های هوشمند در محیط مه، به عنوان گره‌های مه برای میزبانی و سرویس‌دهی به برنامه‌های کاربردی اینترنت اشیا در نظر گرفته می‌شوند. اطلاعات مختلف توسط حسگرهای لایه پایینی جمع‌آوری و به لایه مه فرستاده می‌شود. در لایه مه، درخواست‌های دریافت شده از لایه پایین‌تر به صورت هوشمند بررسی و درخواست‌هایی که به صورت

بلادرنگ هستند و به پردازش، محاسبه و ذخیره‌سازی کمی نیاز داشته باشند، مورد پردازش قرار می‌گیرند. درخواست‌های دیگر به لایه بالاتر که همان لایه ابر است، ارسال می‌گردند. این لایه شامل سرورهای فیزیکی زیادی می‌باشد. دقت داشته باشید که تمرکز اصلی این مقاله بر روی چگونگی تامین سرویس در لایه مه می‌باشد. فرض می‌شود از بین گره‌های مه، گره‌ای که قابلیت-های بیشتری به لحاظ پردازش و حافظه دارد، به عنوان مدیر تامین سرویس انتخاب می‌گردد و این انتخاب در بازه‌های زمانی قابل انجام است و بعد از آن ممکن است گره دیگری به عنوان مدیر تامین سرویس انتخاب گردد. فرآیند تصمیم‌گیری در خصوص اختصاص سرویس‌ها به گره‌های مه و بازپس‌گیری آنها توسط مدیر تامین سرویس انجام خواهد شد. یک ماشین نیز وظیفه کنترل پذیرش را بر عهده خواهد داشت. درخواست سرویس‌های اینترنت اشیا از لایه پایینی به ماشین کنترل پذیرش داده می‌شود. این ماشین تشخیص می‌دهد که درخواست در خود مه پردازش شود یا به ابر منتقل گردد. اگر مهلت اجرای سرویس‌های اینترنت اشیا کمتر از یک حد آستانه باشد، در لایه مه اجرا می‌شود و در غیر اینصورت آن را به محیط ابری ارسال می‌کند. در خصوص درخواست‌هایی که باید در محیط مه اجرا شود، این درخواست‌ها وارد مدیر تامین سرویس می‌شود و باید تشخیص دهیم برای این درخواست رسیده چه منابعی لازم است و کدام گره یا گره‌های مه برای رسیدگی به این درخواست باید انتخاب شود. شکل (۱) شمای کلی این معماری سه لایه را نشان می‌دهد.

مساله بهینه‌سازی چندهدفه مورد نظر به صورت زیر فرموله می‌شود. برای هر سرویس i متعلق به یک برنامه اینترنت اشیا، ویژگی‌های مورد نیاز از قبیل مهلت زمانی $(Deadline^i)$، میزان پردازش مورد نیاز (Req_p^i) بر حسب میلیون دستورالعمل (MI) در هر درخواست، میزان حافظه مورد نیاز (Req_M^i)، میزان فضای ذخیره‌سازی مورد نیاز (Req_S^i) در نظر گرفته می‌شوند. در واقع ویژگی‌های مورد نیاز سرویس i را به صورت مجموعه زیر در نظر می‌گیریم:

$$Service^i = \{Deadline^i, Req_p^i, Req_M^i, Req_S^i\} \quad (۱)$$



شکل (۱): معماری سه لایه ابر- مه- اینترنت اشیا

هر گره مه j دارای منابع مشخصی از قبیل قدرت پردازش $(Proc^j)$، میزان حافظه اصلی (RAM^j) و فضای ذخیره‌سازی $(Storage^j)$ است و در نتیجه گره مه j به صورت رابطه (۲) مشخص می‌شود.

$$Node^j = \{Proc^j, RAM^j, Storage^j\} \quad (۲)$$

اهداف مساله بهینه‌سازی چندهدفه مورد بررسی به صورت زیر تعریف می‌گردند:

هدف ۱: کمینه کردن میانگین هزینه سرویس‌دهی که به صورت $Minimize\left(F_1 = Average(Cost^{service})\right)$ بیان می‌شود و در آن مجموع هزینه سرویس‌دهی به درخواست‌های ارسالی از طرف اینترنت اشیا با $Cost^{service}$ نشان داده می‌شود. این هزینه شامل هزینه پردازش $(Cost^{process})$، هزینه حافظه اصلی $(Cost^{RAM})$ و هزینه فضای ذخیره‌سازی $(Cost^{storage})$ است و به صورت رابطه (۳) تعریف می‌شود.

$$Cost^{service} = Cost^{process} + Cost^{RAM} + Cost^{storage} \quad (۳)$$

هزینه‌های ذکر شده فوق به‌ترتیب به‌صورت روابط (۴) تا (۶) محاسبه می‌شوند.

$$Cost^{process} = \sum_{j=1}^{m}\sum_{i=1}^{n} cost_j^{process}\, Req_p^i \quad (۴)$$

$$Cost^{RAM} = \sum_{j=1}^{m}\sum_{i=1}^{n} cost_j^{RAM}\ Req_M^i \qquad (۵)$$

$$Cost^{storage} = \sum_{j=1}^{m}\sum_{i=1}^{n} cost_j^{Storage}\ Req_S^i \qquad (۶)$$

در این روابط Req_p^i، Req_M^i و Req_S^i به ترتیب میزان پردازش مورد نیاز (برحسب میلیون دستورالعمل)، میزان حافظه مورد نیاز (برحسب بایت) و میزان فضای ذخیره‌سازی مورد نیاز (برحسب بایت) برای هر سرویس درخواستی i است و $cost_j^{process}$، $cost_j^{RAM}$ و $cost_j^{Storage}$ به ترتیب هزینه پردازش (برای هر میلیون دستورالعمل)، هزینه حافظه اصلی (برای هر بایت در ثانیه) و هزینه فضای ذخیره سازی (برای هر بایت در ثانیه) در گره مه j است. همچنین m برابر تعداد گره‌های مه و n حداکثر تعداد سرویس‌های مورد درخواست است. درنهایت، میانگین هزینه سرویس‌دهی به صورت رابطه (۷) محاسبه می‌گردد.

$$Average(Cost^{service}) = \frac{Cost^{service}}{n} \qquad (۷)$$

در این رابطه n تعداد کل سرویس‌های مورد درخواست از طرف اینترنت اشیا است.

هدف ۲: کمینه کردن تاخیر سرویس‌دهی که به صورت رابطه $Minimize(F_2 = Service\_Delay)$ تعریف می‌شود و در آن مدت زمان بین ارسال یک درخواست سرویس توسط یک دستگاه اینترنت اشیا و دریافت یک پاسخ برای آن تاخیر سرویس نامیده می‌شود. متوسط تاخیر سرویس برای سرویس i به صورت رابطه (۸) محاسبه می‌شود.

$$Service\_Delay^i = 2D_{IOT\ to\ Fog} + W_{i,j} + \frac{L_i^{req} + L_i^{resp}}{R_{IoT\ to\ Fog}} \qquad (۸)$$

در این رابطه $D_{IOT\ to\ Fog}$ میانگین تاخیر انتشار بین دستگاه‌های اینترنت اشیا و یک گره مه، $W_{i,j}$ زمان انتظار سرویس درخواست شده i در گره مه j (شامل تاخیر مربوط به انتظار در صف گره مه و تاخیر پردازش سرویس در آن گره مه)، $R_{IoT\ to\ Fog}$ میانگین نرخ انتقال بین یک دستگاه اینترنت اشیا و یک گره در لایه مه، L_i^{req} میانگین اندازه درخواست ورودی برای سرویس i (برحسب بایت) و L_i^{resp} میانگین اندازه پاسخ‌های تولید شده برای سرویس i (برحسب بایت) است.

هدف۳: کمینه‌سازی مجموع انرژی مصرفی که به صورت رابطه $Minimize(F_3 = Total\_energy\_consumption)$ تعریف می‌شود. در این مقاله، از مدل انرژی مصرفی ارائه شده در مرجع [۲۳] استفاده شده است که معادل مجموع انرژی مصرفی در گره‌های مه است و طبق رابطه (۹) محاسبه می‌شود.

$$Total\_energy\_consumption = \sum_{j=1}^{n} E_{f_j}^{energy} \qquad (۹)$$

در این رابطه، $E_{f_j}^{energy}$ انرژی مصرفی در گره مه j ام است و به صورت مجموع انرژی مصرفی برای ارسال داده بین گره‌های مه و انرژی مصرفی برای پردازش سرویس‌ها طبق روابط (۱۰) تا (۱۲) محاسبه می‌شود.

$$E_{f_j}^{energy} = \left(E_{trans}^{energy} + E_{proc}^{energy}\right) \qquad (۱۰)$$

$$E_{trans}^{energy} = \int (P_f^{idle} + (\frac{S_i^{size}}{\beta_f} P_{trans}))\ (t)\ d(t) \qquad (۱۱)$$

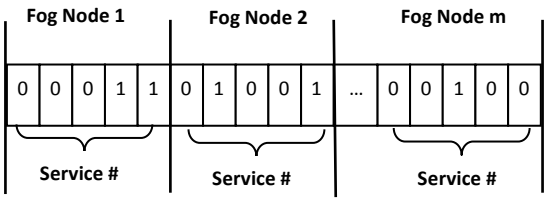
$$E_{proc}^{energy} = \int (P_f^{idle} + (\frac{(S_i^{size}}{Y_f} + P_{proc}))\ (t)\ d(t) \qquad (۱۲)$$

در این روابط، P_f^{idle} انرژی مصرفی گره مه در حالت بیکار، P_{trans} حداکثر توان مصرفی طی ارسال داده، S_i^{size} اندازه سرویس درخواستی، P_{proc} حداکثر توان مصرفی طی پردازش داده، β_f و Y_f به ترتیب پهنای باند بین گره‌ها و توان پردازشی گره مه است.

۴. روش پیشنهادی

در روش پیشنهادی، در ابتدای هر بازه زمانی، درخواست‌هایی که از طرف اینترنت اشیا به ماشین کنترل پذیرش ارسال شده‌اند و باید در محیط مه اجرا شوند بررسی می‌گردند. همچنین آخرین وضعیت گره‌های مه از نظر ظرفیت پردازش، حافظه اصلی و فضای ذخیره‌سازی مورد بررسی قرار می‌گیرند. سپس مدیر تامین سرویس، طبق الگوریتمی بر پایه NSGA-II و اتوماتای یادگیر که در این بخش تشریح خواهد شد، در بازه‌های زمانی مشخص این عملیات تخصیص را انجام خواهد داد. نحوه

کدگذاری و چگونگی نمایش راه‌حل‌ها در روش پیشنهادی به صورت شکل (۲) است.



شکل (۲): مثالی از نمایش یک راه حل در روش پیشنهادی

هر راه‌حل به صورت تخصیص تعدادی سرویس به ماشین‌های مه است. در اینجا فرض می‌شود شماره هر سرویس درخواستی به صورت رشته باینری پنج بیتی قابل نمایش است و لذا هر ژن از کروموزوم دارای پنج بیت است. هر عدد باینری پنج بیتی متناظر با شماره یک سرویس درخواستی است. به عنوان مثال در رشته پنج بیتی اول در شکل (۲)، رشته باینری ۰۰۰۱۱ نوشته شده و متناظر با سرویس شماره ۳ است و با توجه به اینکه این سرویس در اولین ژن کروموزوم نوشته شده است، به این معنی است که روی گره مه شماره یک اجرا می‌گردد. به همین ترتیب سرویس شماره ۹ (۰۱۰۰۱) روی گره مه شماره ۲ اجرا می‌شود و درنهایت سرویس شماره ۴ (۰۰۱۰۰) روی گره مه شماره m اجرا می‌گردد.

در ابتدا جمعیتی اولیه از راه‌حل‌ها ایجاد شده و سپس هر راه‌حل در جمعیت اولیه بر اساس سه تابع هدف F_1، F_2 و F_3 که در بخش قبل معرفی شدند، ارزیابی می‌گردد. در ادامه جمعیت فرزندان از روی جمعیت فعلی (جمعیت والد) و با استفاده از عملگرهای ژنتیکی جهش و تقاطع ایجاد می‌گردد. پس از ایجاد جمعیت فرزندان، این جمعیت با جمعیت والد ترکیب می‌شود. الگوریتم مرتب‌سازی سریع نامغلوب NSGA-II طبق الگوریتم ذکر شده در مرجع [۲۴] بر روی این جمعیت ترکیبی اعمال خواهد شد تا راه‌حل‌ها بر اساس رتبه نامغلوبی در دسته‌های مختلف fr_i $(i = 1, 2, \dots)$ قرار گیرند. برای انتخاب جمعیت جدید از روی جمعیت ترکیبی، ابتدا از راه‌حل‌های دسته‌ای با کمترین رتبه (fr_1) شروع می‌کنیم و سپس راه‌حل‌های دسته دوم (fr_2) و به همین ترتیب ادامه می‌دهیم تا اندازه جمعیت جدید به اندازه M برسد. روال ذکر شده فوق برای ایجاد جمعیت فرزندان و جمعیت ترکیبی و انتخاب جمعیت جدید تا جایی ادامه می‌یابد که به حداکثر تعداد نسل‌ها برسیم. شبه کد الگوریتم پیشنهادی در الگوریتم (۱) نشان داده شده است.

الگوریتم (۱): شبه کد الگوریتم تامین سرویس پیشنهادی

Input: Network parameters (number of requested services, number of fog nodes, Number of services)
Output: Non-dominated solutions (Assigning the services to the fog nodes).
Step 1: Choose population size M, Initial crossover rate (P_c), Initial mutation rate (P_m), Maximum number of generations (gen_{max}).
Set the generation count t = 0;
Step 2: Generate an initial population D_t as follow:
for i=1,…, M do
Step 2.1: Assign each requested services to each fog nodes randomly
Step 3: Evaluate each solution in D_t using objective functions.
Step 4: Receive the crossover rate (P_c) and mutation rate (P_m) for current generation from Learning Automata.
Step 5: Create offspring population E_t from D_t as follow:
Apply the crowded tournament selection;
Apply crossover operator;
Calculate the average generated ranks value of crossover $\overline{Gr}_{cross}$;
Apply mutation operator;
Calculate the average generated ranks value of mutation $\overline{Gr}_{mut}$;
Step 6: Determine reward or penalty response and send it to the LA module.
Step 7: Set $P_t = D_t \cup E_t$;
Step 8: Do fast non-dominated sorting on P_t, resulting non-dominated fronts fr_1, $fr_{2,\dots}$;
Step 9: Set D_{t+1}=Ø; i=1;
while $|D_{t+1}|+|fr_i|<M$ do ($D_{t+1} = D_{t+1} \cup fr_i$, i=i+1);
if $|D_{t+1}| <M$ then
Add the first M-$|D_{t+1}|$ solution from fr_i to D_{t+1};
Step 10: if $t<gen_{max}$ then set t=t+1 and go to step 3.

در الگوریتم NSGA-II پایه، نرخ تقاطع (p_c) و نرخ جهش (p_m) مقدار ثابتی دارند و نمی‌توان از قبل مشخص نمود کدام نرخ تقاطع یا جهش برای مساله مورد نظر مناسب است. مقادیر p_c و p_m می‌توانند بر روی کارایی الگوریتم بهینه‌سازی تاثیر زیادی

داشته باشند. هرچه مقدار p_c بزرگ‌تر باشد، الگوریتم راحت‌تر می‌تواند راه‌حل‌های جدید را تشخیص دهد و سرعت همگرایی بیشتر می‌گردد. در مقابل، مقدار خیلی کوچک برای p_c، منجر به گیر افتادن فرآیند جستجو در بهینه محلی می‌شود. عملگر جهش برای حفظ تنوع راه‌حل‌ها در جمعیت و جلوگیری از همگرایی زودرس و گریز از بهینه محلی مفید است. اگر مقدار p_m خیلی بزرگ باشد، جنبه تصادفی بودن جستجو خیلی زیاد می‌شود. بنابراین باید توازن بین نرخ جهش (مرور فضای جستجو) و نرخ تقاطع (کشف بهترین راه‌حل) کنترل شود. جهت انجام این کار، از اتوماتای یادگیر در روش پیشنهادی استفاده شده است که به طور پویا مقادیر p_c و p_m را بدون نیاز به کنترل خارجی تنظیم می‌کند.

ماشین یادگیر دارای دو عمل است که وظیفه به‌روزرسانی مقادیر p_c و p_m را با توجه به بازخورد به دست آمده از محیط خود به عهده دارند. عمل اول، مقادیر p_c و p_m را با افزایش مقدار α_1 به p_c و کاهش مقدار α_2 از p_m به‌روز می‌کند (روابط (۱۳) و (۱۴)).

$$p_c = p_c + \alpha_1 \quad (۱۳)$$

$$p_m = p_m - \alpha_2 \quad (۱۴)$$

عمل دوم، مقادیر p_c و p_m را با کاهش مقدار α_1 از p_c و افزایش مقدار α_2 به p_m به‌روز می‌کند (روابط (۱۵) و (۱۶)).

$$p_c = p_c - \alpha_1 \quad (۱۵)$$

$$p_m = p_m + \alpha_2 \quad (۱۶)$$

در ابتدا مقادیر اولیه p_c و p_m به ترتیب برابر ۰/۵ و ۰/۰۱ تنظیم می‌گردد و احتمال انتخاب هر یک از این دو عمل، مقادیر مساوی و برابر با ۰/۵ خواهد داشت. برای جلوگیری از تنظیم مقادیر نامعتبر برای p_c و p_m، برای هر یک از مقادیر p_c و p_m کران بالا و پایین تعریف می‌شود و این ماژول باید p_c و p_m را طوری تنظیم نماید که از محدوده مشخص شده، تجاوز نکنند. محدوده مجاز p_m را به صورت $[p_{mLB}, p_{mUB}]$ و محدوده مجاز p_c را به صورت $[p_{cLB}, p_{cUB}]$ نمایش می‌دهیم. در انتهای هر نسل، اتوماتای یادگیر با توجه مقدار احتمال انتخاب عمل‌ها، یکی از عمل‌های خود را انتخاب می‌کند و بر اساس عمل انتخاب شده، مقادیر p_c و p_m را تغییر داده و به مدیر تامین منابع ارسال می‌کند. مقادیر اولیه α_1 و α_2 وکران‌های بالا و پایین با توجه به ماهیت دو عملگر جهش و تقاطع و به‌صورت تجربی تعیین شده است. توجه به بازه انتخابی نرخ جهش و نرخ تقاطع در مقالات معتبر چاپ شده نظیر [۲۵] و [۲۶] به این انتخاب کمک شایانی نموده است. الگوریتم تخصیص سرویس بر اساس مقادیر جدید p_c و p_m اجرا شده و بازخورد مربوطه (پاداش یا جریمه) را برای عمل انتخاب شده توسط اتوماتا به صورت زیر مشخص می‌کند.

فرض کنید $sr_{offspring}$ مجموع رتبه‌های دو راه‌حل فرزند ایجاد شده بعد از اعمال عملگر تقاطع و sr_{parent} نیز مجموع رتبه‌های راه‌حل‌های والد، قبل از اعمال عملگر تقاطع باشند. ابتدا رتبه ایجاد شده توسط عملگر تقاطع را به صورت رابطه (۱۷) تعریف می‌کنیم.

$$Gr_{cross} = sr_{offspring} - sr_{parent} \quad (۱۷)$$

سپس برای یک نسل که در آن تعداد n_{cross} عمل تقاطع انجام شده است، میانگین رتبه‌های تولید شده توسط عملگر تقاطع را به صورت رابطه (۱۸) محاسبه می‌کنیم.

$$\overline{Gr_{cross}} = \frac{\sum Gr_{cross}}{n_{cross}} \quad (۱۸)$$

این پارامتر نشان‌دهنده کارایی عملگر تقاطع در نسل فعلی است. به طور مشابه، رتبه حاصل از عملگر جهش را به صورت رابطه (۱۹) تعریف می‌کنیم که در آن r_{new}، رتبه راه‌حل تولید شده بعد از اعمال عملگر جهش و r_{old} رتبه راه‌حل والد قبل از اعمال عملگر جهش است.

$$Gr_{mut} = r_{new} - r_{old} \quad (۱۹)$$

برای یک نسل که در آن تعداد n_{mut} عمل جهش انجام شده است، میانگین رتبه‌های تولید شده توسط عملگر جهش را به صورت رابطه (۲۰) محاسبه می‌کنیم.

$$\overline{Gr_{mut}} = \frac{\sum Gr_{mut}}{n_{mut}} \quad (۲۰)$$

این پارامتر نشان‌دهنده کارایی عملگر جهش در نسل فعلی است.

پاداش و جریمه طبق مرجع [۲۷] به ترتیب توسط روابط (۲۱) و (۲۲) صورت می‌گیرد.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \qquad \forall j, j \neq i \end{aligned} \tag{۲۱}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \quad \forall j, j \neq i \end{aligned} \tag{۲۲}$$

در این روابط، r تعداد عمل‌های اتوماتا، a پارامتر پاداش و b پارامتر جریمه هستند.

با توجه به مقادیر محاسبه شده برای $\overline{Gr_{cross}}$ و $\overline{Gr_{mut}}$، اگر یکی از شرط‌های زیر برقرار باشد، اتوماتا از محیط بازخورد مثبتی را برای عمل انتخاب شده دریافت می‌کند و عمل انتخاب شده پاداش می‌گیرد و در غیر اینصورت بازخورد منفی از محیط دریافت نموده و عمل انتخاب شده جریمه می‌شود.

- شرط ۱: عمل اول انتخاب شده و $\overline{Gr_{cross}} > \overline{Gr_{mut}}$ باشد.
- شرط ۲: عمل دوم انتخاب شده و $\overline{Gr_{cross}} < \overline{Gr_{mut}}$ باشد.

شرط اول به این معنی است که افزایش نرخ عملگر تقاطع، منجر به بهبود کارایی آن نسبت به عملگر جهش شود و شرط دوم نیز به این معنی است که افزایش نرخ عملگر جهش، منجر به بهبود کارایی آن نسبت به عملگر تقاطع شود.

در ادامه این بخش، به تحلیل پیچیدگی زمانی روش پیشنهادی پرداخته می‌شود.

روش مرتب‌سازی نامغلوب که در الگوریتم پیشنهادی استفاده شده است، شامل مقایسه مقادیر توابع هدف هر راه‌حل با مقادیر توابع هدف راه‌حل‌های دیگر در جمعیت است. لذا همانطور که در مرجع [۲۴] آورده شده، دارای پیچیدگی زمانی $O(N_{obj}M^2)$ می‌باشد (N_{obj} برابر تعداد اهداف و M اندازه جمعیت است). پیچیدگی زمانی عملگر انتخاب تورنومنت جمعیتی برابر با $O(N_{obj}M\log M)$ است، زیرا در بدترین حالت، همه اعضای جمعیت رتبه یکسانی به لحاظ نامغلوبی داشته و متعلق به دسته یکسانی خواهند بود و شامل تعداد N_{obj} مرتب‌سازی مجموعه M عضوی با مرتبه زمانی $M\log M$ خواهد بود. پیچیدگی زمانی عملگرهای دیگر موجود در الگوریتم مثل عملگرهای تقاطع و جهش که شامل عملکرد اتوماتای یادگیر نیز می‌شود، حداکثر برابر $O(M)$ است. بنابراین، پیچیدگی زمانی کل الگوریتم، بیشتر تحت تاثیر الگوریتم مرتب‌سازی نامغلوب بوده و برابر با $O(N_{obj}M^2)$ است که مشابه پیچیدگی زمانی الگوریتم پایه NSGA-II است.

۵. نتایج شبیه‌سازی و ارزیابی

روش پیشنهادی با استفاده از نرم‌افزار iFogsim [۲۸] شبیه‌سازی شده و نتایج حاصل از شبیه‌سازی آن در این بخش مورد تحلیل و مقایسه قرار می‌گیرد. در این مقاله شبیه مقاله [۴]، برای مدل‌سازی نرخ ترافیک ورودی به گره‌های مه از طرف دستگاه‌های اینترنت اشیا، از مجموعه داده آرشیو گروه کاری MAWI [۲۹]، استفاده شده است. همچنین کد برنامه شبیه‌سازی در وبگاه گیت‌هاب[1] قابل دسترس است.

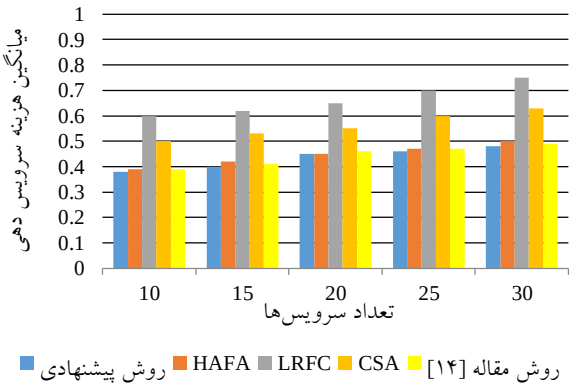
پارامترهای شبیه‌سازی و مقادیر مرتبط در جدول (۲) آورده شده است. در این بخش، نتایج حاصل از شبیه‌سازی روش پیشنهادی با روش‌های HAFA [۱۳]، LRFC [۱۹] و CSA [۶] و روش ارائه شده در مقاله [۱۴] مقایسه شده است. به منظور انتخاب روش‌های مورد مقایسه، تلاش شده روش‌هایی انتخاب شوند که به صورت چندهدفه، حداقل دو هدف از اهداف این مقاله را مد نظر قرار داده باشند. همچنین برای ارزیابی سرعت همگرایی روش پیشنهادی، پس از بررسی روش‌هایی که از الگوریتم‌های فراابتکاری استفاده کرده و به معیار سرعت همگرایی نیز توجه کرده‌اند، روش پیشنهادی با دو روش CSA [۶] و IPGA-SPP [۸]، مقایسه شده است.

در آزمایش اول، با در نظر گرفتن تعداد گره‌های مه متفاوت شامل تعداد ۵، ۱۰ و ۱۵ گره مه و تعداد سرویس‌های متفاوت از ۱۰ تا ۳۰ سرویس، میانگین هزینه سرویس‌دهی مطابق رابطه (۷) اندازه‌گیری شده است. نتایج این آزمایش در شکل‌های (۳)، (۴) و (۵) نشان داده شده است.

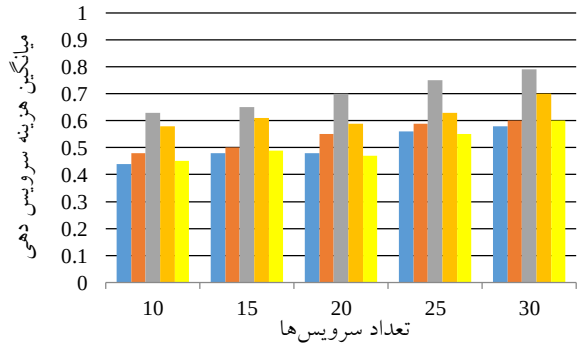
[1] https://github.com/Jameii500/serviceprovisionfog/blob/main/NSGA-LA-Serviceplacement.

جدول (۲): مقادیر پارامترهای شبیه‌سازی

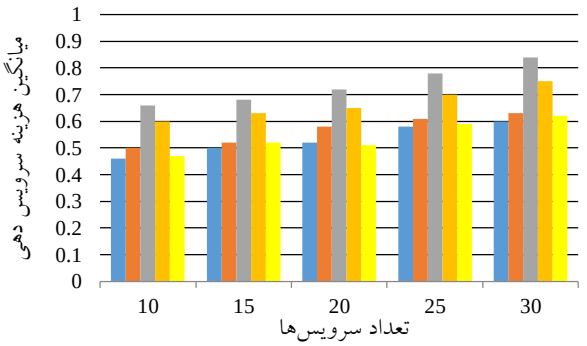
پارامتر	مقدار
gen_{max} (حداکثر تعداد نسل‌ها)	۳۰۰
M (اندازه جمعیت)	۱۰۰
مقدار اولیه نرخ جهش	۰/۰۱
p_{mLB} (کران پایین برای نرخ جهش)	۰/۰۰۱
p_{mUB} (کران بالا برای نرخ جهش)	۰/۵
مقدار اولیه نرخ تقاطع	۰/۵
p_{cLB} (کران پایین برای نرخ تقاطع)	۰/۱
p_{cUB} (کران بالا برای نرخ تقاطع)	۰/۹
α_1 (گام تغییرات نرخ تقاطع)	۰/۰۵
α_2 (گام تغییرات نرخ جهش)	۰/۰۰۱
a, b (متغیرهای پاداش و جریمه در ماشین یادگیر)	۰/۱
r (تعداد عمل‌های اتوماتا)	۲
Req_p^i	MI ۳۰۰–۱۰۰
Req_S^i	MB ۳۰۰–۱۰۰
Req_M^i	MB ۵۰–۵
$Proc^j$	MI/s ۱۰۰۰–۵۰
RAM^j	GB ۸–۴
$Storage^j$	GB ۵۰–۱۰
L_i^{req}	Byte ۵۰–۵
L_i^{resp}	Byte ۵۰–۵
$Deadline^i$	MS ۱۰
$cost_j^{process}$	per MI ۰/۰۰۱
$cost_j^{Storage}$	Per BG/S ۰/۰۰۲
$cost_j^{RAM}$	Per MB/S ۰/۰۰۴
$D_{IoT\ to\ Fog}$	MS ۵–۱
$R_{IoT\ to\ Fog}$	MBPS ۳/۵



شکل (۳): مقایسه میانگین هزینه سرویس‌دهی با ۵ گره مه



شکل (۴): مقایسه میانگین هزینه سرویس‌دهی با ۱۰ گره مه



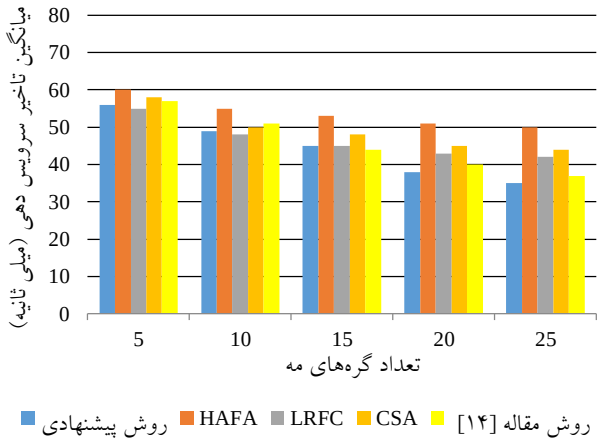
شکل (۵): مقایسه میانگین هزینه سرویس‌دهی با ۱۵ گره مه

همانطور که در شکل‌های (۳) تا (۵) مشاهده می‌شود، با افزایش تعداد سرویس‌ها، هزینه سرویس‌دهی در هر چهار روش افزایش یافته و روش پیشنهادی و روش ارائه شده در مقالـه [۱۴]، بـه دلیل تمرکز بر کاهش هزینه، به طور تقریبی شبیه هم عمل کرده و نسبت به دو روش دیگر، از هزینه کمتری برخوردار هستند.

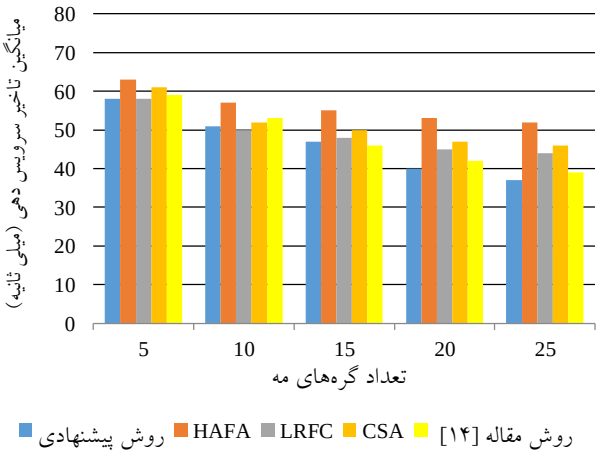
در دومین آزمایش، با در نظر گرفتن تعداد سرویس‌های متفاوت شامل ۱۰، ۲۰ و ۳۰ عدد سرویس و تعداد گره‌های مه متفاوت از ۵ تا ۲۵ گره مه، میانگین تاخیر سرویس‌دهی مطابق رابطه (۸) اندازه‌گیری شده است. نتایج این آزمایش در شکل‌های (۶)، (۷) و (۸) نشان داده شده است.

همانطور که در این شکل‌ها مشاهده می‌شود، با افزایش تعـداد گره‌های مه، تاخیر سرویس‌دهی در تمامی چهار روش کـاهش می‌یابد. این تاخیر برای روش HAFA از روش‌های دیگر بیشتر است. با افزایش تعداد گره‌های مه، کاهش تـاخیر سـرویس در روش پیشنهادی و روش مقاله [۱۴] نسبت به روش‌های دیگـر

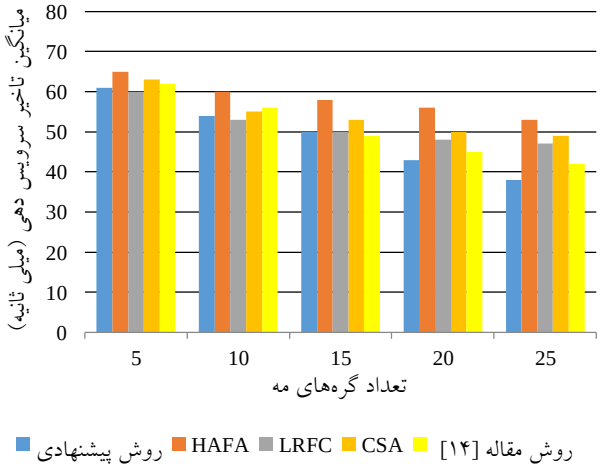
بیشتر است. همچنین با افزایش تعداد سرویس‌ها، میانگین تاخیر سرویس‌دهی افزایش می‌یابد. این امر بیشتر ناشی از تاخیر زمان انتظار سرویس در گره مه می‌باشد.



شکل (۶): مقایسه میانگین تاخیر سرویس‌دهی برای ۱۰ سرویس

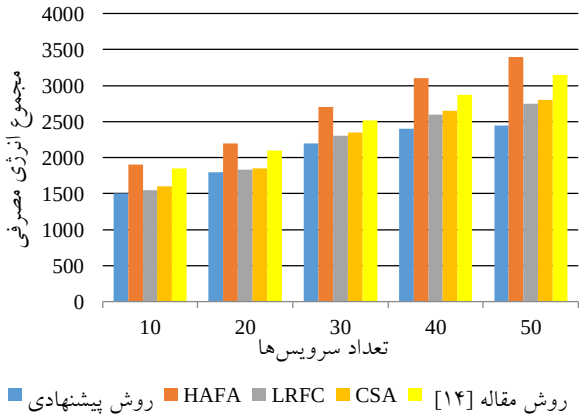


شکل (۷): مقایسه میانگین تاخیر سرویس‌دهی برای ۲۰ سرویس

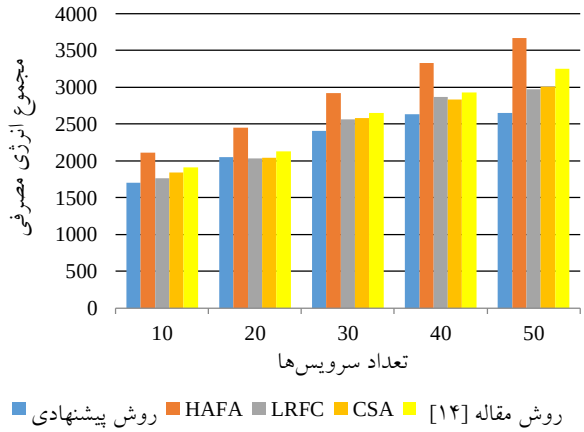


شکل (۸): مقایسه میانگین تاخیر سرویس‌دهی برای ۳۰ سرویس

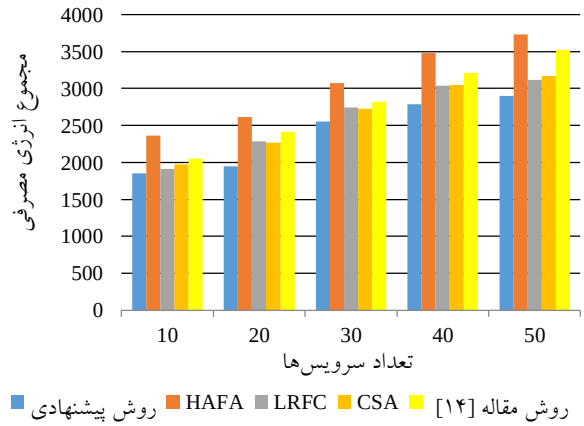
در آزمایش سوم، با در نظر گرفتن تعداد سرویس‌های متفاوت از ۱۰ تا ۵۰ سرویس و تعداد گره‌های مه متفاوت شامل ۵، ۱۰ و ۱۵ گره مه، میانگین انرژی مصرفی طبق رابطه (۹) اندازه‌گیری شده است. نتایج این آزمایش در شکل‌های (۹) تا (۱۱) نشان داده شده است.



شکل (۹): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۵ گره مه



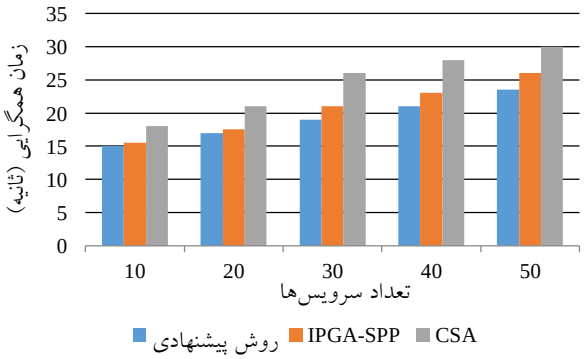
شکل (۱۰): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۱۰ گره مه



شکل (۱۱): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۱۵ گره مه

همانطور که در شکل‌های (۹) تا (۱۱) مشاهده می‌گردد، با افزایش تعداد سرویس‌ها، مجموع انرژی مصرفی در تمامی روش‌ها افزایش می‌یابد. همچنین با افزایش تعداد گره‌های مه، انرژی مصرفی افزایش می‌یابد. با توجه به اینکه روش HAFA و روش مقاله [۱۴]، تمرکز زیادی روی کاهش انرژی مصرفی ندارند، مجموع انرژی مصرفی در این روش‌ها نسبت به سایر روش‌ها بیشتر است. اما از آنجایی که کمینه کردن مصرف انرژی یکی از اهداف روش پیشنهادی است، مجموع انرژی مصرفی در این روش نسبت به روش‌های دیگر کمتر است و این کمتر بودن مصرف انرژی نسبت به روش‌های دیگر، با افزایش تعداد سرویس‌ها بیشتر نمایان می‌گردد.

همانطور که در ابتدای این بخش ذکر شد، روش پیشنهادی از نظر زمان همگرایی با دو روش CSA [۶] و IPGA-SPP [۸]، مقایسه شده است. نتایج این آزمایش در شکل (۱۲) نشان داده شده است. همانطور که در این شکل قابل مشاهده است، روش پیشنهادی از نظر زمان همگرایی بهتر از روش CSA عمل کرده است. دلیل این امر هم تنظیم پویای نرخ جهش و تقاطع در روش پیشنهادی است. همچنین روش IPGA-SPP به دلیل انجام پیکربندی موازی و تبادل راه‌حل‌های نخبه در الگوریتم ژنتیک، از زمان همگرایی بهتری نسبت به روش CSA برخوردار است.



شکل (۱۲): مقایسه زمان همگرایی با تعداد سرویس‌های مختلف

۶. نتیجه‌گیری

در این مقاله، مساله قرار دادن سرویس‌های مورد نیاز اینترنت اشیا در دستگاه‌های مه با محدودیت منابع، به صورت یک مساله بهینه‌سازی چندهدفه مدل‌سازی شد و در ادامه، روشی پویا مبتنی بر الگوریتم ژنتیک چندهدفه با رتبه‌بندی نامغلوب جهت حل این مساله ارائه گردید. در روش پیشنهادی، از اتوماتای یادگیر، جهت بهبود رفتار ژنتیکی و تنظیم پویای نرخ جهش و تقاطع استفاده شد. روش پیشنهادی با استفاده از نرم‌افزار iFogsim شبیه‌سازی شد و نتایج شبیه‌سازی با در نظر گرفتن معیارهای تاخیر سرویس، هزینه و انرژی مصرفی، مورد تحلیل قرار گرفت و با روش‌های HAFA، LRFC و CSA و روش ارائه شده در مقاله [۱۴]، مقایسه شد. نتایج شبیه‌سازی نشان داد، روش پیشنهادی با در نظر گرفتن همزمان سه معیار ذکر شده، کارایی بهتری نسبت به الگوریتم‌های مورد مقایسه دارد.

تعارض منافع: نویسندگان اعلام می‌کنند که هیچ تعارض منافعی ندارند.

مراجع

[1] J. Li, X. Shen, L. Chen, D.P. Van, J. Ou, L. Wosinska, and J. Chen, "Service Migration in Fog Computing Enabled Cellular Networks to Support Real-Time Vehicular Communications," IEEE Access, vol. 7, pp. 13704-13714, 2019, doi: 10.1109/ACCESS.2019.2893571.

[2] M. Shirkhani, K. Khamforoosh, and M. Izadbin, "Providing an improved greedy approach for increasing the number of Served users in cloud-edge networks," Soft Comput. J., vol. 10, no. 1, pp. 32-47, 2021, doi: 10.22052/scj.2022.243195.1005 [In Persian].

[3] M. Nickray and E. hosseini, "A Mobile and Fog-based Computing Method to Execute Smart Device Applications in a Secure Environment," Soft Comput. J., vol. 8, no. 1, pp. 43-57, 2019, doi: 10.22052/8.1.43 [In Persian].

[4] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H.C. Cankaya, Q. Zhang, W. Xie, and J.P. Jue, "FOGPLAN: A Lightweight QoS-Aware Dynamic Fog Service Provisioning Framework," IEEE Internet Things J., vol. 6, no. 3, pp. 5080-5096, 2019, doi: 10.1109/JIOT.2019.2896311.

[5] S. Gholamshahi and S.M.H. Hasheminejad, "A method for identifying software components based on Non-dominated Sorting Genetic Algorithm," Soft Comput. J., vol. 7, no. 2, pp. 47-64, 2019, dor: 20.1001.1.23223707.1397.7.2.4.5 [In Persian].

[6] C. Liu, J. Wang, L. Zhou, and A. Rezaeipanah, "Solving the Multi-Objective Problem of IoT Service Placement in Fog Computing Using Cuckoo Search Algorithm," Neural Process. Lett., vol. 54, no. 3, pp. 1823-1854, 2022, doi: 10.1007/s11063-021-10708-2.

[7] M. Ghobaei-Arani and A. Shahidinejad, "A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment," Expert Syst. Appl., vol. 200, p. 117012, 2022, doi: 10.1016/j.eswa.2022.117012.

[8] B. Wu, X. Lv, W.D. Shamsi, and E.G. Dizicheh, "Optimal deploying IoT services on the fog computing: A metaheuristic-based multi-objective approach," J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 10 Part B, pp. 10010-10027, 2022, doi: 10.1016/j.jksuci.2022.10.002.

[9] B.V. Natesha and R.M.R. Guddeti, "Meta-heuristic Based Hybrid Service Placement Strategies for Two-Level Fog Computing Architecture," J. Netw. Syst. Manag., vol. 30, no. 3, p. 47, 2022, doi: 10.1007/s10922-022-09660-w.

[10] M. Azimzadeh, A. Rezaee, S.J. Jassbi, and M. Esnaashari, "Placement of IoT services in fog environment based on complex network features: a genetic-based approach," Clust. Comput., vol. 25, no. 5, pp. 3423-3445, 2022, doi: 10.1007/s10586-022-03571-w.

[11] N. Sarrafzade, R. Entezari-Maleki, and L. Sousa, "A genetic-based approach for service placement in fog computing," J. Supercomput., vol. 78, no. 8, pp. 10854-10875, 2022, doi: 10.1007/s11227-021-04254-w.

[12] F. Santos, R. Immich, and E.R.M. Madeira, "Multimedia services placement algorithm for cloud-fog hierarchical environments," Comput. Commun., vol. 191, pp. 78-91, 2022, doi: 10.1016/j.comcom.2022.04.009.

[13] S. Shaik and S. Baskiyar, "Distributed service placement in hierarchical fog environments," Sustain. Comput. Informatics Syst., vol. 34, p. 100744, 2022, doi: 10.1016/j.suscom.2022.100744.

[14] M. Tekiyehband, M. Ghobaei-Arani, and A. Shahidinejad, "An efficient dynamic service provisioning mechanism in fog computing environment: A learning automata approach," Expert Syst. Appl., vol. 198, p. 116863, 2022, doi: 10.1016/j.eswa.2022.116863.

[15] S. Pallewatta, V. Kostakos, and R. Buyya, "QoS-aware placement of microservices-based IoT applications in Fog computing environments," Future Gener. Comput. Syst., vol. 131, pp. 121-136, 2022, doi: 10.1016/j.future.2022.01.012.

[16] D. Zhao, Q. Zou, and M.B. Zadeh, "A QoS-Aware IoT Service Placement Mechanism in Fog Computing Based on Open-Source Development Model," J. Grid Comput., vol. 20, no. 2, p. 12, 2022, doi: 10.1007/s10723-022-09604-3.

[17] H.O. Hassan, S. Azizi, and M. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments," IET Commun., vol. 14, no. 13, pp. 2117-2129, 2020, doi: 10.1049/iet-com.2020.0007.

[18] W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, V.B.C. Souza, A. Jukan, G.-J. Ren, and O.G. de Dios, "Evaluating the benefits of combined and continuous Fog-to-Cloud architectures," Comput. Commun., vol. 113, pp. 43-52, 2017, doi: 10.1016/j.comcom.2017.09.011.

[19] F. Murtaza, A. Akhunzada, S. ul Islam, J. Boudjadar, and R. Buyya, "QoS-aware service provisioning in fog

computing," J. Netw. Comput. Appl., vol. 165, p. 102674, 2020, doi: 10.1016/j.jnca.2020.102674.

[20] M. Zare, Y.E. Sola, and H. Hasanpour, "Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm," J. King Saud Univ. Comput. Inf. Sci., vol. 35, no. 1, pp. 368-381, 2023, doi: 10.1016/j.jksuci.2022.12.006.

[21] N.B. Salah and N.B.B. Saoud, "Adaptive data placement in the Fog infrastructure of IoT applications with dynamic changes," Simul. Model. Pract. Theory, vol. 119, p. 102557, 2022, doi: 10.1016/j.simpat.2022.102557.

[22] A. Alammari, S.A. Moiz, and A. Negi, "Enhanced layered fog architecture for IoT sensing and actuation as a service," Sci. Rep., vol. 11, p. 21693, 2021, doi: 10.1038/s41598-021-00926-y.

[23] B.V. Natesha and R.M.R. Guddeti, "Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," J. Netw. Comput. Appl., vol. 178, p. 102972, 2021, doi: 10.1016/j.jnca.2020.102972.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182-197, 2002, doi: 10.1109/4235.996017.

[25] T.-P. Hong, H.-S. Wang, W.-Y. Lin, and W.-Y. Lee, "Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process," Appl. Intell., vol. 16, no. 1, pp. 7-17, 2002, doi: 10.1023/A:1012815625611.

[26] J. Zhang, H.S.-H. Chung, and W.-L. Lo, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms," IEEE Trans. Evol. Comput., vol. 11, no. 3, pp. 326-335, 2007, doi: 10.1109/TEVC.2006.880727.

[27] K.S. Narendra and M.A.L. Thathachar, Learning automata - an introduction, Prentice Hall 1989, ISBN 978-0-13-527011-0, pp. 1-476.

[28] R. Buyya and S.N. Srirama, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit," in Fog and Edge Computing: Principles and Paradigms, Wiley, 2019, pp.433-465, doi: 10.1002/9781119525080.ch17.

[29] MAVI (Sep. 2022), Wide mawi working group traffic archive, [Online]. Available: http://mawi.wide.ad.jp.