تامین پویای سرویس در محیط مه مبتنی بر اتوماتای یادگیر و الگوریتم ژنتیک چند هدفه

سید مهدی جامعی*، استادیار

گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی واحد شهرقدس، تهران، ایران.

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 برنامه های 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 سرویس های مورد نیاز اینترنت اشیا در دستگاه های مه با محدودیت منابع، به عنوان یک مساله NP-hard شناخته می شود. در این مقاله، روشی پویا مبتنی بر الگوریتم ژنتیک چند هدفه با رتبه بندی نامغلوب جهت حل این مساله ارائه می گردد. در روش پیشنهادی، از اتوماتای یادگیر، جهت بهبود رفتار ژنتیکی و تنظیم پویای نرخ جهش و تقاطع استفاده می‌شود. روش پیشنهادی با استفاده از نرم افزار iFogsim شبیه سازی 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 به ترتیب ۱۱ و ۲۱ درصد کاهش داشته است. همچنین روش پیشنهادی از نظر میانگین تاخیر سرویس دهی نسبت به دو روش CSA و HAFA به تربیت ۷ و ۱۵ درصد کاهش داشته است. از نظر انرژی مصرفی نیز روش پیشنهادی نسبت به روش های دیگر بهبود حداقل ۸ درصدی را نشان می دهد.

واژه‌های کلیدی: اینترنت اشیا، محاسبات مه، محاسبات ابری، تامین پویای سرویس، الگوریتم ژنتیک، اتوماتای یادگیر.

* نویسنده مسئول، **Sm.jameii@iau.ac.ir**

Dynamic Service Provisioning in Fog Environment based on Learning Automata and Multi-objective Genetic Algorithm

Seyed Mahdi Jameii *, Assistant Professor

Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran.

Abstract: Nowadays, the number of applications that require a short response time is increasing, and utilizing fog environment has recently received a lot of attention. Due to the dynamics of the resource usage pattern in most Internet of Things applications, a fixed location cannot be considered for the placement and execution of services in the fog environment, and therefore the services must be dynamically placed in the fog environment. The problem of placing required Internet of Things services in cloud devices with limited resources is known as an NP-hard problem. In this article, a dynamic method based on multi-objective genetic algorithm with non-dominated ranking is presented to solve this problem. In the proposed method, learning automata are used to improve genetic behavior and dynamically adjust mutation and crossover rates. The proposed method is simulated using iFogsim and the simulation results show that the proposed method has a better efficiency than the compared algorithms by simultaneously considering the three criteria of service delay, cost and energy consumption. In terms of cost, compared to the two CSA and LRFC methods, it has decreased by 11% and 21%, respectively. Also, in terms of the average service delay, compared to the CSA and HAFA methods, the proposed method has decreased by 7% and 15 %, respectively. In terms of energy consumption, the proposed method shows an improvement of at least 8% compared to other methods.

Keywords: *IoT; Fog Computing; Cloud Computing; Dynamic Service Provisioning; Genetic Algorithm; Learning Automata.*

*** corresponding author, Sm.jameii@Iau.ac.ir**

۱. مقدمه

پیشرفت‌های اخیر در زمینه‌های اینترنت اشیا[۱] منجر به افزایش تعداد روزافزون برنامه‌های پیچیده شده است. این برنامه‌ها حساس به تأخیر بوده و نیازمند پردازش در زمان واقعی[۲] هستند. اطمینان از کیفیت سرویس[۳] برای برنامه‌های حساس به تاخیر یک امر ضروری است[۱]. این برنامه‌ها نمی‌توانند تأخیر 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 داده ها را با سرعت و بصورت بلادرنگ رسیدگی کنند و این نوع فرآیند پردازش متمرکز باعث بروز تاخیر پاسخ‌دهی می گردد. جهت غلبه بر محدودیت‌های رایانش ابر، مفهوم رایانش مه یا لبه پیشنهاد شده است[۲]. رایانش مه یک معماری است که در آن تعداد زیادی از دستگاه‌های ناهمگن، از طریق یک شبکه، باهم در ارتباط هستند و می‌توانند با تاخیر و ترافیک کم، پاسخگوی نیاز کاربران خود باشند. رایانش مه در مقایسه با محاسبات ابر دارای تأخیر 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 زمان تأخیر پردازش را کاهش دهد. محاسبات مه می‌تواند بطور موثر، نیازهای برنامه های کاربردی بلادرنگ و حساس به تاخیر را برآورده کرده و مشکلات پهنای باند شبکه را برطرف سازد[۳].

مکان‌های استقرار سرویس‌های اینترنت اشیا روی منابع موجود در محیط مه تأثیر مهمی بر زمان پاسخ کلی یک برنامه کاربردی می‌گذارند. به دلیل ماهیت پویای محیط مه و تغییرات دائمی رفتار برنامه‌های اینترنت اشیاء در طول زمان، چالش بزرگتر این است که چگونه این سرویس‌ها را بصورت 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 می شود که برای حل آن باید اهداف مختلفی بطور همزمان مورد توجه قرار گیرند. در واقع با یک مساله بهینه سازی چند هدفه 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 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 چند هدفه با رتبه بندی نامغلوب[۴] جهت حل این مساله ارائه می‌گردد که از اتوماتای یادگیر، جهت بهبود رفتار ژنتیکی و تنظیم پویای نرخ جهش (p_m) و نرخ تقاطع (p_c) استفاده می‌کند. دلیل استفاده از الگوریتم ژنتیک چند هدفه با رتبه بندی نامغلوب کارایی بالای این الگوریتم در مواجهه با مسائل بهینه‌سازی چند هدفه است که در سال‌های اخیر بسیار مورد توجه قرار گرفته و برای حل انواع مختلف مسائل بهینه‌سازی چند هدفه بکار گرفته شده 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. درحالی‌که مقادیر نرخ تقاطع و 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 بهینه‌محلی[۵] می‌گردد. عملگر جهش نیز برای حفظ تنوع راه‌حل‌ها در جمعیت

---

[۱] Internet of Things (IOT)

[۲] Real-Time

[۳] Quality of Service (QOS)

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

[۵] Local Optima

و جلوگیری از همگرایی زودرس[1] 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[2]) و نرخ تقاطع (کشف بهترین راه‌حل[3]) بطور مناسب کنترل شود. لذا جهت انجام این کار، از اتوماتای یادگیر در روش پیشنهادی استفاده شده است که بطور پویا مقادیر نرخ تقاطع و نرخ جهش را بدون نیاز به کنترل خارجی تنظیم می‌کند.

نوآوری های این مقاله عبارتند از:

- مدلسازی مساله تامین پویای سرویس در محیط بصورت چند هدفه.
- ارائه روشی مبتنی بر الگوریتم ژنتیک چند هدفه جهت حل مساله تامین پویای سرویس.
- بکارگیری اتوماتای یادگیر جهت تنظیم پویای پارامترهای الگوریتم ژنتیک چندهدفه.
- ارزیابی روش پیشنهادی جهت نشان دادن کارایی آن به لحاظ زمان سرویس‌دهی، هزینه و مصرف انرژی.

ادامه مقاله بصورت زیر سازماندهی شده است: در بخش دوم، کارهای گذشته مرتبط با مساله مرور خواهند شد. 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 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 بصورت یک مساله بهینه‌سازی چند هدفه در نظر گرفته‌اند و تلاش کرده‌اند با استفاده از روش‌های فراابتکاری و همچنین روش‌های غیر فراابتکاری این مساله را حل کنند. لذا در ادامه، کارهای مرتبط در دو زیر بخش روش‌های فراابتکاری و روش‌های غیر فراابتکاری تشریح می‌شوند.

۲-۱. روش‌های فراابتکاری

در مقاله [۶] یک چارچوب محاسباتی مبتنی بر محیط ابر-مه جهت مدیریت درخواست‌های سرویس و مکان‌یابی سرویس‌ها روی گره‌های مه ارائه شده است. در چارچوب پیشنهادی، مساله استقرار سرویس به عنوان یک مسئله 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 است. معیارهایی از قبیل بهره وری منابع مه، مصرف انرژی و تاخیر سرویس 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 اشیاء مطلوب استفاده کرد. نتایج شبیه‌سازی نشان می‌دهد که روش پیشنهادی، مصرف منابع و درصد پذیرش خدمات را افزایش می‌دهد و تاخیر سرویس و مصرف انرژی را در مقایسه با سایر مکانیسم‌های مبتنی بر فراابتکاری کاهش می‌دهد. در مقاله [۸]، مسئله قرار دادن سرویس اینترنت اشیا در محیط محاسبات مه را مد نظر قرار داده و یک الگوریتم ژنتیک موازی بهبود یافته به‌نام IPGA-SPP ارائه نموده است. از آنجایی که الگوریتم ژنتیک ممکن است در بهینه محلی گیر کند، روش پیشنهادی آن‌را به

[1] Premature Convergence
[2] Exploring the search pace
[3] Exploiting the best solution

صورت موازی با یک حافظه مشترک همراه با چندین عملگر نخبه پیکربندی کرده است. روش پیشنهادی، توزیع منابع را برای متعادل سازی بار در نظر می گیرد و اجرای سرویس را برای کاهش تأخیر اولویت بندی می کند. همچنین، با حفظ مجموعه ای از راه حل های پارتو با ایجاد مصالحه بین تأخیر سرویس، هزینه خدمات، استفاده از منابع و زمان سرویس، مساله را به عنوان یک مسئله چند هدفه حل می کند. در مقاله [۹]، مساله استقرار سرویس اینترنت اشیا بر روی گره‌های مه برای به حداقل رساندن هزینه‌های سرویس و اطمینان از کیفیت خدمات به‌عنوان یک مساله NP-hard در نظر گرفته شده است. برای حل این مساله، الگوریتم ترکیبی فراابتکاری بنام 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 داد که الگوریتم ترکیبی پیشنهادی، زمان خدمات، هزینه خدمات را به حداقل می‌رساند. در مقاله [۱۰]، یک الگوریتم قرار دادن سرویس مه مبتنی بر ویژگی شبکه‌های پیچیده بنام 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 اولویت بندی کند. این کار، میانگین تعداد پرش ها از درخواست گره ها به خدمات درخواستی را بهبود می بخشد و تاخیر و ترافیک را در شبکه کاهش می دهد. 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 غیر فراابتکاری

در مقاله [۱۲] با در نظر گرفتن یک محیط مبتنی بر محاسبات ابر–مه، دو مدل بر اساس میانگین متحرک یکپارچه اتورگرسیو [۱]و حافظه کوتاه مدت بلند مدت[۲]، جهت قرار گیری سرویس چندرسانه‌ای که از پیش‌بینی ترافیک داده آگاه است، پیشنهاد شده است. هدف اصلی، انتخاب حداقل تعداد گره‌ها با توجه به ظرفیت‌های سخت‌افزاری آن‌ها برای ارائه سرویس‌های چندرسانه‌ای به گونه‌ای است که تأخیر 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 روش پیشنهادی، 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 یادگیری برای تعیین تصمیم‌های ارائه سرویس‌ها 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[۳]) طراحی شده است که از یک حلقه کنترل خود مدیریتی برای ارائه برنامه‌های اینترنت اشیا در زیرساخت مه پیروی می‌کند. نتایج شبیه‌سازی به‌دست‌آمده با استفاده از ردیابی ترافیک مصنوعی و دنیای واقعی نشان داد که الگوریتم پیشنهادی تاخیر سرویس، هزینه و نقض تاخیر سرویس را در مقایسه با الگوریتم‌های دیگر کاهش می‌دهد. در مقاله [۱۵] روشی جهت زمان‌بندی برنامه های اینترنت اشیا آگاه از کیفیت سرویس مبتنی بر میکروسرویس برای قرار گیری در محیطهای مه پیشنهاد شده است. هدف

---

[۱] ARIMA

[۲] LSTM

[۳] Dynamic Service Provisioning Manager

اصلی روش پیشنهادی، 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[۱] جهت استقرار سرویس های اینترنت اشیا بر روی گره های مه ارائه شده است. در این روش، معیارهای هزینه سرویس و مصرف انرژی به عنوان توابع هدف در نظر گرفته شده‌اند. راه حل پیشنهادی، مصرف منابع و نسبت پذیرش سرویس‌ها را افزایش می دهد و تاخیر سرویس و مصرف انرژی را در مقایسه با سایر مکانیسم های مبتنی بر فرا ابتکاری کاهش می دهد. در مقاله [۱۷] یک رویکرد ابتکاری برای قرار دادن سرویس های اینترنت اشیا بر اساس اولویت شبکه و مصرف انرژی در محیط محاسباتی ابر و مه پیشنهاد شده است. مولفین این مقاله، برنامه های اینترنت اشیا را با حداکثر دو سرویس در نظر گرفتند و روی مه و ابر قرار دادند. مولفین مقاله [۱۸] روشی جهت سرویس دهی به برنامه های کاربردی اینترنت اشیا ارائه کردند که در آن، زمان سرویس دهی، ازدحام شبکه و مصرف انرژی را کاهش دادند. در این مقاله دو استراتژی تخصیص منبع به نام های اولین برازش[۲] وبرازش تصادفی[۳] را برای حل مشکل قرار دادن سرویس در محیط مه–ابر پیشنهاد شده است. مولفین 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 داده ها با استفاده از سیاست های مبتنی بر یادگیری معرفی شده است. در مقاله [۲۰]، روشی برای قرار دادن برنامه های 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. روش پیشنهادی 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 شبیه سازی نشان می دهد که روش پیشنهادی هزینه و تأخیر را در مقایسه با روش‌هایی مانند DDQL و IMPALA بهبود می بخشد. 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 برنامه های کاربردی اینترنت اشیا ایجاد شده است که هدف آن، به حداقل رساندن تأخیر نوشتن، خواندن و مهاجرت با استفاده از تعدادی تکرار داده است. کارایی رویکرد پیشنهادی در کاهش ذخیره‌سازی و تأخیر دسترسی با کپی‌های کمتر داده با استفاده از شبیه‌سازی نشان داده شده است.

در جدول (۱) خلاصه‌ای از روش‌های بررسی شده آورده شده است.

[۱] ODMA

[۲] First-Fit

[۳] Random Fit

جدول (۱): خلاصه کارهای مرتبط

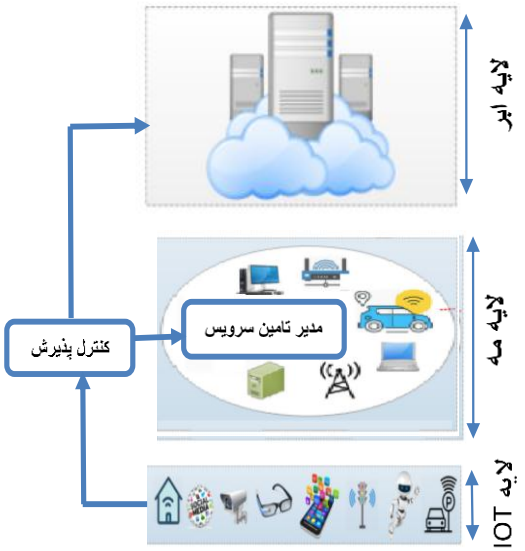
مرجع	تکنیک مورد استفاده	ابزار شبیه سازی	معیارهای کارایی (اهداف)	مزایا	معایب
[۶]	الگوریتم تکاملی مبتنی بر جستجوی فاخته	Matlab	بهره وری منابع مه، مصرف انرژی، تاخیر سرویس‌دهی	ارائه یک میان افزار کنترلی برای مدیریت سرویس ها، توجه به ناهمگونی منابع و سرویس ها	عدم تحلیل سرعت همگرایی، عدم توجه به مقیاس‌پذیری
[۷]	الگوریتم فراابتکاری بهینه‌سازی نهنگ	iFogSim	تاخیر سرویس-دهی، هزینه سرویس‌دهی	توجه به مصرف منابع و درصد پذیرش خدمات	در نظر گرفتن یک تابع شایستگی بصورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو وجواب های 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 بصورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو وجواب های نامغلوب
[۹]	ترکیب الگوریتم ژنتیک و الگوریتم بهینه‌سازی گروهی ذرات	Python	هزینه سرویس-دهی، زمان سرویس‌دهی	توجه به الزامات کیفیت سرویس، توجه به مقیاس-پذیری	در نظر گرفتن یک تابع شایستگی بصورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو وجواب های نامغلوب
[۱۰]	گروه‌بندی گره‌های مه در جوامع متعادل قبل از قرار دادن سرویس، بر اساس ساختار شبکه و وضعیت گره‌ها و پیوندها	YAFS simulator	تاخیر سرویس-دهی، مقیاس پذیری،	توجه به توازن بار در دسته-بندی گره‌های مه، اولویت-بندی جوامع گره‌ها، کاهش ترافیک شبکه	عدم توجه به مصرف انرژی، عدم بهینه‌سازی جوامع با تغییر محیط
[۱۱]	الگوریتم ژنتیک	iFogSim	مصرف انرژی، هزینه سرویس-دهی	بهبود بکارگیری منابع شبکه	در نظر گرفتن یک تابع شایستگی بصورت ترکیب وزن‌دار و عدم توجه به بحث جبهه پارتو وجواب های نامغلوب
[۱۲]	مدل ARIMA و شبکه LSTM	iFogSim	هزینه سرویس-دهی، تاخیر سرویس‌دهی	انتخاب گره‌های مه نزدیک‌تر به کاربر، کاهش درخواست-های ارسالی	عدم تحلیل انرژی مصرفی، عدم توجه به بحث جبهه پارتو وجواب های نامغلوب
[۱۳]	معماری سلسله‌مراتبی و خوشه‌بندی شده برای گره‌های مه	PFogSim (extended from CloudSim)	هزینه‌های محاسباتی و ارتباطی، زمان پاسخ، سربار	توجه به سربار، توجه به مقیاس‌پذیری	عدم توجه به معیار مصرف انرژی، عدم توجه به توازن بار
[۱۴]	در نظر گرفتن مدیر ارائه سرویس پویا مستقل مبتنی بر اتوماتای یادگیر	iFogsim	تاخیر سرویس، هزینه، نقض تاخیر سرویس	توجه به پویایی درخواست سرویس‌ها و بکارگیری اتوماتای یادگیر برای مدیریت	عدم توجه به معیار مصرف انرژی، عدم توجه به بحث جبهه پارتو وجواب های

[۱۵]	در نظر گرفتن میکروسرویس‌ها بصورت گراف جهت دار بدون دور	iFogsim	هزینه، توان عملیاتی	توجه به کیفیت سرویس، مقیاس پذیری	عدم توجه به معیار مصرف انرژی، عدم توجه به پویایی سرویس‌ها
[۱۶]	الگوریتم مدل توسعه منبع باز (ODMA)	iFogsim	تاخیر سرویس-دهی، مصرف انرژی	توجه به نسبت پذیرش سرویس‌ها	عدم تحلیل پیچیدگی زمانی، عدم توجه به مقیاس‌پذیری
[۱۷]	دسته بندی سرویس ها در دو دسته سرویس های عادی و سرویس های حیاتی	Java	مصرف انرژی، تاخیر شبکه	توجه به اولویت شبکه	عدم توجه به معیار هزینه، عدم توجه به چندین سرویس وابسته
[۱۸]	تکنیک اولین برازش و برازش تصادفی جهت اختصاص منابع	Omnetpp	زمان سرویس دهی، ازدحام شبکه، مصرف انرژی	توجه به پهنای باند شبکه، توجه به احتمال قطعی سرویس ها	عدم توجه به معیار هزینه، عدم تحلیل پیچیدگی زمانی
[۱۹]	یک لایه هوشمند بین اینترنت اشیا و ابر برای پردازش داده ها	CloudSim, iFogsim	تاخیر، مصرف انرژی	زمانبند تطبیقی و یادگیرنده برای سرویس های حساس به تاخیر	عدم توجه به معیار هزینه، عدم تحلیل پیچیدگی زمانی
[۲۰]	رویکرد جدید یادگیری تقویتی (A3C)	OpenAI Gym	هزینه، تأخیر	توجه به مهلت سرویس و محدودیت منابع	عدم توجه به معیار انرژی ، زمان پاسخ بالا برای بعضی از سرویس ها
[۲۱]	الگوریتم حریصانه به نام قرار دادن تکرارهای چندگانه داده تطبیقی برنامه های کاربردی اینترنت اشیا	iFogSim	تاخیر دسترسی به داده‌ها	کاهش تاخیر خواندن و نوشتن داده ها، کاهش تعداد مهاجرت ها	عدم توجه به معیار انرژی، عدم توجه به معیار هزینه

۳. مدلسازی و فرموله‌سازی مساله

در روش پیشنهادی، مطابق [۲۲] یک معماری سه لایه‌ای را برای اشیاء، مه و ابر در نظر می‌گیریم. در پایین‌ترین لایه که لایه اینترنت اشیاء نام دارد، تمامی حسگرها و اشیاء هوشمند نظیر تبلت، موبایل و ... قرار دارند. در لایه مه که لایه میانی می‌باشد، دستگاه هایی مانند روتر و دروازه‌های هوشمند در محیط مه، به عنوان گره های مه برای میزبانی و سرویس دهی به برنامه های کاربردی اینترنت اشیا در نظر گرفته می‌شوند. اطلاعات مختلف توسط حسگرهای لایه پایینی جمع آوری شده و به لایه مه فرستاده می‌شود. در لایه مه، درخواست‌های دریافت شده از لایه پایین‌تر به صورت هوشمند بررسی شده و درخواست‌هایی که به صورت بلادرنگ هستند و به پردازش، محاسبه و ذخیره سازی کمی نیاز داشته باشند، مورد پردازش قرار می گیرند. درخواست‌های دیگر به لایه بالاتر که همان لایه ابر است، ارسال می‌گردند. این لایه شامل سرورهای فیزیکی زیادی می‌باشد و تمرکز اصلی این مقاله بر روی چگونگی تامین سرویس در لایه مه می‌باشد. فرض می‌شود از بین گره‌های مه، گره‌ای که قابلیت‌های بیشتری به لحاظ پردازش و حافظه دارد، به عنوان مدیر تامین سرویس انتخاب می گردد و این انتخاب در بازه های زمانی قابل انجام است و بعد از آن ممکن است گره دیگری به عنوان مدیر تامین سرویس انتخاب گردد. فرآیند تصمیم گیری درخصوص اختصاص سرویس ها به گره‌های مه و بازپس

گیری آن‌ها توسط مدیر تامین سرویس انجام خواهد شد. یک ماشین نیز وظیفه کنترل پذیرش را بر عهده خواهد داشت. در خواست سرویس‌های اینترنت اشیا از لایه پایینی به ماشین کنترل پذیرش داده می شود. 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 درخواست ها وارد مدیر تامین سرویس می شود 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.



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

مساله بهینه‌سازی چند هدفه مورد نظر بصورت 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^i_p) بر حسب میلیون دستورالعمل (MI) در هر درخواست، میزان حافظه موردنیاز(Req^i_M) ، میزان فضای ذخیره سازی مورد نیاز (Req^i_S) در نظر گرفته می‌شوند. در واقع ویژگی‌های مورد نیاز هر سرویس i را بصورت مجموعه زیر در نظر می‌گیریم:

$$Service^i=\{Deadline^i, Req^i_p, Req^i_M, Req^i_S\} \quad (۱)$$

هر گره مه j دارای منابع مشخصی از قبیل قدرت پردازش ($Proc^j$)، میزان حافظه اصلی (RAM^j) و میزان فضای ذخیره سازی ($Storage^j$) است و در نتیجه هر گره مه j بصورت رابطه (۲) مشخص می‌شود:

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

اهداف مساله بهینه‌سازی چند هدفه بصورت زیر تعریف می‌گردد:

هدف ۱: کمینه‌سازی میانگین هزینه سرویس‌دهی

$$Minimize\ (F_1=Average\ Cost^{service})$$

مجموع هزینه سرویس‌دهی به درخواست‌های ارسالی از طرف اینترنت اشیا با $Cost^{service}$ نشان داده می‌شود که شامل هزینه پردازش ($Cost^{process}$)، هزینه حافظه اصلی ($Cost^{RAM}$) و هزینه فضای ذخیره سازی ($Cost^{Storage}$) است:

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

هر یک از هزینه‌های ذکر شده فوق بصورت روابط (۴) تا (۶) محاسبه می شوند:

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

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

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

Req^i_P، Req^i_M و Req^i_S به ترتیب میزان پردازش مورد نیاز (بر حسب میلیون دستورالعمل)، میزان حافظه موردنیاز (بر حسب بایت) و میزان فضای ذخیره‌سازی مورد نیاز (برحسب بایت) برای هرسرویس درخواستی i است.

$cost_j^{process}$، $cost_j^{RAM}$ و $cost_j^{Storage}$ به ترتیب هزینه پردازش (در هر میلیون دستورالعمل)، هزینه حافظه اصلی (در هر بایت در ثانیه) و هزینه فضای ذخیره سازی (در هر بایت در ثانیه) در گره مه j است. همچنین m برابر تعداد گره های مه و n حداکثر تعداد سرویس های مورد درخواست است.

در نهایت، میانگین هزینه سرویس‌دهی بصورت رابطه (۷) محاسبه می‌گردد:

$$\text{Average Cost}^{\text{service}} = \frac{\sum_{i=1}^{n} \text{Cost service}^i}{n}) \qquad (۷)$$

که در آن n تعداد کل سرویس های مورد درخواست از طرف اینترنت اشیا است.

هدف ۲: کمینه‌سازی تاخیر سرویس‌دهی

Minimize (F_2=Sevice Delay)

مدت زمان بین ارسال یک درخواست سرویس توسط یک دستگاه اینترنت اشیاء و دریافت یک پاسخ برای آن تاخیر سرویس نامیده می‌شود. متوسط تاخیر سرویس برای یک سرویس i به صورت رابطه (۸) محاسبه می‌شود:

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

که در آن $D_{\text{IoT to Fog}}$ 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 (بر حسب بایت) و L_i^{resp} میانگین اندازه پاسخ های تولید شده برای سرویس i (بر حسب بایت) است.

هدف۳: کمینه‌سازی مجموع انرژی مصرفی

Minimize (F_3=Total energy Consumption)

در این مقاله، از مدل انرژی مصرفی ارائه شده در [۲۳] استفاده شده است که معادل مجموع انرژی مصرفی در گره‌های مه است و طبق رابطه (۹) محاسبه می‌شود:

$$E_{energ}^{total} = \sum_{j=1}^{n} E_{f_j}^{energy} \qquad (۹)$$

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

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

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

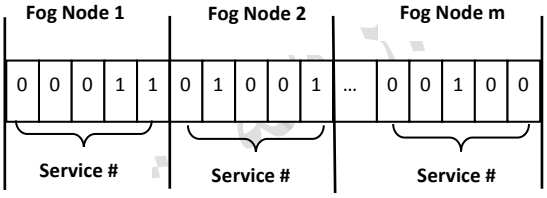
$$E_{proc}^{energy} = \int (P_f^{idle} + (\frac{(S_i^{size}}{\square_f} + P_{proc}))\ (t)\ d(t) \qquad (۱۲)$$

در این روابط، P_f^{idle} انرژی مصرفی گره مه درحالت 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 $\square_f$ به ترتیب پهنای باند بین گره‌ها و توان پردازشی گره مه است.

۴. روش پیشنهادی

در روش پیشنهادی، در ابتدای هر بازه زمانی، درخواست هایی 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 می گیرند. 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 ها را انجام خواهد داد. نحوه کدگذاری و چگونگی نمایش راه‌حل‌ها در روش پیشنهادی بصورت شکل (۲) است:



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

هر راه حل بصورت تخصیص تعدادی سرویس به ماشین‌های مه می باشد. در اینجا فرض می‌شود شماره هر سرویس در خواستی بصورت 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. بعنوان 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 در نهایت سرویس شماره ۴ (۰۰۱۰۰) روی گره مه شماره m اجرا می‌گردد.

در ابتدا جمعیتی اولیه از راه حل ها ایجاد می گردد. سپس هر راه حل در جمعیت اولیه بر اساس سه تابع هدف F_1, F_2, F_3 که در بخش قبل معرفی شدند ارزیابی می‌گردد. سپس 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. در ادامه، الگوریتم مرتب‌سازی سریع نامغلوب NSGA-II طبق الگوریتم ذکر شده در [۲۴] بر روی این جمعیت ترکیبی اعمال خواهد شد تا راه‌حل‌ها بر اساس رتبه نامغلوبی در دسته‌های مختلف fr_i ($i=1, 2,\ldots$) قرارگیرند. 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 به‌همین ترتیب ادامه می‌دهیم تا اندازه جمعیت جدید به اندازه M برسد. روال ذکر شده فوق برای ایجاد جعیت فرزندان و جمعیت ترکیبی و انتخاب جمعیت جدید تا جایی ادامه می‌یابد که به حداکثر تعداد نسل‌ها برسیم. شبه کد الگوریتم پیشنهادی در شکل (۳) نشان داده شده است.

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 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 می‌گردد. 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) بطور مناسب کنترل شود. جهت انجام این کار، از اتوماتای یادگیر در روش پیشنهادی استفاده شده است که بطور پویا مقادیر p_c و p_m را بدون نیاز به کنترل خارجی تنظیم می‌کند.

Service Provisioning Algorithm

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, $fr_{2,\ldots}$;

Step 9: Set $D_{t+1}=\emptyset$; 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.

شکل (۳): شبه کد الگوریتم تامین سرویس پیشنهادی

اتوماتای یادگیر دارای دو عمل است. این دو عمل وظیفه بروز رسانی مقادیر p_c و p_m را با توجه به بازخورد بدست 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 مقدارα_2 از p_m بروز می‌کند:

$$p_c = p_c + \alpha_1 \quad (۱۳)$$

$$p_m = p_m - \alpha_2 \quad (۱۴)$$

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 بروز می‌کند:

$$p_c = p_c - \alpha_1 \quad (۱۵)$$

$$p_m = p_m + \alpha_2 \quad (۱۶)$$

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 هریک 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 بصورت $[p_{mLB}, p_{mUB}]$ و محدوده مجاز p_c را بصورت $[p_{cLB}, p_{cUB}]$ 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 $\alpha1$ و $\alpha2$ وکران‌های بالا و پایین با توجه به ماهیت دو عملگر جهش و تقاطع و بصورت تجربی تعیین شده است. توجه به بازه

انتخابی نرخ جهش و نرخ تقاطع در مقالات معتبر چاپ شده نظیر [۲۵]، [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 بصورت زیر مشخص می‌کند:

فرض شود $sr_{offspring}$ 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 بصورت رابطه (۱۳) 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 بصورت رابطه (۱۴) محاسبه می‌کنیم:

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

این پارامتر نشان‌دهنده کارایی عملگر تقاطع در نسل فعلی است.

بطور مشابه، رتبه حاصل از عملگر جهش را بصورت رابطه (۱۹) تعریف می‌کنیم که در آن r_{new}، رتبه راه‌حل تولید شده بعد از اعمال عملگر جهش و r_{old} رتبه راه حل 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 بصورت 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 [۲۷] به ترتیب توسط روابط (۲۱) و (۲۲) صورت می‌گیرد:

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

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

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 باشد و $\overline{Gr}_{cross} > \overline{Gr}_{mut}$

شرط ۲: عمل دوم انتخاب شده باشد و $\overline{Gr}_{cross} < \overline{Gr}_{mut}$

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 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.

بنابراین همانطور که در [۲۴] آورده شده است، 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 $O(N_{obj}\ M\ logM)$ 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 ($M\ logM$) 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 $O(N_{obj}M^2)$ است که این مقدار مشابه پیچیدگی زمانی الگوریتم پایه NSGA-II است.

۵. نتایج شبیه‌سازی و ارزیابی

روش پیشنهادی با استفاده از نرم افزار iFogsim [۲۸] شبیه‌سازی شده و در این بخش نتایج حاصل از شبیه‌سازی روش پیشنهادی 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 [۳۰] قابل دسترس است.

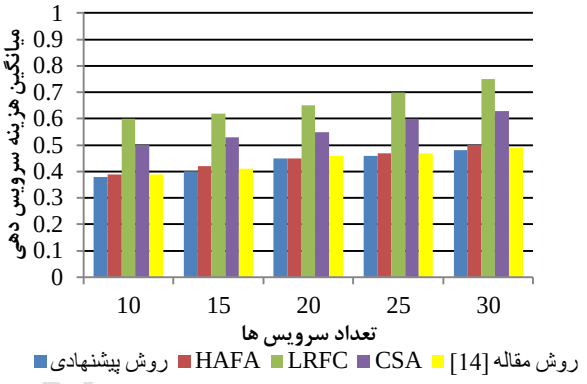
پارامترهای شبیه‌سازی و مقادیر مربوطه در جدول (۲) آورده شده است. در این آزمایشات، نتایج حاصل از شبیه‌سازی روش پیشنهادی با روش‌های HAFA [۱۳] ، LRFC [۱۹] و CSA[۶] و روش پیشنهادی مقاله [۱۴] مقایسه شده است. جهت انتخاب روش‌های مورد مقایسه، تلاش شده روش‌هایی انتخاب شوند که بصورت چند هدفه، حداقل دو هدف از اهداف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

جدول (۲): مقادیر پارامترهای شبیه‌سازی

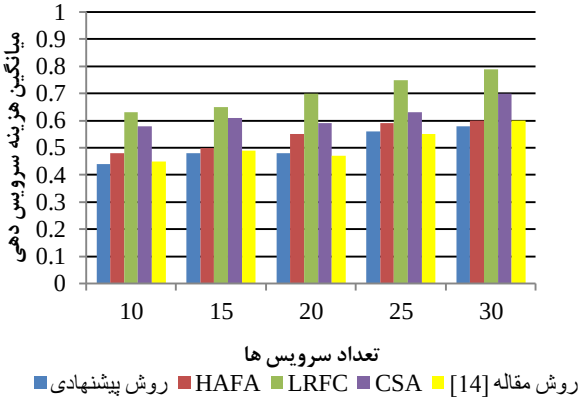
پارامتر	مقدار
gen_{max} (حداکثر تعداد نسل‌ها)	300
M (اندازه جمعیت)	100
مقدار اولیه نرخ جهش	0.01
p_{mLB} (کران پایین برای نرخ جهش)	0.001
p_{mUB} (کران بالا برای نرخ جهش)	0.5
مقدار اولیه نرخ تقاطع	0.5
p_{cLB} (کران پایین برای نرخ تقاطع)	0.1
p_{cUB} (کران بالا برای نرخ تقاطع)	0.9
α_1 (گام تغییرات نرخ تقاطع)	0.05
α_2 (گام تغییرات نرخ جهش)	0.001
a, b (پارامترهای پاداش و جریمه برای اتوماتای یادگیر)	0.1
r (تعداد عمل های اتوماتا)	2
Req^i_p	100-300 MI
Req^i_S	100-300 MB
Req^i_M	5-50 MB
$Proc^j$	50-1000 MI/s
RAM^j	4-8 GB
$Storage^j$	10-50 GB
L_i^{req}	5-50 Byte
L_i^{resp}	5-50 Byte
$Deadline^i$	10 MS
$cost_j^{process}$	0.001 per MI
$cost_j^{Storage}$	0.002 Per BG/S
$cost_j^{RAM}$	0.004 Per MB/S
$D_{IoT\ to\ Fog}$	1-5 MS
$R_{IoT\ to\ Fog}$	3.5 MBPS

آزمایش اول: اندازه‌گیری میانگین هزینه سرویس‌دهی

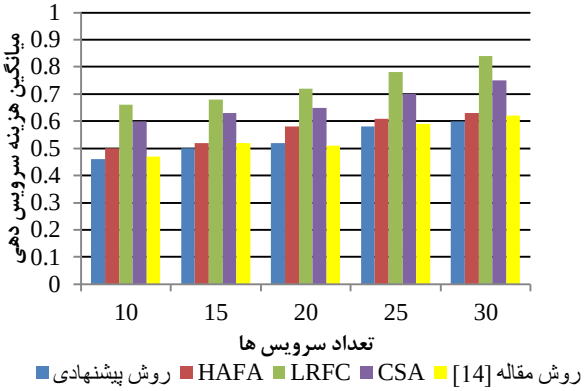
در اولین آزمایش، با در نظر گرفتن تعداد گره‌های *مه متفاوت (تعداد* ۵، ۱۰ و ۱۵ گره مه) و تعداد سرویس‌های متفاوت (از ۱۰ تا ۳۰ سرویس)، میانگین هزینه سرویس‌دهی مطابق رابطه (۷) اندازه‌گیری شده است. نتایج این آزمایش در شکل‌های (۴) و (۵) و (۶) نشان داده شده است.



شکل (۴): مقایسه میانگین هزینه سرویس‌دهی با ۵ گره مه



شکل (۵): مقایسه میانگین هزینه سرویس‌دهی با ۱۰ گره مه

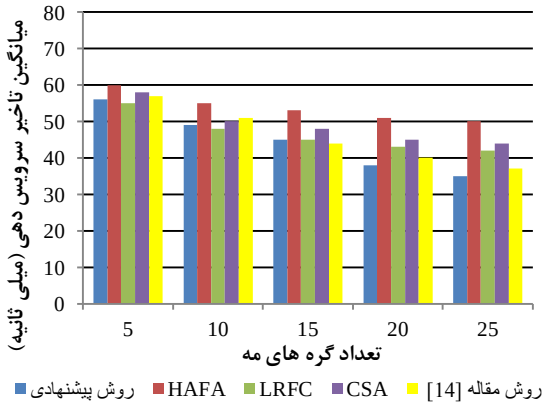


شکل (۶): مقایسه میانگین هزینه سرویس‌دهی با ۱۵ گره مه

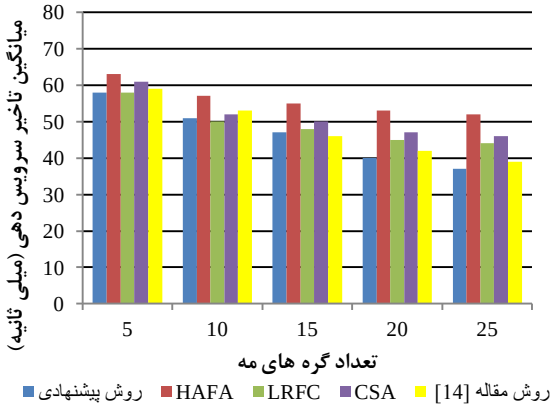
همانطور که مشاهده می‌شود، با افزایش تعداد سرویس‌ها، هزینه سرویس دهی برای هر چهار روش افزایش یافته و روش پیشنهادی و روش پیشنهادی مقاله [۱۴] به دلیل تمرکز بر کاهش هزینه، تقریبا شبیه هم عمل کرده و نسبت به دو روش دیگر، از هزینه کمتری برخوردار هستند. همچنین با افزایش تعداد سرویس‌ها، هزینه سرویس‌دهی در همه روش‌ها افزایش یافته است.

آزمایش دوم: اندازه‌گیری میانگین تاخیر سرویس‌دهی

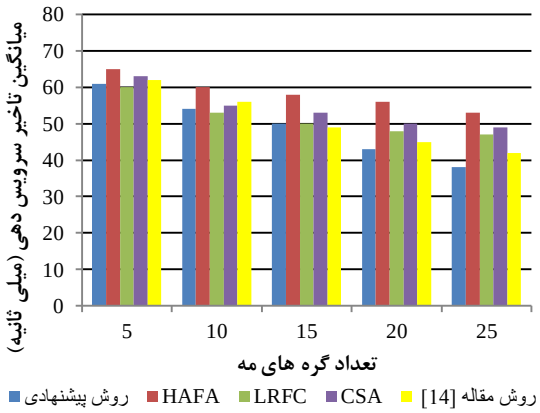
در دومین آزمایش، با در نظر گرفتن تعداد سرویس های متفاوت (۱۰، ۲۰ و ۳۰ عدد سرویس) و تعداد گره‌های مه متفاوت (از ۵ تا ۲۵ گره مه)، میانگین تاخیر سرویس‌دهی مطابق رابطه (۸) اندازه‌گیری شده است. نتایج این آزمایش در شکل‌های (۷) و (۸) و (۹) نشان داده شده است.



شکل (۷): مقایسه میانگین تاخیر سرویس دهی برای ۱۰ سرویس



شکل (۸): مقایسه میانگین تاخیر سرویس دهی برای ۲۰ سرویس

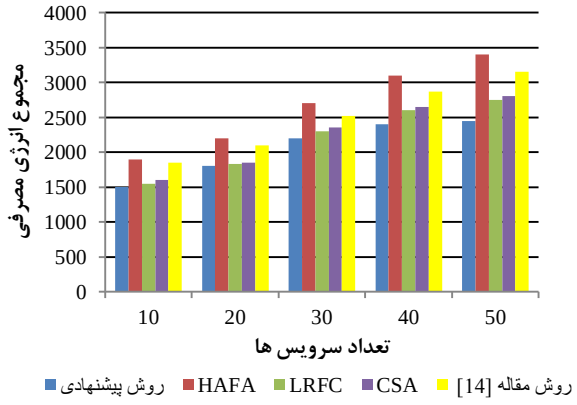


شکل (۹): مقایسه میانگین تاخیر سرویس دهی برای ۳۰ سرویس

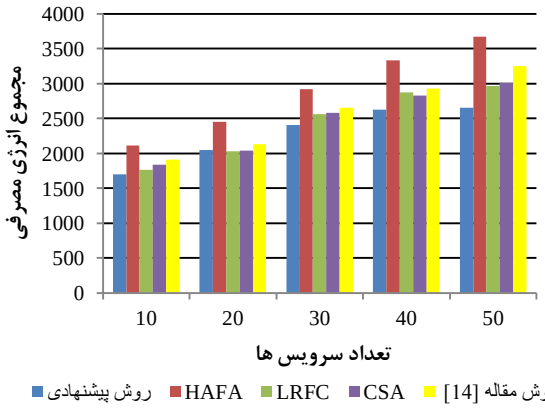
همانطور که در شکل‌های (۷) و (۸) و (۹) مشاهده می‌شود، با افزایش تعداد گره‌های مه، تاخیر سرویس دهی در هر چهار روش کاهش می یابد. این تاخیر برای روش HAFA از روش‌های دیگر بیشتر است. با افزایش تعداد گره های مه، کاهش تاخیر سرویس در روش پیشنهادی و روش مقاله [۱۴] نسبت به روش‌های دیگر بیشتر است. همچنین با افزایش تعداد سرویس‌ها، میانگین تاخیر سرویس‌دهی افزایش می‌یابد. این امر بیشتر ناشی از تاخیر زمان انتظار سرویس در گره مه می‌باشد.

آزمایش سوم: اندازه‌گیری مجموع انرژی مصرفی

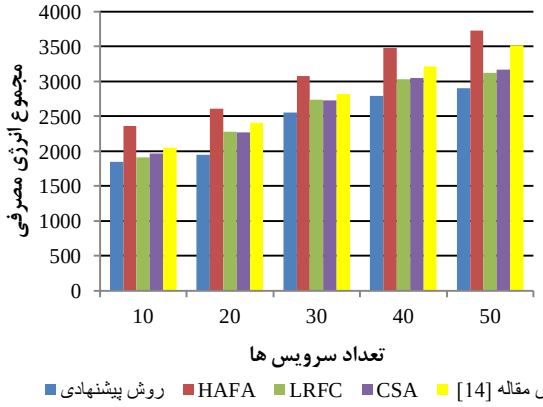
در سومین آزمایش، با در نظر گرفتن تعداد سرویس های متفاوت (۱۰ تا ۵۰ سرویس) و تعداد گره های مه متفاوت (از ۵، ۱۰ و ۱۵ گره مه)، میانگین انرژی مصرفی طبق رابطه (۹) اندازه‌گیری شده است. نتایج این آزمایش در شکل های (۱۰) و (۱۱) و (۱۲) نشان داده شده است.



شکل (۱۰): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۵ گره مه



شکل (۱۱): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۱۰ گره مه



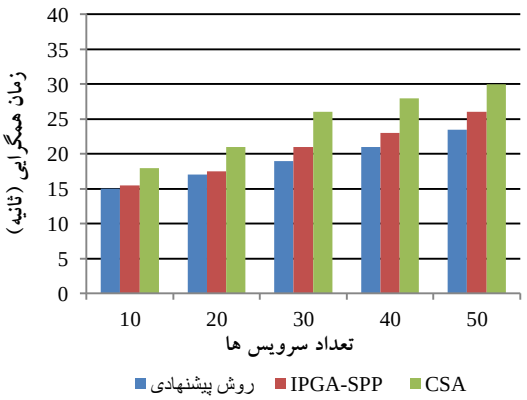
شکل (۱۲): مقایسه مجموع انرژی مصرفی با در نظر گرفتن ۱۵ گره مه

همانطور که در شکل های (۱۰) و (۱۱) و (۱۲) مشاهده می‌گردد، با افزایش تعداد سرویس‌ها، مجموع انرژی مصرفی در همه 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 روش‌های دیگر بیشتر است. با توجه به اینکه کمینه‌سازی 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 می گردد.

آزمایش چهارم: ارزیابی سرعت همگرایی

همانطور که در ابتدای این بخش ذکر شد، روش پیشنهادی از نظرزمان همگرایی با دو روش CSA [۶] و IPGA-SPP [۸] مقایسه شده است. نتایج این آزمایش در شکل (۱۳) نشان داده شده است. همانطور که در این شکل دیده می‌شود، 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 (۱۳): مقایسه زمان همگرایی با درنظر گرفتن تعداد سرویس‌های مختلف

۶. نتیجه‌گیری

در این مقاله، مساله قرار دادن سرویس‌های مورد نیاز اینترنت اشیا در دستگاه های مه با محدودیت منابع، بصورت یک مساله بهینه‌سازی چند هدفه مدل‌سازی شد و در ادامه، روشی پویا مبتنی بر الگوریتم ژنتیک چند هدفه با رتبه بندی 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 نرم افزار **iFogsim** شبیه سازی شد و نتایج شبیه سازی با در نظر گرفتن معیارهای تاخیر سرویس، هزینه و انرژی مصرفی، مورد تحلیل قرار گرفت و با روش‌های **HAFA**، **LRFC** و **CSA** و روش مقاله [۱۴] مقایسه شد. نتایج شبیه‌سازی نشان داد روش پیشنهادی با در نظر گرفتن همزمان سه معیار ذکر شده، کارایی بهتری را نسبت به الگوریتم‌های مورد مقایسه نشان می‌دهد.

مراجع

[1] J. Li, Sh. Xiaoman, Ch. Lei, P. Dung, O. Jiannan, W. Lena, and C. Jiajia, "Service migration in fog computing enabled cellular networks to support real-time vehicular communications." IEEE Access, Vol.7, pp.13704-13714, 2019.

[۲] شیرخانی، محمد، خام فروش، کیهان، ایزدبین، مهسا، ارائه روش حریصانه بهبود یافته برای افزایش تعداد کاربران سرویس داده شده در شبکه های ابر لبه، مجله محاسبات نرم، جلد ۱۰، شماره ۱، ص ۳۲-۴۷، بهار و تابستان ۱۴۰۰.

[۳] حسینی، انتصا، نیک رأی، محسن، ارائه یک روش مبتنی بر پردازش مه سیار برای اجرای برنامه‌های دستگاه‌های هوشمند در محیط امن، مجله محاسبات نرم، جلد ۸، شماره ۱، ص ۴۳-۵۷، بهار و تابستان ۱۳۹۸.

[4] A. Yousefpour, et al. "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework", IEEE Internet of Things Journal, Vol. 6, No. 3, pp. 5080-5096, 2019.

[۵] غلامشاهی، شبنم، هاشمی‌نژاد، سید محمدحسین، روشی برای تشخیص مؤلفه‌های نرم‌افزاری مبتنی بر الگوریتم ژنتیک مرتب‌سازی نامغلوب، مجله محاسبات نرم، جلد ۷، شماره ۲، ص ۴۷-۶۴، پاییز و زمستان ۱۳۹۷.

[6] L. Chang, J. Wang, L. Zhou, and A. Rezaeipanah, "Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm", Neural Processing Letters, Vol. 54, No. 3, pp. 1823-1854, 2022.

[7] M. Ghobaei-Arani, and A. Shahidinejad, A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment. *Expert Systems with Applications*, Vol. *200*, p.117012, 2022.

[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. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), pp.10010-10027, 2022.

[9] B.V. Natesha, and R.M.R Guddeti, Meta-heuristic Based Hybrid Service Placement Strategies for Two-Level Fog Computing Architecture. *Journal of Network and Systems Management*, Vol. *30*, *No.* 3, pp.1-23, 2022.

[10] M. AzimzadehA. Rezaee, S.J. Jassbi, and M. Esnaashari, Placement of IoT services in fog environment based on complex network features: a genetic-based approach. *Cluster Computing*, pp.1-23, 2022.

[11] N. Sarrafzade, R. Entezari-Maleki, and L. Sousa, A genetic-based approach for service placement in fog computing. *The Journal of supercomputing*, Vol.*78*, *No.* 8, pp.10854-10875, 2022.

[12] F. Santos, I. Roger, and R. Edmundo, "Multimedia services placement algorithm for cloud–fog hierarchical environments", Computer Communications, Vol. 191. pp. 78-91, 2022.

[13] Sh. Shaik, and B. Sanjeev, "Distributed service placement in hierarchical fog environments." Sustainable Computing: Informatics and System, Vol. 34, 2022.

[14] M. Tekiyehband, M. Ghobaei-Arani and A. Shahidinejad, An efficient dynamic service provisioning mechanism in fog computing environment: A learning automata approach. *Expert Systems with Applications*, Vol. *198*, p.116863, 2022.

[15] S. Pallewatta, K. Vassilis, and B. Rajkumar, "QoS-aware placement of microservices-based IoT applications in Fog computing environments", Future Generation Computer Systems, Vol. 131, pp. 121-136, 2022.

[16] Z. Defu, Q. Zou, and M. Boshkani Zadeh, "A QoS-Aware IoT Service Placement Mechanism in Fog Computing Based on Open-Source Development Model", Journal of Grid Computing, Vol. 20, No. 2, pp. 1-29, 2022.

[17] H. Hiwa Omer, S. Azizi, and M. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments", IET communications, vol. 14, No. 13, pp. 2117-2129, 2020.

[18] W. Ramírez, et al., "Evaluating the benefits of combined and continuous Fog-to-Cloud architectures", Computer Communications, Vol. 113, pp. 43-52, 2017.

[19] F. Murtaza, A. Akhunzada, I. Islam, B. Jalil, and B. Rajkumar, "QoS-aware service provisioning in fog computing", Journal of Network and Computer Appications, Vol. 165, 2020.

[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. *Journal of King Saud University-Computer and Information Sciences*, 2022.

[21] N.B. Salah, and N.B.B. Saoud, Adaptive data placement in the Fog infrastructure of IoT applications with dynamic changes. *Simulation Modelling Practice and Theory*, Vol. *119*, p.102557, 2022.

[22] A. Alammari, S.A. Moiz, and A. Negi,. Enhanced layered fog architecture for IoT sensing and actuation as a service. *Scientific Reports*, Vol. *11*, *No.* 1, pp.1-23, 2021.

[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, Journal of Network and Computer Applications, 178, p.102972, 2021

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002.

[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, Applied intelligence, Vol. 16, pp.7-17, 2002.

[26] J. Zhang, H.S.H. Chung, and W.L. Lo, Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Transactions on evolutionary Computation, Vol. 11, No. 3, pp.326-335, 2007.

[27] K. S. Narendra, M. A. L. Thathachar, Learning automata: An introduction, Prentice Hall, 1989.

[28] M. Redowan, and R. Buyya, "Modelling and simulation of fog and edge computing environments using iFogSim toolkit." Fog and edge computing: Principles and paradigms, pp.1-35, 2019.

[29] Wide mawi working group traffic archive," http://mawi.wide.ad.jp.

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