Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 11, Number 3, Autumn, pp. 64-82
Received: 02/12/2022, Revised: 08/28/2022, Accepted: 11/08/2022

NSGAII-based task scheduling model for smart city applications in cloud-fog environment

Atousa Daghayeghi[1] and Mohsen Nickray[2*]
1- Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran.
2*- Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran.
[1] atousa.daghayeghi@stu.qom.ac.ir, and [2*] m.nickray@qom.ac.ir

Corresponding author's address: Mohsen Nickray, Faculty of Computer Engineering and Information Technology, University of Qom, Qom, Iran.

Abstract- The advent of Internet of Things (IoT) technology has led to the concept of the smart city, in which smart devices are recognized as a necessity. The applications installed on these devices generate large volumes of data that often require real-time processing. However, these devices have limited capabilities and are not capable of processing large amounts of data. Moving all this data to cloud data centers results in higher bandwidth usage, latency, cost and energy consumption. Therefore, providing services to delay-sensitive smart city applications in the cloud is a challenging issue, and meeting the requirements of these applications requires the use of a hybrid cloud and fog paradigm. Fog computing as a complement to the cloud allows data to be processed near smart devices. However, the resources in the fog layer are heterogeneous and have different capabilities, hence, appropriate scheduling of these resources is of great importance. In this paper, the problem of task scheduling for the smart city applications in the cloud-fog environment has been addressed. To this purpose, the task scheduling problem has been modeled as a multi-objective optimization problem, which aims to minimize service delay and energy consumption of the system under deadline constraint. Then, in order to solve this problem and achieve an appropriate scheduling strategy, non-dominated sorting genetic algorithm II (NSGA-II) with customized operators has been applied. In addition, in order to improve the diversity of the population and the convergence speed of the proposed algorithm, a combination of chaotic map and opposition-based learning methods have been used to generate the initial population. Also, the approach based on the penalty function has been employed to penalize the solutions that do not meet the deadline constraint. The simulation results reveal that the proposed scheduling algorithm, compared to its best competitor, improves service response delay, waiting time, execution delay and system energy consumption by 1.49%, 1.70%, 2.7% and 1.86%, respectively. Furthermore, by properly assigning tasks to the computing nodes, compared to the best competitor, the percentage of missed-deadline tasks is reduced by 1.89%.
Keywords- Task scheduling, Smart city, Fog computing, Non-dominated sorting genetic algorithm II.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۱، شماره ۳، پاییز ۱۴۰۱ ، صفحه ۶۴-۸۲
دریافت: ۱۴۰۰/۱۱/۲۳ ، بازنگری: ۱۴۰۱/۰۶/۰۶ ، پذیرش: ۱۴۰۱/۰۸/۱۷

ارائه‌ی یک مدل زمان‌بندی وظایف مبتنی بر الگوریتم ژنتیک چند هدفه با مرتب‌سازی نامغلوب برای برنامه‌های کاربردی شهر هوشمند در محیط ابر-مه

آتوسا دقایقی[۱]، محسن نیک‌رای*[۲]
۱- دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه قم، قم، ایران.
*۲- دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه قم، قم، ایران.
[1] atousa.daghayeghi@stu.qom.ac.ir, [2*] m.nickray@qom.ac.ir

* نشانی نویسنده مسئول: محسن نیک‌رای، قم، بلوار الغدیر، دانشگاه قم، دانشکده مهندسی کامپیوتر و فناوری اطلاعات.

چکیده- ظهور تکنولوژی اینترنت اشیا مفهوم شهر هوشمند را ایجاد کرده که در این پارادایم، دستگاه‌های هوشمند به عنوان یک ضرورت شناخته می‌شوند. برنامه‌های کاربردی نصب‌شده بر روی این دستگاه‌ها باعث تولید حجم زیادی داده می‌شوند که اغلب نیازمند پردازش بلادرنگ می‌باشند. بااین‌حال، این دستگاه‌ها دارای قابلیت‌های محدودی هستند و قادر به پردازش حجم زیاد داده‌ها نمی‌باشند. انتقال همه‌ی این داده‌ها به مراکز داده‌ی ابری منجر به استفاده از پهنای باند، تاخیر، هزینه و مصرف انرژی بیشتر می‌شود. ازاین‌رو، ارائه خدمات به برنامه‌های کاربردی شهر هوشمند حساس به تاخیر در ابر یک موضوع چالش برانگیز است و پاسخگویی به نیازمندی‌های این برنامه‌ها، مستلزم استفاده از پارادایم ترکیبی ابر و مه می‌باشد. رایانش مه به عنوان مکملی برای ابر امکان می‌دهد تا داده‌ها در نزدیکی دستگاه‌های هوشمند پردازش شوند. بااین‌حال، منابع موجود در لایه‌ی مه ناهمگن و دارای قابلیت‌های متفاوتی می‌باشند، بنابراین زمان‌بندی مناسب این منابع از اهمیت زیادی برخوردار است. در این مقاله، به مسأله‌ی زمان‌بندی وظایف برای برنامه‌های کاربردی شهر هوشمند در محیط ابر-مه پرداخته شده‌است. به این منظور، مسأله‌ی زمان‌بندی وظیفه به صورت یک مسأله‌ی بهینه‌سازی چند هدفه مدل شده‌است که اهداف آن، کاهش تاخیر ارائه‌ی خدمات و مصرف انرژی سیستم با در نظر گرفتن قید مهلت زمانی می‌باشد. سپس به منظور حل این مسأله و دستیابی به استراتژی زمان‌بندی مناسب، الگوریتم ژنتیک چند هدفه با مرتب‌سازی نامغلوب با اپراتورهای سفارشی به کار گرفته شده‌است. علاوه‌براین، به منظور بهبود تنوع جمعیت و سرعت همگرایی الگوریتم پیشنهادی، برای تولید جمعیت اولیه از ترکیب روش‌های نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد استفاده شده‌است. همچنین رویکرد مبتنی بر تابع جریمه برای راه‌حل‌هایی که قید مهلت زمانی را برآورده نمی‌کنند، به کار گرفته شده‌است. نتایج شبیه‌سازی‌ها نشان می‌دهد که الگوریتم زمان‌بندی پیشنهادی، در مقایسه با بهترین رقیب خود، تاخیر ارائه‌ی خدمات، زمان انتظار، تاخیر اجرای وظیفه و مصرف انرژی سیستم را به ترتیب ۱/۴۹، ۱/۷۰، ۲/۷ و ۱/۸۶ درصد بهبود می‌دهد. علاوه‌براین، با تخصیص مناسب وظایف به گره‌های محاسباتی در مقایسه با بهترین رقیب، درصد وظایفی که مهلت زمانیشان را از دست می‌دهند به میزان ۱/۸۹ درصد کاهش می‌دهد.

واژه‌های کلیدی: زمان‌بندی وظایف، شهر هوشمند، رایانش مه، الگوریتم ژنتیک چند هدفه با مرتب‌سازی نامغلوب.

۱- مقدمه

در سال‌های اخیر، به مفهوم شهر هوشمند به عنوان ابزاری برای غلبه بر چالش‌های مرتبط با رشد سریع اسکان در شهر[۱]، توجه زیادی شده‌است. بخش‌های مختلف در شهر هوشمند، برای دستیابی به نتایج قابل توجه از طریق تجزیه و تحلیل اطلاعات بلادرنگ مشارکت می‌کنند. شهر هوشمند باعث کاهش ازدحام ترافیک و صرفه‌جویی

در مصرف انرژی می‌شود و در نتیجه، کیفیت زندگی را بهبود می‌دهد [۱]. اینترنت اشیا[۲] و هوش مصنوعی، دو تکنولوژی اطلاعات پیشرفته هستند که برای بهبود سرویس‌های موجود در شهر هوشمند، بکار گرفته می‌شوند [۲] و [۳]. اگرچه شهر هوشمند، سرویس‌های زیادی را به شهروندان ارائه می‌دهد، با این‌حال، ایجاد شهر هوشمند با چالش‌های زیادی روبرو است. چالش اول، ایجاد یک شبکه‌ی توزیع‌شده و با مقیاس بزرگ بلادرنگ و دقیق با تعداد زیادی دستگاه‌های IoT هوشمند (به عنوان مثال تلفن‌های هوشمند، دستگاه‌های پوشیدنی، حسگرها و غیره) است. از طرفی، شبکه‌ی IoT هوشمند، حجم زیادی داده و وظایف محاسباتی[۳] را تولید می‌کند، که منجر به چالش تجزیه و تحلیل داده بزرگ[۴] می‌شود [۱] و [۲]. باتوجه به افزایش تولید داده‌ها در لایه‌ی سنجش[۵] شهر هوشمند، دستگاه‌های IoT با افزایش فشار پردازش داده روبرو می‌شوند. با این حال، این دستگاه‌ها دارای قابلیت‌های محاسباتی مختلفی هستند و بیشتر آن‌ها، قابلیت پردازش داده‌ی محدودی دارند [۴]. بنابراین، ایجاد شهر هوشمند، نیازمند یک پارادایم[۶] محاسباتی برای پشتیبانی از تجزیه و تحلیل داده‌ی بزرگ است [۱].

اولین راه‌حل برای غلبه بر مسائل ذکر شده، استفاده از پارادایم رایانش ابر است که در آن، داده‌های تولید شده توسط برنامه‌های کاربردی هوشمند در دستگاه‌های IoT به مراکز داده‌ی ابری متمرکز برای پردازش ارسال می‌شوند [۵]. اگرچه، لایه‌ی ابر، منابع محاسباتی و ذخیره‌سازی نامحدودی را ارائه می‌دهد، با این‌حال، ارسال حجم زیادی از وظایف محاسباتی به آن منجر به افزایش تاخیر، افزایش ازدحام شبکه و کاهش بهره‌وری می‌شود [۶]. بنابراین، برای برنامه‌های کاربردی که به تاخیر حساس هستند و نیازمند پاسخ سریع هستند، مناسب نیست [۵]. رایانش مه[۷]، یک پارادایم محاسباتی نوظهور است، که اولین بار توسط سیسکو در سال ۲۰۱۲ معرفی شد که برای حل چالش‌ها و محدودیت‌های مربوط به رایانش ابر طراحی شده‌است [۷]. رایانش مه به عنوان یک پارادایم محاسباتی توزیع‌شده در نظر گرفته می‌شود که قابلیت‌های پردازش و ذخیره‌سازی را در دستگاه‌های میانی (به عنوان مثال دروازه‌ها[۸]، نقاط دسترسی[۹] و روترها) در لبه‌ی شبکه نزدیک به دستگاه‌های IoT ممکن می‌کند [۸]. این پارادایم برای ایجاد برنامه‌های کاربردی توزیع‌شده که حساس به تاخیر و حریم خصوصی هستند، مناسب‌تر است [۹]. با این حال، قابلیت‌های پردازشی و ذخیره‌سازی گره‌های مه کمتر از مراکز داده‌ی ابری است [۱۰]. بنابراین، وظایف محاسباتی که نسبت به تاخیر حساس هستند در مه پردازش می‌شوند، درحالیکه وظایف محاسباتی که نیازمند پاسخ سریع نیستند به مراکز داده‌ی ابری ارسال می‌شوند [۱۱]. در نتیجه، ترکیب مه و ابر یک معماری جامع یعنی مدل رایانش مه-ابر را توسعه می‌دهد که در آن، مه و ابر مکمل یکدیگر هستند و این دو نمی‌توانند جایگزین هم شوند [۹] و [۱۲]. با این حال، این مدل محاسباتی با چالش‌هایی از جمله مدیریت منابع، زمان‌بندی وظیفه[۱۰] و تعادل بار[۱۱] روبرو است.

مدیریت منبع به عنوان فرآیند تخصیص منابع پردازشی و ذخیره‌سازی به مجموعه‌ای از وظایف محاسباتی دریافت شده از برنامه‌های کاربردی تعریف می‌شود [۱۳]. زمان‌بندی وظیفه به عنوان یک فرآیند تصمیم‌گیری پیچیده در نظر گرفته می‌شود و زمان‌بندی موثر وظایف می‌تواند عملکرد سیستم‌های محاسباتی را بهبود دهد [۱۴]. بر طبق [۱۵]، اغلب مسائل زمان‌بندی شامل چهار عنصر اصلی منابع، وظایف محاسباتی، اهداف و قیود هستند. در مسأله‌ی زمان‌بندی وظیفه، نگرانی‌های کاربران بیشتر در رابطه با تاخیر سرویس، هزینه، مهلت زمانی و امنیت است در حالیکه، برای ارائه‌دهندگان سرویس کاهش مصرف انرژی، بهبود تعادل بار و استفاده منبع اهمیت دارد [۱۶]. بنابراین، ارائه‌ی یک الگوریتم زمان‌بندی مناسب که هم مزیت کاربران و هم مزیت ارائه‌دهندگان سرویس را در نظر می‌گیرد، ضرورت دارد.

برای یافتن راه‌حل برای مسأله‌ی زمان‌بندی وظیفه، رویکردهای دقیق[۱۲] و اکتشافی[۱۳] بکار گرفته می‌شوند [۱۵]. با این‌حال با توجه به np-hard بودن مسأله‌ی زمان‌بندی وظایف، رویکرد دقیق برای این مسائل عملی نمی‌باشد. رویکردهای اکتشافی قادر به یافتن پاسخ نزدیک به بهینه[۱۴] در زمان مناسب برای این مسائل هستند. الگوریتم‌های تکاملی مبتنی بر جمعیت[۱۵]، قدرتمندترین الگوریتم در میان رویکردهای اکتشافی هستند [۱۷]. به طور کلی برای حل مسائل بهینه‌سازی چند هدفه، دو روش وجود دارد [۱۸]. روش اول تبدیل مسأله به یک مسأله‌ی تک هدفه با دادن وزن متفاوت به هر هدف است. بااین‌حال، مقدار وزن از قبل باید تعیین شود و نتیجه‌ی بهینه‌سازی، تا حد زیادی تحت تاثیر وزن اختصاص داده‌شده به هر هدف است. با تغییر تقاضای کاربر (به عنوان مثال اولویت بالاتر تاخیر سرویس نسبت به مصرف انرژی)، مقادیر وزن باید مجدداً تنظیم شود و الگوریتم باید مجدداً اجرا شود. دومین روش استفاده از الگوریتم تکاملی چند هدفه برای دستیابی به مجموعه‌ای از راه‌حل‌های بهینه است که به آن‌ها مجموعه‌ی پارتو[۱۶] گفته می‌شود. به این ترتیب، سیستم نیازی به اجرای مجدد الگوریتم‌ها حتی در صورت تغییر ترجیحات کاربران ندارد. الگوریتم ژنتیک چند هدفه با مرتب‌سازی نامغلوب[۱۷] [۱۹]، یک الگوریتم تکاملی چند هدفه است که هدف اصلی آن یافتن راه‌حل‌های بهینه‌ی پارتو در یک اجرای شبیه‌سازی است. مزایای NSGA-II عبارتند از: ۱) از یک تکنیک

مرتب‌سازی نامغلوب[18] استفاده می‌کند تا راه‌حلی تا حد امکان نزدیک به راه‌حل پارتو-بهینه ارائه کند. ۲) از تکنیک فاصله‌ی ازدحامی[19] برای ارائه‌ی تنوع در راه‌حل استفاده می‌کند. ۳) همچنین از تکنیک‌های نخبه‌گرایانه[20] برای حفظ بهترین راه‌حل جمعیت فعلی در نسل بعدی استفاده می‌کند.

در این مقاله، یک طرح زمان‌بندی وظیفه مبتنی بر الگوریتم NSGA-II ارائه شده‌است که هدف آن بهینه‌سازی تاخیر ارائه‌ی خدمات و مصرف انرژی سیستم است، درحالیکه محدودیت مهلت زمانی وظایف محاسباتی را در نظر می‌گیرد. به طور خلاصه، نوآوری‌های اصلی این مقاله به صورت زیر است:

- مسأله‌ی زمان‌بندی وظیفه به صورت یک مسأله‌ی بهینه‌سازی چند هدفه با اهداف حداقل‌سازی تاخیر ارائه‌ی خدمات و مصرف انرژی سیستم تحت محدودیت مهلت زمانی در محیط مه-ابر فرموله شده‌است.
- برای حل مسأله‌ی زمان‌بندی وظیفه و یافتن یک پاسخ ممکن برای آن، الگوریتم NSGA-II با عملگرهای تقاطع[21] و جهش[22] شخصی‌سازی‌شده پیشنهاد شده‌است. از آن‌جاکه همه‌ی الگوریتم‌های فراابتکاری شامل NSGA-II ممکن است در بهینه‌ی محلی گرفتار شوند و در معرض همگرایی زود هنگام قرار گیرند، برای تولید جمعیت اولیه از ترکیب روش نگاشت بی‌نظمی[23] و یادگیری مبتنی بر تضاد[24] استفاده کرده‌ایم.
- علاوه‌براین، رویکرد تابع جریمه[25] برای راه‌حل‌های غیر ممکن که قیود مدل را برآورده نمی‌کنند، بکار گرفته شده‌است. به این منظور، راه‌حل‌هایی که محدودیت مهلت زمانی را برآورده نمی‌کنند با افزایش مقدار هزینه‌شان جریمه می‌شوند.
- عملکرد طرح زمان‌بندی وظیفه‌ی پیشنهادی، با استفاده از معیارهای ارزیابی مختلف شامل میانگین تاخیر پاسخ سرویس، میانگین درصد وظایفی که مهلت زمانیشان را از دست می‌دهند، میانگین زمان انتظار، میانگین تاخیر اجرای وظیفه، میانگین مصرف انرژی ارزیابی شده‌است و عملکرد آن با برخی از الگوریتم‌های زمان‌بندی موجود مقایسه شده‌است.

ادامه‌ی این مقاله به صورت زیر سازماندهی شده‌است. در بخش ۲، مقالات مرتبط بررسی شده‌است. در بخش ۳، مدل سیستم و فرموله‌سازی مسئله‌ی بهینه‌سازی ارائه شده‌است. در بخش ۴، به جزئیات الگوریتم زمان‌بندی پیشنهادی اختصاص یافته‌است. در بخش ۵ به ارزیابی روش پیشنهادی این مقاله پرداخته شده‌است و در نهایت، در بخش ۶، نتیجه‌گیری نهایی و کارهای آتی ارائه شده‌است.

۲- پژوهش‌های مرتبط

در این بخش به بررسی پژوهش‌های مرتبط خواهیم پرداخت. به این منظور، پژوهش‌های مرتبط در سه دسته قرار گرفته‌اند. در ابتدا، کارهای انجام شده در رابطه با شهر هوشمند مبتنی بر رایانش مه را بررسی خواهیم کرد و پس از آن، به بررسی کارهای انجام شده در رابطه با مدیریت منبع و زمان‌بندی وظایف در محیط ابر یا مه با بهره‌گیری از روش‌های بهینه‌سازی خواهیم پرداخت. در نهایت، کارهای انجام شده در محیط ترکیبی مه-ابر را بررسی خواهیم کرد.

۲-۱- شهر هوشمند مبتنی بر مه

در [۲]، یک مدل رایانش مه مشارکتی[26] در شهر هوشمند با هدف حداقل کردن تاخیر سرویس و مصرف انرژی سیستم ارائه شده که در آن، گره‌های مه منابعشان را با ارسال بارکاری[27] خود به گره‌های دیگر به اشتراک می‌گذارند. در [۲۰]، یک روش استقرار سرویس IoT[28] برای شهرهای هوشمند در رایانش لبه[29]، ارائه شده‌است که اهداف استفاده منبع، تعادل بار، مصرف انرژی گره‌های رایانش لبه تحت محدودیت زمان را در نظر می‌گیرد. به منظور یافتن راه‌حل بهینه برای این مسأله، نویسندگان از الگوریتم تکاملی مبتنی بر قوت پارتو[30] استفاده کرده‌اند. در [۲۱]، یک روش تخلیه‌بار محاسباتی[31] هوشمند برای سرویس‌های مشارکتی شهر هوشمند در رایانش لبه ارائه شده‌است که حداقل‌سازی زمان پاسخ سرویس، بهینه‌سازی مصرف انرژی و حفظ تعادل بار در میان گره‌های محاسباتی لبه را در نظر می‌گیرد. نویسندگان در [۲۲]، یک معماری شبکه‌ی شهر هوشمند که توسط رایانش مه پشتیبانی می‌شود، ارائه داده‌اند. در این کار، یک ساختار چند بخشی با در نظر گرفتن سه نوع ارتباط بین دستگاه‌ها پیشنهاد شده‌است که در کاهش تاخیر و مصرف انرژی تاثیر قابل توجهی دارد. در [۲۳]، یک معماری سه لایه برای شهر هوشمند مبتنی بر مه با هدف کاهش تاخیر سرویس ارائه شده است که در آن وسایل نقلیه به عنوان گره‌های مه در نظر گرفته شده‌اند و واحدهای کنار جاده[32] نقش یک زمان‌بند متمرکز را دارند.

۲-۲- زمان‌بندی وظایف مبتنی بر روش‌های بهینه‌سازی در محیط ابر یا مه

در سال‌های اخیر پژوهش‌های متعددی در رابطه با مسأله‌ی زمان‌بندی وظیفه انجام شده‌است که از الگوریتم‌های ابتکاری و فرااکتشافی برای یافتن یک راه‌حل نزدیک به بهینه در یک زمان مناسب استفاده می‌کنند. برخی از این پژوهش‌ها بهینه‌سازی یک هدف را در نظر می‌گیرند. برای مثال، در [۱۵]، رویکرد برنامه‌ریزی خطی عدد صحیح[۳۳] برای مدل کردن مسأله‌ی زمان‌بندی درخواست‌های IoT، با تمرکز بر کاهش تاخیر سرویس به کارگرفته شده‌است. نویسندگان رویکردی بر اساس الگوریتم ژنتیک برای دستیابی به راه‌حل‌های ممکن، ارائه داده‌اند. علاوه‌براین، در این کار، از رویکردی برای جریمه کردن راه‌حل‌هایی که محدودیت مدل را برآورده نمی‌کنند، استفاده شده‌است. نویسندگان در [۲۴]، یک رویکرد زمان‌بندی وظیفه‌ی انرژی-آگاه در محیط مه پیشنهاد داده‌اند. برای دستیابی به راه‌حل بهینه در زمان مناسب، از یک روش هیبریدی استفاده شده‌است که الگوریتم بهینه‌سازی علف هرز مهاجم[۳۴] را همراه با الگوریتم تکاملی فرهنگی[۳۵] به کار می‌گیرد. در [۲۵]، یک چارچوب زمان‌بندی جریان کاری در محیط رایانش لبه موبایل[۳۶] با هدف کاهش مصرف انرژی ارائه شده‌است. این پژوهش بر مبنای الگوریتم بهینه‌سازی پروانه (BOA)[۳۷] توسعه داده شده‌است و به منظور بهبود عملکرد BOA، نویسندگان اپراتورهای تقاطع و جهش الگوریتم ژنتیک را به کار گرفته‌اند. این پژوهش، همچنین یک روش اولویت‌بندی وظیفه را برای یافتن ترتیب اجرای وظایف به کار می‌گیرد. نویسندگان در [۲۶]، یک الگوریتم فراابتکاری ترکیبی برای زمان‌بندی وظیفه ارائه داده‌اند که از الگوریتم ژنتیک به دلیل توانایی اکتشاف خوب و از الگوریتم شبیه‌سازی تبرید[۳۸] برای توانایی بهره‌برداری آن استفاده می‌کند. هدف در نظر گرفته شده در این کار، کاهش زمان پاسخ می‌باشد.

برخی دیگر از پژوهش‌ها یک مسأله‌ی چند هدفه را در نظر می‌گیرند و موازنه بین اهداف مختلف را بررسی می‌کنند. به عنوان مثال، در [۲۷]، یک مسأله‌ی بهینه‌سازی چند هدفه با در نظر گرفتن کاهش زمان اجرای سیستم و مصرف انرژی برای مسأله‌ی زمان‌بندی وظیفه در رایانش مه پیشنهاد شده‌است. نویسندگان به منظور یافتن یک استراتژی بهینه برای تخصیص وظایف در میان منابع مه در دسترس، یک رویکرد بهینه‌سازی با نام بهینه‌سازی جفت‌گیری مورچه[۳۹]، توسعه داده‌اند. در [۲۸]، یک رویکرد بهینه‌سازی جدید با نام الگوریتم زندگی زنبورها[۴۰] برای مسأله‌ی زمان‌بندی کار در محیط رایانش مه، ارائه شده‌است که هدف آن، یافتن یک موازنه بین زمان اجرای CPU و حافظه‌ی تخصیص یافته برای کارها می‌باشد. مقاله [۲۹]، یک روش زمان‌بندی وظیفه مبتنی بر الگوریتم ژنتیک-بهینه‌سازی کلونی مورچه هیبریدی[۴۱] برای اداره کردن درخواست‌های کاربر در محیط ابر ارائه داده‌است، که اهداف آن، کاهش زمان پاسخ و تکمیل وظیفه و بهبود بازدهی می‌باشد. در [۳۰]، یک طرح تخلیه بار محاسباتی در شبکه‌ی IoT مبتنی بر مه ارائه شده‌است که اهداف آن، ایجاد موازنه بین کاهش زمان تکمیل وظیفه و مصرف انرژی دستگاه‌های هوشمند می‌باشد. طرح تخلیه‌بار محاسباتی پیشنهادی به صورت یک مسأله‌ی برنامه‌ریزی غیر خطی عدد صحیح مختلط[۴۲] مدل شده‌است و یک الگوریتم هیبریدی که مزایای الگوریتم ژنتیک و الگوریتم بهینه‌سازی ازدحام ذرات (PSO)[۴۳] را به کار می‌گیرد برای دستیابی به سیاست تخلیه‌بار بهینه پیشنهاد شده‌است. [۳۱]، مسأله‌ی مشترک مدیریت جریان داده و استقرار سرویس را در نظر می‌گیرد که هدف آن، کاهش هزینه‌ی استقرار و تاخیر سرویس می‌باشد. برای حل مسأله و دستیابی به راه‌حل ممکن، یک الگوریتم چند هدفه مبتنی بر ACO پیشنهاد شده‌است که دارای تنوع و دقت بالای راه‌حل است. در مقاله‌ی [۳۲] بر مسأله‌ی زمان‌بندی وظیفه تمرکز شده که اهداف آن حداقل کردن حافظه‌ی تخصیص یافته و زمان اجرای CPU است. برای حل این مسأله، الگوریتم جستجوی ممنوع[۴۴] استفاده شده که مزیت‌های الگوریتم بهینه‌سازی مگس میوه[۴۵] و نزدیک‌ترین همسایه‌ی تقریبی[۴۶] را برای بهبود توانایی جستجوی TSA ترکیب می‌کند. پژوهشگران در [۳۳]، یک الگوریتم زمان‌بندی ترکیبی در محیط ابر ارائه کرده‌اند که از الگوریتم جستجوی سنجاب (SSA)[۴۷] موازی و سیستم استنتاج فازی، برای تخصیص وظایف به ماشین‌های مجازی استفاده می‌کند. اهداف در نظر گرفته شده، انرژی، زمان اجرای کل و میزان امنیت می‌باشد. با تمرکز بر مسأله‌ی زمان‌بندی وظایف در محاسبات ابری، [۳۴] یک الگوریتم زمان‌بندی کار ترکیبی جدید به نام بهینه‌سازی ازدحام جزئی واکنش شیمیایی[۴۸] پیشنهاد می‌دهد که مزایای الگوریتم بهینه‌سازی واکنش‌های شیمیایی کلاسیک و بهینه‌سازی ازدحام جزئی را ترکیب می‌کند و کیفیت راه‌حل را از نظر اهدافی مانند هزینه، انرژی و زمان بهبود می‌دهد.

۲-۳- زمان‌بندی وظایف مبتنی بر روش‌های بهینه‌سازی در محیط ترکیبی مه-ابر

نویسندگان [۱۶]، بر مسأله‌ی زمان‌بندی وظیفه در محیط ابر-مه با در نظر گرفتن محدودیت مهلت زمانی تمرکز کرده‌اند. سیاست زمان‌بندی وظیفه‌ی پیشنهادی، الگوریتم اولویت مبتنی بر سستی (LBPA)[۴۹] و الگوریتم بهینه‌سازی کلونی مورچه (ACO)[۵۰] را ترکیب می‌کند. برای محاسبه‌ی اولویت وظایف، LBPA، محدودیت مهلت زمانی وظیفه را در نظر می‌گیرد و بر این اساس، صف وظایف

را مرتب می‌کند. سپس، برای دستیابی به سیاست زمان‌بندی وظیفه‌ی بهینه، الگوریتم ACO با هدف کاهش مصرف انرژی کل، به کار گرفته شده‌است. با این‌حال این کار یک مسأله‌ی بهینه‌سازی تک‌هدفه را در نظر می‌گیرد. همچنین زمان‌بندی به صورت ایستا[۵۱] در نظر گرفته شده و وظایف به طور همزمان تولید می‌شوند که به دور از واقعیت است. از طرفی نویسندگان، الگوریتم پیشنهادیشان را بر روی مجموعه داده کوچکی ارزیابی می‌کنند. پژوهشگران در [۳۵]، یک الگوریتم زمان‌بندی وظیفه در محیط ابر-مه بر اساس بهینه‌سازی مبتنی بر اکوسیستم مصنوعی[۵۲]، پیشنهاد کرده‌اند که هدف آن حداقل کردن زمان پاسخ می‌باشد. در این کار، از توانایی بهره‌برداری بالای الگوریتم ازدحام سالپ[۵۳] برای بهبود رفتار همگرایی روش پیشنهادی استفاده شده‌است. در این کار هم یک مسأله‌ی بهینه‌سازی تک‌هدفه در نظر گرفته شده و به اهدافی مانند انرژی توجهی نشده است. علاوه‌براین، مسأله‌ی زمان‌بندی به صورت ایستا می‌باشد. نویسندگان در [۳۶]، با در نظر گرفتن اهداف زمان اجرا و هزینه‌های عملیاتی، یک رویکرد جدید برای مسأله‌ی زمان‌بندی وظایف در محیط مه-ابر ارائه داده‌اند و یک الگوریتم زمان‌بندی آگاه از زمان-هزینه بر مبنای الگوریتم ژنتیک توسعه داده‌اند، که به کاربران مختلف، انعطاف‌پذیری لازم به منظور اولویت‌دهی بین زمان اجرا و هزینه ارائه می‌شود. با این‌حال در این کار برای حل مسأله‌ی بهینه‌سازی چند هدفه از الگوریتم بهینه‌سازی تک‌هدفه استفاده شده‌است و توجهی به مصرف انرژی سیستم نشده‌است. مقاله [۳۷]، یک مسأله‌ی زمان‌بندی وظیفه را به صورت یک مسأله‌ی بهینه‌سازی چند هدفه در محیط ابر-مه فرموله می‌کند. اهداف مورد نظر در این کار، کاهش زمان اجرا و هزینه‌های کل هستند. به منظور دستیابی به استراتژی زمان‌بندی بهینه، نویسندگان از الگوریتم NSGA-II استفاده کرده‌اند که اپراتورهای تقاطع و جهش گسسته را به جای اپراتورهای پیوسته به کار می‌گیرد. این روش تولید مثل اشخاص[۵۴]، منجر به کشف بیشتر فضای جستجو می‌شود. با این‌حال در این کار یک زمان‌بندی ایستا در نظر گرفته شده است و فقط به پیاده‌سازی مدل ریاضی در متلب اکتفا شده است. همچنین به مصرف انرژی سیستم توجهی نشده و محدودیت‌های مهلت زمانی در مسأله در نظر گرفته نشده‌است و از یک روش تصادفی برای مقداردهی اولیه جمعیت استفاده شده که احتمال پوشش کامل فضای جستجو را کاهش می‌دهد. [۳۸]، مسأله‌ی زمان‌بندی وظیفه در محیط ابر-مه را در نظر می‌گیرد و یک الگوریتم هیبریدی مبتنی بر SSA و علف هرز مهاجم[۵۵] برای تخصیص وظایف تولید شده به گره‌های مه و ابر مناسب پیشنهاد می‌دهد که هدف آن ایجاد موازنه بین اهداف زمان و هزینه می‌باشد.

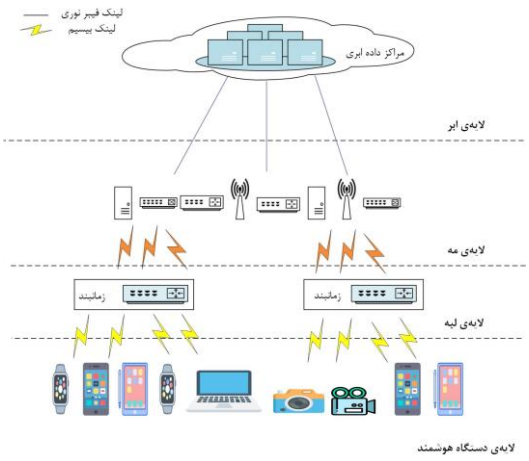
در این کار هم برای حل مسأله‌ی بهینه‌سازی چند هدفه از الگوریتم بهینه‌سازی تک هدفه استفاده شده و نویسندگان مصرف انرژی را در نظر نمی‌گیرند.

تعداد کمی از کارهای انجام‌شده، مشارکت و یکپارچه کردن محیط ابر و مه را در نظر می‌گیرند. از طرفی، بیشتر رویکردهای موجود، یک الگوریتم بهینه‌سازی پارتو را برای حل مسأله‌ی زمان‌بندی وظیفه‌ی چند هدفه به کار نمی‌گیرند. علاوه‌براین، اغلب رویکردها، زمان‌بندی ایستا را در نظر گرفته‌اند که وظایف به طور همزمان تولید می‌شوند که چنین رویکردهایی نمی‌تواند واقعیت را منعکس کند. همچنین، بیشتر این رویکردها، از روشی برای برآورده کردن قیود مسأله استفاده نمی‌کنند. در نتیجه، در این کار، یک الگوریتم زمان‌بندی وظیفه، در محیط دینامیک ابر-مه پیشنهاد شده که حداقل کردن زمان پاسخ سرویس و مصرف انرژی با محدودیت مهلت زمانی را در نظر می‌گیرد. برای دستیابی به این هدف، از الگوریتم NSGA-II استفاده شده‌است. علاوه‌براین، رویکردی مبتنی بر تابع جریمه برای اداره کردن راه‌حل‌های غیر ممکن که قیود مسأله را برآورده نمی‌کنند، به کار گرفته شده‌است.

۳- مدل سیستم

ما یک شبکه شهر هوشمند مبتنی بر مه، مشابه با شکل ۱ در نظر گرفته‌ایم. این معماری مطابق با معماری ارائه شده در [۲] است، با این حال، ما یک لایه بیشتر با نام لایه‌ی لبه در نظر گرفته‌ایم. همان‌طور که در شکل ۱ مشاهده می‌شود، معماری دارای چهار لایه‌ی دستگاه‌های هوشمند، لبه، مه و ابر می‌باشد. لایه‌ی دستگاه‌های هوشمند متشکل از مجموعه‌ی D دستگاه هوشمند است که به صورت $D = \{1, 2, \dots, D\}$ نشان داده می‌شود و شامل حسگرها، دستگاه‌های موبایل و غیره است که مجهز به تکنولوژی‌های ارتباطات مختلف هستند و توان محاسباتی و باتری محدودی دارند. این دستگاه‌ها وظایفی را تولید می‌کنند که باید در گره‌های محاسباتی موجود در لایه‌ی مه و ابر پردازش شوند. لایه‌ی لبه، مرز بین لایه‌ی دستگاه‌های هوشمند و لایه‌ی مه می‌باشد. در این لایه سرورهایی مستقر هستند که میزبان زمان‌بندهایی هستند که وظیفه‌ی زمان‌بندی وظایف دریافت‌شده از لایه‌ی دستگاه و تخصیص آن‌ها به گره‌های محاسباتی مناسب را بر عهده دارند. به منظور کاهش تاخیر و مصرف انرژی دستگاه‌های IoT، ما فرض می‌کنیم که این سرورها نزدیک به دستگاه‌های IoT و در فاصله‌ی یک هاپ از آن‌ها قرار دارند. این سرورها با منابع محاسباتی محدودی در نظر گرفته شده‌اند، بنابراین تنها تابع زمان‌بند پیشنهادی بر روی آن‌ها اجرا می‌شود و قادر به اجرای وظایف محاسباتی ارسال شده از

سمت دستگاه‌های IoT نیستند. پس از اجرای تابع زمان‌بند، وظایف به گره‌های محاسباتی مورد نظر در لایه‌ی مه و ابر ارسال می‌شوند. بر اساس [۳۹]، به طور کلی دو رویکرد مختلف متمرکز و توزیع‌شده برای کنترل استراتژی زمان‌بندی وظایف وجود دارد. در حالت متمرکز یک سرور مرکزی وجود دارد که با اجرای تابع زمان‌بند، تصمیم‌گیری را انجام می‌دهد و گره‌های محاسباتی مناسب را به وظایف بر اساس نیازمندی‌های آن‌ها اختصاص می‌دهد. در حالت توزیع‌شده، هر دستگاه هوشمند با تعامل با سایر گره‌ها، گره‌ی مناسب جهت انجام وظیفه را تعیین می‌کند. در این مقاله، ما رویکرد متمرکز را به منظور اجرای طرح زمان‌بندی به کار گرفته‌ایم. لایه‌ی مه، متشکل از مجموعه‌ی K از گره‌های مه توزیع‌شده و ناهمگن است که دارای قابلیت‌های پردازشی و ذخیره‌سازی مختلفی هستند و به صورت $K = \{1, 2, \dots, K\}$ نشان داده می‌شود. هر گره‌ی مه به صورت $< f, HC >$ مدل شده‌است که f نشان‌دهنده‌ی شناسه گره مه و HC نشان‌دهنده‌ی قابلیت‌های محاسباتی گره می‌باشد. قابلیت‌های محاسباتی گره به صورت تاپل (PR, CP, TP) است، که در آن، PR، CP و TP، به ترتیب نشان‌دهنده‌ی فرکانس CPU گره، توان محاسباتی و توان انتقال می‌باشد. در نهایت، در لایه‌ی ابر مراکز داده‌ی ابری مستقر هستند. هر مرکز داده‌ی ابری به صورت $< c, HC >$ نشان داده می‌شود که در آن، c نشان‌دهنده‌ی شناسه‌ی مرکز داده‌ی ابری و HC نشان‌دهنده‌ی قابلیت‌های سخت‌افزاری آن است که با استفاده از یک تاپل مشابه با قابلیت‌های محاسباتی گره مه نمایش داده می‌شود.



شکل ۱: مدل سیستم شامل لایه دستگاه‌های هوشمند، لایه لبه، لایه مه، لایه ابر.

هر دستگاه IoT، مجموعه‌ای از وظایف محاسباتی را تولید می‌کند که به صورت $T = \{t_1, t_2, \dots, t_n\}$ نشان داده می‌شود، که در آن n، نشان‌دهنده‌ی تعداد کل وظایف تولید شده در لایه‌ی دستگاه می‌باشد. هر وظیفه t_i به صورت یک تاپل (ts_i, cr_i, dl_i, pr_i) مدل شده‌است که ts_i نشان‌دهنده‌ی سایز داده‌ی ورودی، cr_i نشان‌دهنده‌ی نیازمندی‌های محاسباتی وظیفه، dl_i نشان‌دهنده‌ی مهلت زمانی وظیفه و pr_i نشان‌دهنده‌ی اولویت وظیفه است که با توجه به مهلت زمانی وظیفه تعیین می‌شود. در طرح ما فرض بر این است که برخی وظایف حساس به تاخیر هستند، بنابراین، به این وظایف اولویت بالاتری جهت پردازش ارائه می‌شود. نمادهایی که در این مقاله استفاده شده‌است، در جدول ۱ خلاصه شده‌اند.

جدول ۱: نمادهای استفاده شده در روش پیشنهادی.

نماد	مفهوم
D	مجموعه‌ای از دستگاه‌های هوشمند
K	مجموعه‌ی گره‌های مه
ts_i	سایز وظیفه‌ی ورودی
cr_i	نیازمندی‌های محاسباتی وظیفه
dl_i	مهلت زمانی وظیفه
pr_i	اولویت وظیفه
LR_j	نرخ انتقال لینک
LPL_j	طول لینک فیزیکی
PS_j	سرعت انتشار لینک
PR_j	نرخ پردازش گره‌ی محاسباتی
TP_j	توان انتقال گره
CP_j	توان پردازش گره
p_s	سایز جمعیت
p_c	احتمال تقاطع
p_m	احتمال جهش
maxIter	حداکثر تعداد تکرار

۳-۱- مدل محاسباتی

از آن‌جا که تمرکز اصلی این مقاله، حداقل‌سازی تاخیر پاسخ سرویس و مصرف انرژی کل سیستم است، در ادامه به نحوه‌ی محاسبه‌ی این موارد خواهیم پرداخت.

در این مقاله، تاخیر پاسخ سرویس مطابق با تعریف ارائه‌شده در [۴۰] در نظر گرفته شده‌است که در آن تاخیر سرویس برای یک درخواست برابر است با فاصله‌ی زمانی بین لحظه‌ای که دستگاه هوشمند درخواست را ارسال می‌کند و هنگامی که پاسخ آن درخواست را دریافت می‌کند که این تاخیر شامل تاخیر پردازش[۵۶]، تاخیر انتقال[۵۷]، تاخیر انتشار[۵۸] و تاخیر صف[۵۹] می‌باشد. بنابراین، تاخیر پاسخ سرویس برای t_i، با استفاده از رابطه‌ی (۱) محاسبه می‌شود.

$$SRT_i^j = TD_i^j + PD_i^j + CD_i^j + WT_i^j \quad (۱)$$

در رابطه‌ی (۱)، SRT_i^j نشان‌دهنده‌ی تاخیر پاسخ سرویس، TD_i^j نشان‌دهنده‌ی تاخیر انتقال، PD_i^j نشان‌دهنده‌ی تاخیر انتشار، CD_i^j نشان‌دهنده‌ی تاخیر پردازش و WT_i^j نشان‌دهنده‌ی تاخیر صف می‌باشد. تاخیر انتقال برای ارسال وظیفه‌ی i به گره‌ی محاسباتی j به صورت زیر محاسبه خواهد شد.

$$TD_i^j = \sum_{h=1}^{n} \frac{ts_i}{LR_j} \quad (۲)$$

در رابطه‌ی (۲)، ts_i نشان‌دهنده‌ی سایز وظیفه، h نشان‌دهنده‌ی تعداد هاپ از دستگاه تولید کننده‌ی وظیفه به گره‌ی محاسباتی مورد نظر و LR_j نرخ انتقال لینک در مسیر گره‌ی محاسباتی j می‌باشد.

برای محاسبه‌ی تاخیر انتشار از رابطه زیر استفاده شده‌است:

$$PD_i^j = \sum_{h=1}^{n} \frac{LPL_j}{PS_j} \quad (۳)$$

در رابطه‌ی (۳)، LPL_j و PS_j، به ترتیب طول لینک فیزیکی و سرعت انتشار لینک در مسیر گره‌ی محاسباتی j می‌باشد.

به منظور محاسبه‌ی تاخیر پردازش هر وظیفه از رابطه‌ی (۴) استفاده شده‌است:

$$CD_i^j = \frac{\lambda\ ts_i}{PR_j} \quad (۴)$$

در رابطه‌ی (۴)، CD_i^j تاخیر پردازش وظیفه‌ی i در گره‌ی محاسباتی j را نشان می‌دهد. λ چرخه‌ی CPU مورد نیاز برای پردازش یک بیت از وظیفه و PR_j نرخ پردازش برای گره‌ی محاسباتی j می‌باشد.

تاخیر صف برابر است با مجموع تاخیر پردازش تعداد وظایف موجود در صف گره‌ی محاسباتی j و با استفاده از رابطه‌ی (۵) محاسبه می‌شود:

$$WT_i^j = \sum_{l=1}^{n} CD_l^j \quad (۵)$$

در این رابطه، n نشان‌دهنده‌ی تعداد وظایف موجود در صف گره‌ی محاسباتی j می‌باشد.

مصرف انرژی کل سیستم برابر است با مجموع انرژی مصرف شده برای انتقال وظایف به گره‌ی محاسباتی j و انرژی مصرف شده توسط گره‌ی محاسباتی j به منظور پردازش وظیفه‌ی دریافت شده که مطابق با رابطه‌ی (۶) می‌باشد:

$$SEC_i^j = TEC_i^j + CEC_j^i \quad (۶)$$

در رابطه‌ی (۶)، SEC_i^j نشان‌دهنده‌ی انرژی مصرف‌شده برای انتقال و پردازش وظیفه‌ی i، TEC_i^j نشان‌دهنده‌ی انرژی مصرف‌شده برای انتقال وظیفه‌ی i به گره‌ی محاسبات j و CEC_j^i نشان‌دهنده‌ی انرژی مصرف‌شده توسط گره‌ی محاسباتی j برای پردازش وظیفه‌ی i می‌باشد. انرژی مصرف شده برای انتقال وظیفه i به گره‌ی محاسباتی j با استفاده از رابطه‌ی (۷) محاسبه شده‌است:

$$TEC_i^j = \sum_{h=1}^{n} \left(\frac{ts_i}{LR_j} \times TP_j\right) \quad (۷)$$

در رابطه‌ی (۷)، TP_j نشان‌دهنده‌ی توان انتقال در مسیر گره‌ی محاسباتی j می‌باشد. علاوه‌براین، انرژی مصرف‌شده توسط گره‌ی محاسباتی j برای پردازش وظیفه‌ی i با استفاده از رابطه‌ی (۸) محاسبه می‌شود:

$$CEC_i^j = CD_i^j \times CP_j \quad (۸)$$

که در این رابطه، CP_j نشان‌دهنده‌ی توان محاسباتی گره محاسباتی j می‌باشد.

۳-۲- فرموله‌سازی مسأله‌ی حداقل‌سازی تاخیر و مصرف انرژی سیستم

در این بخش، با توجه به مدل محاسباتی ذکر شده در بخش قبلی، مسأله‌ی زمان‌بندی وظیفه به صورت یک مسأله‌ی بهینه‌سازی چند هدفه فرموله شده‌است. هدف مسأله‌ی بهینه‌سازی فرموله‌شده، تخصیص وظایف به گره‌های محاسباتی مختلف در لایه‌ی مه و ابر است، به طوری‌که تاخیر سرویس و مصرف انرژی کل سیستم حداقل شود. مطابق با مدل سیستم، مسأله‌ی بهینه‌سازی به صورت زیر فرموله شده‌است:

$$Objective1: \ Minimize_S \sum_{j=1}^{M} \sum_{i=1}^{N} SRT_i^j \quad (۹)$$

$$Objective2: \ Minimize_S \sum_{j=1}^{M} \sum_{i=1}^{N} SEC_i^j \quad (۱۰)$$

به شرطی که:

$$S = < s_{ij} >_{i \in N, j \in M} \quad (۱۱)$$

$$s_{ij} \in \{0,1\}, i \in N, j \in M \quad (۱۲)$$

$$\sum_{j=1}^{M} s_{ij} = 1, \forall i \in N \quad (۱۳)$$

$$SRT_i^j \leq dl_i, \forall i \in N, \forall j \in M \quad (۱۴)$$

قیود (۱۱) و (۱۲) بیان می‌کنند که تعداد $N \times M$ متغیر تصمیم‌گیری در ماتریس S وجود دارد که در آن، N تعداد وظایف و M تعداد گره‌های محاسباتی می‌باشد و مقادیر این متغیرها باینری هستند. قید (۱۳) تضمین می‌کند که هر وظیفه، فقط به یک گره‌ی محاسباتی اختصاص داده می‌شود. در نهایت، قید (۱۴) نشان می‌دهد که وظیفه‌ی i باید قبل از مهلت زمانیش، تکمیل شده باشد.

۴- روش پیشنهادی

همانطور که قبلاً ذکر شد، اهداف مسأله بهینه‌سازی در این مقاله، حداقل‌سازی تاخیر سرویس و مصرف انرژی کل سیستم است. زمان‌بندی وظیفه به عنوان یک مسأله‌ی NP-hard در نظر گرفته می‌شود. در مسائل NP-hard، با افزایش سایز مسأله، پیچیدگی مسأله به صورت نمایی رشد می‌کند، بنابراین، راه‌حل‌های دقیق برای این مسائل مناسب نیستند [۴۰]. با انگیزه‌ی آنچه بیان شد، ما در این مقاله، یک روش زمان‌بندی وظیفه‌ی مبتنی بر NSGA-II پیشنهاد می‌دهیم.

۴-۱- طرح زمان‌بندی وظیفه مبتنی بر NSGA-II

در این بخش، مدل زمان‌بندی وظیفه‌ی پیشنهادی، با استفاده از الگوریتم NSGA-II [۱۹]، توضیح داده شده‌است. NSGA-II، یک الگوریتم قدرتمند برای حل مسائل چند هدفه‌ی تکاملی است که از مفهوم غلبه[۶۰] برای تقریب زدن مجموعه‌ای از راه‌حل‌های نامغلوب[۶۱] استفاده می‌کند [۴۱]. راه‌حل s_1، راه‌حل s_2 را مغلوب می‌کند، در صورتی‌که مقادیر همه‌ی توابع هدف در راه‌حل s_1 بهتر از مقادیر توابع هدف راه‌حل s_2 باشد. علاوه‌براین، راه‌حل s_1'، راه‌حل s_2' را مغلوب نمی‌کند، در صورتی‌که s_2' حداقل یک هدف با مقدار بهتر داشته باشد. بنابراین، راه‌حل‌های جمعیت[۶۲] به صورت راه‌حل‌های غالب و مغلوب دسته‌بندی می‌شوند. جبهه‌ی[۶۳] پارتوی بهینه، به صورت مجموعه‌ای از راه‌حل‌ها تعریف می‌شود که توسط هیچ راه‌حل دیگری مغلوب نشوند. بنابراین، راه‌حل‌های موجود در جبهه‌ی پارتو، قادرند یک یا چند هدف را بهینه کنند. ویژگی‌های اصلی الگوریتم NSGA-II عبارتند از[۳۷]: ۱) استفاده از روشی به منظور مرتب‌سازی سریع راه‌حل‌های نامغلوب، ۲) استفاده از روشی برای تخمین تراکم[۶۴] راه‌حل‌ها و ۳) استفاده از روشی به منظور انتخاب والدین[۶۵] برای تولید فرزندان[۶۶]. در مسأله‌ی زمان‌بندی وظیفه، اشخاص جمعیت، راه‌حل‌های در دسترس برای مسأله هستند. در مسأله‌ی زمان‌بندی پیشنهادی در این مقاله، هر راه‌حل جمعیت، طرح تخصیص وظایف به گره‌های مه و ابر را نشان می‌دهد. راه‌حل‌هایی که قادرند محدودیت‌ها را برآورده کنند به عنوان راه‌حل‌های ممکن برای مسأله در نظر گرفته می‌شوند.

در رویکرد پیشنهادی، زمان‌بند با دریافت تعداد مشخصی وظیفه از گره‌های IoT، تابع زمان‌بند را اجرا می‌کند. در ابتدا، زمان‌بند با ترکیب روش نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد تعداد مشخصی راه‌حل را با توجه به سایز جمعیت[۶۷] تولید می‌کند. پس از آن، مقادیر توابع هدف به ازای همه‌ی راه‌حل‌ها محاسبه می‌شوند. سپس، براساس مقادیر تابع هدف، راه‌حل‌ها با استفاده از رویکرد مرتب‌سازی نامغلوب سریع[۶۸]، رتبه‌بندی[۶۹] می‌شوند. پس از محاسبه‌ی رتبه‌ی راه‌حل‌ها، معیار فاصله‌ی ازدحامی به منظور ایجاد تمایز بین راه‌حل‌ها با رتبه‌ی یکسان محاسبه می‌شود. راه‌حل‌ها با مقادیر فاصله‌ی ازدحامی بیشتر، دارای اولویت بالاتری در نظر گرفته می‌شوند به این دلیل که این راه‌حل‌ها می‌توانند منجر به تنوع بیشتر راه‌حل‌های مسأله شوند. با توجه به دو معیار رتبه‌بندی و فاصله‌ی ازدحامی، والدین برای تولید راه‌حل‌های جدید انتخاب می‌شوند و اپراتورهای تقاطع و جهش بر روی آن‌ها اعمال می‌شود. سپس، ممکن بودن این راه‌حل‌ها با توجه به قیود در نظر گرفته شده بررسی می‌شوند و مقادیر تابع هدف با در نظر گرفتن جریمه برای راه‌حل‌های غیر ممکن محاسبه می‌شود. این فرآیند تا زمانی‌که معیار خاتمه برآورده نشده باشد، تکرار می‌شود. پس از برآورده شدن معیار خاتمه، تابع زمان‌بند، مجموعه‌ای از راه‌حل‌های نامغلوب را به عنوان راه‌حل‌های مسأله تعیین می‌کند. با توجه به بهترین راه‌حل، زمانبند وظایف را به گره‌های محاسباتی مورد نظر ارسال می‌کند. پس از اجرای وظایف توسط گره‌های محاسباتی، نتایج به گره‌های IoT منبع برگردانده می‌شود. طرح زمان‌بندی وظیفه‌ی پیشنهادی در الگوریتم ۱ ارائه شده‌است. در ادامه، مولفه‌های مدل در جزئیات بیشتری توضیح داده خواهند شد.

الگوریتم ۱: زمان‌بندی وظیفه مبتنی بر NSGA-II

ورودی: p_m ،$maxIter$،p_s

خروجی: مجموعه راه‌حل‌های نامغلوب

۱: جمعیت اولیه با سایز p_s را با استفاده از روش ترکیبی نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد تولید نمایید.

۲: مقدار هزینه‌ی اعضای جمعیت را با استفاده از رابطه‌ی (۹) و (۱۰) محاسبه کنید.

۳: با توجه به هزینه‌ی اعضای جمعیت، با استفاده از تابع مرتب‌سازی نامغلوب سریع، جبهه‌ای که هر عضو به آن متعلق است، مشخص نمایید.

۴: فاصله‌ی ازدحامی را برای اعضای موجود در هر جبهه محاسبه نمایید.

۵: **آغاز حلقه‌ی اول:** برای $i = 1$ تا $maxIter$ انجام دهید:

۶: آرایه‌ی خالی p_{off} را برای ذخیره‌ی جمعیت فرزندان ایجاد نمایید.

۷: **آغاز حلقه‌ی دوم:** برای $j = 1$ تا p_s انجام دهید:

۸: والد اول را با استفاده از چرخه‌ی رولت انتخاب نمایید.

۹: والد دوم را با استفاده از چرخه‌ی رولت انتخاب نمایید.

۱۰: اپراتور تقاطع دو نقطه‌ای را بر روی والد اول و دوم اعمال و فرزند اول و دوم را ایجاد نمایید.

۱۱: یک عدد تصادفی ایجاد نمایید.

۱۲: **آغاز شرط اول:** اگر عدد تصادفی کوچکتر از p_m است:

۱۳. اپراتور جهش را بر روی فرزند اول و دوم اعمال کنید.

۱۴: **پایان شرط اول.**

۱۵: فرزند اول و دوم را در آرایه‌ی p_{off} ذخیره کنید.

۱۶: **پایان حلقه‌ی دوم.**

۱۷: مقدار هزینه‌ی جمعیت فرزندان p_{off} را با استفاده از رابطه‌ی (۹) و (۱۰) محاسبه کنید.

۱۸: جمعیت فرزندان p_{off} و جمعیت نسل قبل را ادغام نمایید.

۱۹: جبهه‌های جمعیت ادغام‌شده را تعیین کنید.

۲۰: فاصله‌ی ازدحامی اعضای هر جبهه‌ی جمعیت ادغام‌شده را محاسبه نمایید.

۲۱: جمعیت ادغام‌شده را بر اساس جبهه‌ها و فاصله‌ی ازدحامی مرتب کنید.

۲۲: به تعداد p_s عضو از جمعیت مرتب‌شده را به عنوان جمعیت جدید انتخاب نمایید.

۲۳: **پایان حلقه‌ی اول.**

۲۴: اعضای جبهه‌ی پارتو را به عنوان راه‌حل‌های بهینه‌ی مسأله، تعیین نمایید.

۴-۱-۱- مقداردهی اولیه جمعیت

مقداردهی اولیه جمعیت، یکی از گام‌های مهم در الگوریتم‌های تکاملی است که معمولاً به صورت تصادفی اجرا می‌شود. بااین‌حال، این گام تاثیر قابل توجهی بر روی کیفیت راه‌حل، سرعت همگرایی و کاهش تلاش‌های محاسباتی دارد. مشکل مقداردهی اولیه تصادفی این است که با توجه به محدود بودن سایز جمعیت، با افزایش در تعداد ابعاد فضای جستجو، احتمال پوشش نواحی امیدوارکننده در فضای جستجو کاهش می‌یابد. بنابراین، به منظور کمک به الگوریتم‌های تکاملی برای دستیابی به راه‌حل بهتر، یک تکنیک مقداردهی اولیه مناسب باید به کار گرفته شود. تکنیک نگاشت بی‌نظمی برای تولید اعداد استفاده می‌شود. انواع مختلف نگاشت بی‌نظمی شامل نگاشت منطقی[۷۰]، نگاشت تکراری[۷۱] و غیره است که می‌توان برای تولید جمعیت اولیه از آن‌ها استفاده کرد. عملکرد الگوریتم‌های تکاملی از لحاظ تنوع جمعیت، سرعت همگرایی و فرار از بهینه‌ی محلی با استفاده از این نگاشت‌ها بهبود می‌یابد[۴۲]. از طرفی مفهوم یادگیری مبتنی بر تضاد [۴۳] برای بهبود قابلیت‌های اکتشاف الگوریتم‌های بهینه‌سازی استفاده می‌شود. در نظر گرفتن یک راه‌حل و متضاد آن می‌تواند به کشف راه‌حل بهینه سرعت بخشد و دقت راه‌حل را بهبود دهد. بنابراین، در این مقاله از ترکیب روش نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد برای تولید جمعیت اولیه استفاده شده‌است. ساختار یک راه‌حل در شکل ۲ نشان داده شده‌است. هر شخص در جمعیت، نشان‌دهنده‌ی یک راه‌حل برای مسأله‌ی زمان‌بندی است که به صورت یک آرایه‌ی دو بعدی $N \times M$ از عناصر s_{ij} با مقادیر باینری نشان داده می‌شود، که در آن، N نشان‌دهنده‌ی تعداد وظایفی است که باید زمان‌بندی شوند و M نشان‌دهنده‌ی تعداد کل گره‌های محاسباتی موجود است که برابر است با مجموع گره‌های مه و مراکز داده‌ی ابری. درصورتی‌که، وظیفه‌ی i به گره‌ی محاسباتی j اختصاص داده شود، مقدار s_{ij} برابر با یک و در غیر این صورت، مقدار آن برابر با صفر است. سایز جمعیت، تعداد راه‌حل‌های تولید شده در جمعیت را نشان می‌دهد. در ابتدا، دنباله‌های بی‌نظمی توسط نگاشت منطقی و با استفاده از رابطه (۱۵) تولید می‌شوند:

$$Z_{(I+1)} = a \times Z_I \times (1 - Z_I) \qquad (۱۵)$$

که در آن، Z_0 به صورت تصادفی انتخاب می‌شود، $Z_I \epsilon (0,1)$، I تعداد تکرار بی‌نظمی است و $a \epsilon (0,4]$. رفتار نگاشت منطقی با استفاده از پارامتر a کنترل می‌شود. از آنجا که نگاشت منطقی در مقدار $a = 4$ رفتار بی‌نظمی جامعی را نشان می‌دهد، برای پارامتر a مقدار ۴ را در نظر گرفته‌ایم [۴۳]. بعد از تولید جمعیت اولیه با استفاده از نگاشت منطقی، متضاد هر راه‌حل با استفاده از معادله‌ی (۱۶) محاسبه می‌شود:

$$s_{ij}^{o} = s_{ij}^{up} + s_{ij}^{lb} - s_{ij} \qquad (۱۶)$$

که در آن، s_{ij}^{o} عنصر iامین سطر و jامین ستون راه‌حل متضاد S^o است. s_{ij}^{up} و s_{ij}^{lb}، به ترتیب حد بالا و حد پایین راه‌حل هستند. علاوه‌براین، s_{ij} نشان‌دهنده‌ی عنصر iامین سطر و jامین ستون راه‌حل S است. سپس، راه‌حل با مقادیر پیوسته به راه‌حل‌های باینری تبدیل می‌شوند. برای انجام این کار، حداکثر مقدار هر سطر ماتریس برابر با یک و سایر مقادیر به صفر تنظیم می‌شوند. بعد از آن، مقادیر هزینه‌ی راه‌حل و متضاد آن محاسبه می‌شوند و راه‌حلی که کمترین مقدار هزینه را دارد، به عنوان جمعیت اولیه انتخاب می‌شود.

		Computing nodes				
		cn_1	cn_2	cn_3	...	cn_m
Tasks	t_1	0	0	1	...	0
	t_2	1	0	0	...	0
	t_3	0	0	0	...	1
	...	...	...	...	...	...
	t_n	0	1	0	...	0

شکل ۲: ساختار راه‌حل زمان‌بند پیشنهادی به صورت ماتریس N×M که N تعداد وظایف و M تعداد کل گره‌های محاسباتی است.

۴-۱-۲- محاسبه‌ی مقدار هزینه هر راه‌حل

با ارزیابی کیفیت اشخاص موجود در جمعیت، می‌توان در مورد اینکه کدام راه‌حل‌ها از جمعیت حذف شوند و کدام راه‌حل‌ها برای نسل[۷۲] بعدی حفظ شوند، تصمیم‌گیری کرد. در این مقاله، مقادیر هزینه‌ی هر راه‌حل با استفاده از دو تابع هدف ارزیابی می‌شوند: ۱) تاخیر سرویس و ۲) مصرف انرژی کل سیستم، که به ترتیب با استفاده از روابط (۹) و (۱۰) محاسبه می‌شوند.

۴-۱-۳- مرتب‌سازی جمعیت با استفاده از مقادیر تابع هدف

در NSGA-II برای مرتب‌سازی راه‌حل‌های جمعیت، از مکانیسم مرتب‌سازی نامغلوب سریع[۷۳]، استفاده می‌شود [۱۹]. در این روش، به هر راه‌حل s_i در جمعیت، یک مقدار عدد صحیح[۷۴] و یک مجموعه[۷۵] اختصاص داده می‌شود. بر این اساس، هر راه‌حل s_i با سایر راه‌حل‌ها در جمعیت، از لحاظ رابطه‌ی غلبه مقایسه می‌شود. درصورتی‌که، راه‌حل s_i توسط راه‌حل مقایسه‌شده، مغلوب شود، یک واحد به مقدار عدد صحیح اضافه خواهد شد، در غیر این صورت، راه‌حل مقایسه‌شده به مجموعه‌ی راه‌حل s_i اضافه خواهد شد. بنابراین، مقدار عدد صحیح نشان‌دهنده‌ی تعداد راه‌حل‌هایی است که s_i را مغلوب می‌کنند. علاوه‌براین، مجموعه‌ی اختصاص داده‌شده به راه‌حل s_i، نشان‌دهنده‌ی راه‌حل‌هایی است که توسط s_i مغلوب

می‌شوند. بر اساس این دو پارامتر به هر راه‌حل در جمعیت، یک رتبه اختصاص داده می‌شود که نشان‌دهنده‌ی جبهه‌ای است که راه‌حل به آن تعلق دارد. راه‌حل‌ها با رتبه‌ی صفر به راه‌حل‌های جبهه‌ی پارتوی بهینه متعلق هستند. پس از محاسبه‌ی جبهه‌ی پارتوی بهینه، از راه‌حل‌های موجود در آن، چشم‌پوشی می‌شود و بر این اساس، راه‌حل‌های باقیمانده برای محاسبه‌ی جبهه‌ای که به آن تعلق دارند، پردازش می‌شوند. این فرآیند تا زمانی‌که همه‌ی راه‌حل‌های موجود در جمعیت در جبهه‌های مختلف قرار گرفته باشند، تکرار می‌شود. سپس، به راه‌حل‌هایی که در جبهه‌های یکسان قرار دارند، یک رتبه‌ی تنوع[۷۶] با استفاده از استراتژی حفظ تنوع[۷۷] اختصاص داده می‌شود [۴۴]. تراکم راه‌حل‌ها در اطراف یک راه‌حل خاص در جمعیت، با استفاده از معیار فاصله‌ی ازدحامی محاسبه می‌شود که برابر است با فاصله‌ی میانگین دو راه‌حل همسایه در دو طرف آن راه‌حل در امتداد هر یک از توابع هدف. در واقع، هدف اصلی فاصله‌ی ازدحامی یافتن فاصله‌ی اقلیدسی[۷۸] بین راه‌حل‌های همسایه در یک فضای H-بعدی است که H نشان‌دهنده‌ی تعداد توابع هدف مسأله می‌باشد. مقدار بزرگتر این معیار منجر به تنوع بیشتر در راه‌حل‌های جمعیت می‌شود. بنابراین به راه‌حل با فاصله‌ی ازدحامی بیشتر، اولویت بالاتری اختصاص داده می‌شود[۱۹]. برای هر راه‌حل s_i، دو راه‌حل مجاور در جبهه‌ی یکسان در نظر گرفته می‌شوند و سپس فاصله‌ی ازدحامی راه‌حل با استفاده از رابطه‌ی (۱۷) محاسبه می‌شود:

$$CD_i = \begin{cases} \infty, & if\ i = 1\ or\ l \\ \frac{\sum_{h=1}^{H}(f_h(s_{i+1}) - f_h(s_{i-1}))}{f_h^{max} - f_h^{min}}, & otherwise \end{cases} \quad (۱۷)$$

در این رابطه، l نشان‌دهنده‌ی تعداد راه‌حل‌ها در جبهه‌ی یکسان است و H تعداد توابع هدف است. f_h^{max} و f_h^{min} مقادیر حداقل و حداکثر hمین تابع هدف هستند.

۴-۱-۴- اپراتورهای انتخاب، تقاطع و جهش

اپراتور انتخاب[۷۹]، به منظور تعیین چگونگی انتخاب والدین و ایجاد جمعیت جدید، استفاده می‌شود. روش‌های مختلفی برای انتخاب والدین وجود دارد که عبارتند از انتخاب تصادفی[۸۰]، انتخاب رقابتی[۸۱]، انتخاب چرخه‌ی رولت[۸۲] و غیره [۴۵]. در الگوریتم پیشنهادی، از رویکرد چرخه‌ی رولت برای انتخاب والدین‌ها (راه‌حل‌های نسل فعلی که فرزندان یا راه‌حل‌های نسل بعدی از آن‌ها تولید می‌شوند) استفاده شده‌است. به این منظور، به جبهه‌های مختلف، احتمالی متناسب با رتبه‌ی آن‌ها ارائه می‌شود، بنابراین جبهه با رتبه‌ی کمتر دارای احتمال بیشتری نسبت به جبهه با رتبه‌ی بالاتر است. پس از انتخاب یک جبهه، مجدداً، یکی از راه‌حل‌های موجود در جبهه‌ی انتخاب شده، با استفاده از چرخه‌ی رولت انتخاب می‌شود که راه‌حل‌ها با فاصله‌ی ازدحامی بیشتر، احتمال بالاتری برای انتخاب شدن دارند.

اپراتورهای تقاطع و جهش برای تولید راه‌حل‌های جدید از راه‌حل‌های فعلی به کار گرفته می‌شوند. تقاطع یک اپراتور اکتشاف[۸۳] است که راه‌حل‌های کاندید جدید را بر اساس راه‌حل‌های فعلی، جستجو می‌کند [۳۰]. دو راه‌حل جدید (فرزند) با اجرای اپراتور تقاطع بر روی دو والد (راه‌حل‌های فعلی)، تولید می‌شوند. روش‌های مختلفی برای تقاطع وجود دارد که عبارتند از تقاطع تک نقطه‌ای[۸۴]، تقاطع دو نقطه‌ای[۸۵] و تقاطع یکنواخت[۸۶] [۴۶]. در این مقاله، از اپراتور تقاطع دو نقطه‌ای استفاده شده‌است. بنابراین برای هر دو راه‌حل والد، دو عدد صحیح بین دو و تعداد گره‌های محاسباتی به عنوان نقطه‌ی تقاطع انتخاب می‌شود. سپس، هر راه‌حل به سه بخش با استفاده از نقاط تقاطع تقسیم می‌شود و زیر راه‌حل‌ها با یکدیگر ترکیب شده و راه‌حل‌های جدید را تولید می‌کنند.

جهش یک اپراتور مهم در الگوریتم‌های تکاملی است که فضای جستجوی راه‌حل‌ها را گسترش می‌دهد. این اپراتور، منجر به کشف نواحی امیدوار کننده در فضای جستجو و ارائه‌ی تنوع در میان جمعیت می‌شود و از همگرایی جمعیت به نقاط بهینه‌ی محلی جلوگیری می‌کند. در این کار، به منظور انجام عملیات جهش، سه اپراتور جهش مختلف تعریف شده‌است. اپراتور اول، زمان‌بند را مجبور می‌کند که وظیفه‌ی مورد نظر را به یکی از مراکز داده‌ی ابری که به صورت تصادفی انتخاب شده، ارسال کند. در اپراتور دوم، برنامه‌ی تخصیص دو وظیفه که به طور تصادفی انتخاب شده‌اند، در گره‌های محاسباتی، مبادله می‌شود. اپراتور سوم، مقدار یک ژن که به طور تصادفی انتخاب‌شده به مقدار مخالف تغییر می‌دهد.

۴-۱-۵- بررسی امکان‌سنجی راه‌حل‌ها و ارزیابی مجدد مقدار هزینه‌ی راه‌حل‌ها

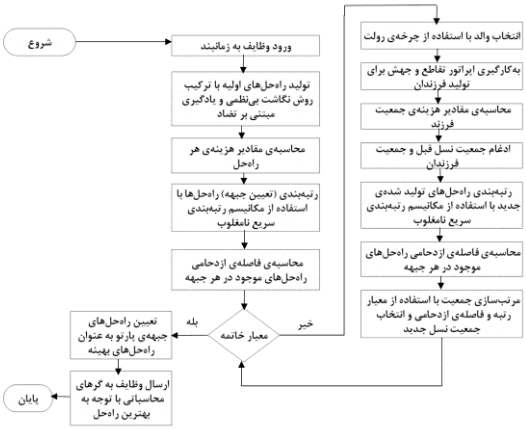
تولید جمعیت جدید با استفاده از اپراتورهای تقاطع و جهش ممکن است منجر به تولید راه‌حل‌هایی شود که قیود مسأله را برآورده نمی‌کنند. به این منظور، برای برآورده کردن قیود مسئله، مشابه با [۳۰]، در این مقاله، رویکرد تابع جریمه به کار گرفته شده‌است. به راه‌حل‌هایی که قید (۱۴) مرتبط با مهلت زمانی وظیفه را برآورده نکنند و میزان تاخیر پاسخ آن‌ها بیشتر از مهلت زمانی وظیفه باشد، جریمه‌ای اختصاص داده خواهد شد. مقدار جریمه با میزان تخطی از قید متناسب است. در ابتدا، میزان تخطی هر وظیفه $V(i)$، با استفاده از رابطه‌ی (۱۸) محاسبه خواهد شد:

$$V(i) = \begin{cases} 0, & if\ SRT_i^j \le dl_i \\ SRT_i^j - dl_i, & otherwise \end{cases} \quad (۱۸)$$

مقدار جریمه به صورت مجموع میزان تخطی همه‌ی وظایف در آن راه‌حل در نظر گرفته می‌شوند. بنابراین، با افزایش هزینه‌ی این راه‌حل‌ها، احتمال باقی ماندن آن‌ها در نسل‌های بعدی کاهش خواهد یافت.

۴-۱-۶- شرایط خاتمه

بر طبق [۴۷]، سه نوع شرایط خاتمه در الگوریتم‌های تکاملی عبارتند از: ۱) رسیدن به حداکثر تعداد تکرار از پیش تعریف شده، ۲) رسیدن به حداکثر تعداد ارزیابی توابع هدف از پیش تعریف‌شده و ۳) رسیدن به تعداد معینی تکرار بدون تغییر قابل توجهی در مقدار تابع هدف. معیار خاتمه در الگوریتم زمان‌بندی پیشنهادی، رسیدن به تعداد معینی از تکرارها می‌باشد. در شکل ۳، مراحلی که زمان‌بند با دریافت وظایف ورودی انجام می‌دهد، نشان داده شده‌است.



شکل ۳: فلوچارت تابع زمان‌بند پیشنهادی مبتنی بر الگوریتم NSGA-II بهبود یافته

۴-۱-۷- پیچیدگی زمانی الگوریتم پیشنهادی

پیچیدگی زمانی یک تکرار الگوریتم به شرح زیر است:

۱) پیچیدگی مقداردهی اولیه جمعیت: که مرتبط است با پیچیدگی زمانی برای تولید جمعیت مبتنی بر نگاشت بی‌نظمی و جمعیت متضاد که برابر است با $O(PLI)$ که P سایز جمعیت، L تعداد متغیرها (ابعاد) و I تعداد تکرارهای نگاشت بی‌نظمی است.

۲) پیچیدگی مربوط به بخش مرتب‌سازی نامغلوب: به منظور شناسایی راه‌حل‌های اولین جبهه‌ی نامغلوب در جمعیت با سایز P، هر راه‌حل با هر راه‌حل دیگر در جمعیت مقایسه می‌شود تا مشخص شود که بر آن غالب است یا خیر که این نیازمند $O(HP)$ مقایسه برای هر راه‌حل است که H تعداد توابع هدف است. وقتی‌که این فرآیند برای یافتن همه‌ی اعضای اولین سطح نامغلوب در جمعیت انجام می‌شود، پیچیدگی کل $O(HP^2)$ است.

۳) پیچیدگی مربوط به تخصیص فاصله‌ی ازدحامی: پیچیدگی این بخش مربوط به مرتب‌سازی راه‌حل‌ها با توجه به مقدار هر تابع هدف است. با توجه به این‌که H مرتب‌سازی مستقل (مربوط به هر تابع هدف) نیاز است، در بدترین مورد وقتی‌که همه‌ی راه‌حل‌ها در یک جبهه باشند، مرتب‌سازی دارای پیچیدگی $O(HPlogP)$ است.

۴) پیچیدگی مربوط به اپراتور انتخاب: پیچیدگی این بخش مربوط به فرآیند انتخاب در مراحل مختلف الگوریتم است و برابر است با $O(2Plog(2P))$.

بنابراین پیچیدگی کلی الگوریتم $O(HP^2)$ است که مربوط به بخش مرتب‌سازی نامغلوب الگوریتم می‌باشد.

۵- ارزیابی عملکرد روش ارائه‌شده

به منظور ایجاد محیط شبیه‌سازی و توپولوژی شبکه، در این مقاله، از زبان برنامه‌نویسی پایتون و کتابخانه‌ی SimPy [۴۸] استفاده شده‌است. در ادامه، تنظیمات شبیه‌سازی، معیارهای ارزیابی و الگوریتم‌های مبنا جهت مقایسه شرح داده خواهند شد. شایان ذکر است که پارامترهای شبیه‌سازی بر اساس مقاله‌ی [۴۹] تنظیم شده‌است.

۵-۱- تنظیمات شبیه‌سازی

به منظور ایجاد توپولوژی شبکه‌ی شهر هوشمند، ما یک گراف بدون جهت با ۲۰ گره‌ی IoT، یک گره‌ی زمان‌بند، ۱۲ گره‌ی مه و ۳ مرکز داده‌ی ابری *ایجاد* کرده‌ایم. در این مقاله، گره‌های مه و مراکز داده‌ی ابری به طور ناهمگن و با قابلیت‌های محاسباتی مختلفی در نظر گرفته شده‌اند و فرکانس CPU گره‌های مه و مراکز داده‌ی ابری مطابق با دستگاه‌های شرکت سیسکو تنظیم شده‌است. بنابراین، ظرفیت پردازشی گره‌های مه و مراکز داده‌ی ابری به ترتیب در بازه‌ی $[0.2, 2.4] \times 10^9$ هرتز و $[4, 6] \times 10^9$ هرتز در نظر گرفته شده‌است. سرعت انتشار بین همه‌ی گره‌ها به 3×10^8 تنظیم شده‌است. علاوه‌براین، فاصله‌ی بین دستگاه‌های IoT و گره‌ی زمان‌بند، گره‌ی زمان‌بند به گره‌های مه و گره‌های مه به مراکز داده‌ی ابری، به ترتیب، ۷۰ متر، ۱ کیلومتر و ۱۶ کیلومتر در نظر گرفته شده‌اند. ارتباطات بین دستگاه‌های IoT و گره‌ی زمان‌بند به صورت بیسیم و براساس پروتکل ارتباطات IEEE 802.11n و سایر ارتباطات شبکه بر اساس فیبر نوری فرض شده‌اند. ازاین‌رو، نرخ انتقال بین دستگاه‌های IoT و گره‌ی زمان‌بند، ۶۰۰ مگابیت بر ثانیه و بین زمان‌بند و گره‌های مه و همچنین بین گره‌های مه و مراکز داده‌ی ابری، به ترتیب، ۱ و ۴ گیگابیت بر ثانیه فرض شده‌است. همچنین، توان محاسباتی گره‌های مه در محدوده‌ی ۰/۹ تا ۱/۲۳ وات و توان محاسباتی مراکز داده‌ی

ابری در محدوده‌ی ۱/۶تا ۱/۸ وات تنظیم شده‌است. به‌علاوه، توان انتقال بین دستگاه‌های IoT و گره‌ی زمان‌بند، گره‌ی زمان‌بند و گره‌های مه و گره‌ی مه و مراکز داده‌ی ابری به ترتیب، ۰/۱، ۰/۲۵ و ۱/۲ وات در نظر گرفته شده‌است.

در این مقاله، وظایف به طور مصنوعی تولید می‌شوند و نرخ تولید داده، ۰/۳ ثانیه تنظیم شده‌است. علاوه‌براین، به منظور انجام آزمایشات و ارزیابی عملکرد روش پیشنهادی، وظایف در سایزهای مختلفی تولید شده‌اند. چرخه‌ی CPU مورد نیاز برای اجرای یک بیت وظیفه، ۲۹۷/۶۲ در نظر گرفته شده‌است. علاوه‌براین، نیازمندی‌های محاسباتی و مهلت زمانی وظایف، به طور تصادفی به ترتیب در محدوده‌ی $10^6 \times [7.375, 12.5]$ هرتز و [500, 3800] میلی‌ثانیه، تعیین شده‌است. در این مقاله، سایز داده‌ی ورودی و خروجی، یکسان فرض شده‌است. همچنین، تعداد ۱۲۰۰ وظیفه توسط گره‌های IoT به منظور ارزیابی روش پیشنهادی تولید می‌شوند. گره‌ی زمان‌بند، تابع زمان‌بند را پس از دریافت هر ۲۰۰ وظیفه اجرا می‌کند و وظایف را به گره‌های محاسباتی تعیین شده ارسال می‌کند. لازم به ذکر است که در این مقاله، وظایف به عنوان کوچکترین واحدهای پردازش در نظر گرفته شده‌اند، بنابراین نمی‌توانند به چندین زیر وظیفه جهت پردازش بر روی گره‌های محاسباتی مختلف، تقسیم شوند. هنگامی‌که یک وظیفه به یک گره‌ی محاسباتی ارسال می‌شود، درصورتی‌که گره‌ی محاسباتی بیکار[87] باشد، وظیفه را پردازش خواهد کرد، در غیر این صورت، وظیفه در صف گره‌ی محاسباتی تا زمان در دسترس شدن گره، منتظر خواهد ماند.

۵-۲- الگوریتم‌های مبنا جهت مقایسه

الگوریتم زمان‌بندی اولویت مبتنی بر مهلت زمانی[88]: در این الگوریتم [۵۰]، به هر وظیفه متناسب با مهلت زمانیش یک اولویت داده می‌شود و وظایف با توجه به اولویتشان، به ترتیب به گره‌های محاسباتی اختصاص داده می‌شوند.

الگوریتم زمان‌بندی مبتنی بر انتخاب تصادفی[89]: در این الگوریتم [۵۱]، هر وظیفه‌ی ورودی به صورت تصادفی و با احتمال یکسان به گره‌های محاسباتی مختلف اختصاص داده می‌شود.

الگوریتم زمان‌بندی ژنتیک[90]: در این طرح زمان‌بندی [۱۵]، ما اهداف و قیود یکسانی با طرح زمان‌بندی پیشنهادیمان تعریف کرده‌ایم با این حال برای وظایف، با توجه به مهلت زمانیشان، اولویتی تعریف نشده‌است. با توجه به اینکه، الگوریتم ژنتیک، یک الگوریتم تک هدفه است، مجموع وزندار دو هدف به عنوان تابع هدف مسأله در نظر گرفته شده‌است و از رابطه‌ی (۱۹) استفاده شده‌است:

$$Minimize\ \beta_1 \times SRT + \beta_2 \times SEC \qquad (۱۹)$$

در رابطه‌ی (۱۹)، β_1 و β_2 فاکتور وزن در نظر گرفته می‌شوند و مقدار آن‌ها بسته به میزان اهمیت هر تابع هدف تعیین می‌شود. ما اهمیت توابع هدف را مساوی و برابر با ۰/۵در نظر گرفته‌ایم.

الگوریتم زمان‌بندی تکامل تفاضلی باینری[91]: در این طرح زمان‌بندی [۵۲]، اهداف و قیود یکسانی با طرح زمان‌بندی پیشنهادی استفاده شده‌است و از رابطه (۱۶) برای تبدیل دو هدف مسأله به یک هدف استفاده شده‌است.

الگوریتم زمان‌بندی ترکیب ژنتیک و PSO: در این طرح زمان‌بندی [۳۰]، به منظور بهبود قابلیت جستجوی الگوریتم PSO از اپراتورهای الگوریتم ژنتیک استفاده شده‌است.

الگوریتم زمان‌بندی NSGA-II: در این طرح زمان‌بندی [۳۷] از الگوریتم NSGA-II برای زمان‌بندی وظایف استفاده شده‌است که متفاوت با روش پیشنهادی در این مقاله، جمعیت اولیه به صورت تصادفی مقداردهی می شود.

۵-۳- معیارهای عملکرد

برای ارزیابی عملکرد الگوریتم زمان‌بندی پیشنهادی از معیارهای میانگین تاخیر پاسخ سرویس، میانگین درصد وظایفی که مهلت زمانیشان را از دست می‌دهند، میانگین زمان انتظار، میانگین تاخیر اجرای وظیفه، میانگین مصرف انرژی استفاده شده‌است.

در ادامه، نتایج حاصل از آزمایش‌های انجام شده برای ارزیابی عملکرد الگوریتم زمان‌بندی پیشنهادی ارائه خواهد شد. همه‌ی آزمایشات در ۵ تکرار انجام شده‌است و میانگین مقادیر به دست آمده، در نمودارها نمایش داده شده‌است.

۵-۴- تنظیم پارامترهای الگوریتم NSGA-II

برطبق [۳۷]، چهار پارامتر تاثیرگذار در کیفیت راه‌حل‌های به دست آمده توسط الگوریتم‌های تکاملی عبارتند از: سایز جمعیت، حداکثر تعداد تکرار، احتمال تقاطع و احتمال جهش. این پارامترها، کیفیت راه‌حل‌های به دست آمده را تحت تاثیر قرار می‌دهند. بنابراین، باید تاثیر این پارامترها بر عملکرد الگوریتم پیشنهادی بررسی شود و به منظور دستیابی به راه‌حل‌های مناسب‌تر، بهترین مقدار این پارامترها باید تعیین شود. بنابراین، در این بخش، در ابتدا آزمایشاتی برای تنظیم این پارامترها انجام شده‌است. بعد از به دست آوردن مناسب‌ترین مقدار برای هر پارامتر، این مقدار در تمام آزمایشات بعدی ثابت نگه داشته شده‌است. نتایج مربوط به آزمایشات در جدول ۲ نمایش داده شده‌است و نتیجه‌ی بهترین مقدار برای پارامتر موردنظر به صورت پررنگ نشان داده شده‌است. با توجه به اینکه، الگوریتم NSGA-II، یک الگوریتم تکاملی چند هدفه است، مجموعه‌ای از راه‌حل‌های نامغلوب را به عنوان خروجی ارائه می‌دهد.

بنابراین، برای یافتن بهترین پاسخ از بین راه‌حل‌های نامغلوب، مجموع وزندار مقادیر دو تابع هدف محاسبه شده است و راه‌حل نامغلوب با کمترین مقدار مجموع وزندار به عنوان راه‌حل نهایی در نظر گرفته شده‌است.

همانطور که در جدول ۲ مشخص است، افزایش سایز جمعیت منجر به کاهش مجموع وزندار توابع هدف می‌شود. بااین‌حال، برای سایز جمعیت بزرگتر از ۵۰ مقدار بهبود حاصل‌شده بسیار کم است، در حالی‌که، زمان اجرا به طور شدیدی افزایش می‌یابد. بنابراین، سایز جمعیت، ۵۰ تنظیم شده‌است. علاوه‌براین، با توجه به آزمایشات انجام‌شده، پس از تکرار ۲۵، در مقدار توابع هدف بهبودی حاصل نمی‌شود، ازاین‌رو، حداکثر تعداد تکرار الگوریتم، ۲۵ تنظیم شده‌است. لازم به ذکر است که مقدار مجموع وزندار توابع هدف در جدول ۲ برای تابع هدف تاخیر سرویس با واحد زمانی ثانیه و برای مصرف انرژی با واحد ژول محاسبه شده‌است.

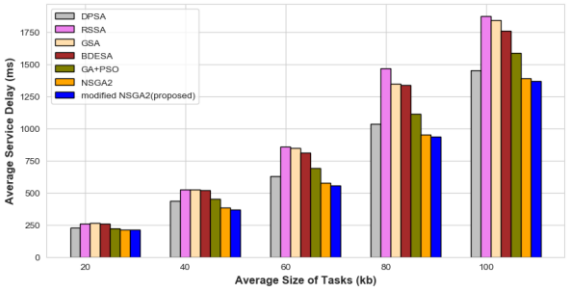
جدول ۲: تنظیم پارامترهای NSGA-II.

parameters	values				
P_s	Tested values	۳۰	**۵۰**	۷۰	۱۳۰
	Weighted sum of objectives	۰/۲۵۸	**۰/۲۳۳**	۰/۲۳۲	۰/۲۳۰
	Run time(s)	۲۰/۸۹	۳۵/۰۶	۴۹/۰۴	۹۰/۶۳
P_c	Tested values	**۱**	۰/۹۷	۰/۹۵	۰/۹۰
	Weighted sum of objectives	**۰/۲۲۸**	۰/۲۳۱	۰/۲۳۰	۰/۲۴۴
P_m	Tested values	۰/۰۳	۰/۰۵	۰/۱	**۰/۱۵**
	Weighted sum of objectives	۰/۳۰۸	۰/۲۹۳	۰/۲۴۴	**۰/۲۲۸**

۵-۵- نتایج آزمایشات مقایسه‌ی عملکرد الگوریتم پیشنهادی

در این بخش، به مقایسه‌ی نتایج الگوریتم پیشنهادی با الگوریتم‌های مبنا که در بخش ۵-۲ توضیح داده شد، پرداخته‌ایم. لازم به ذکر است که نتایج نمایش داده شده در نمودارها، حاصل از محاسبه‌ی میانگین معیارهای ارزیابی برای ۱۲۰۰ وظیفه‌ی ورودی و ۵ تکرار می‌باشد.

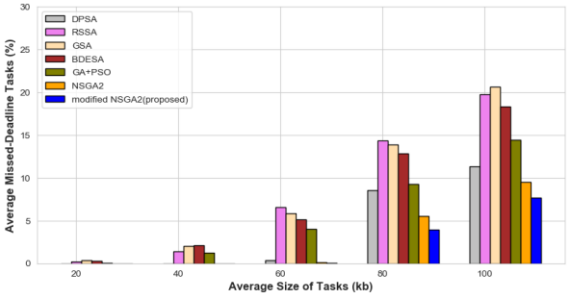
به منظور بررسی تاثیر سایز مختلف وظایف بر روی عملکرد الگوریتم پیشنهادی، وظایفی با سایزهای مختلف ۲۰، ۴۰، ۶۰، ۸۰ و ۱۰۰ کیلوبایت، تولید شده‌است. نتایج مربوط به تاثیر افزایش سایز داده‌ی ورودی بر روی میانگین تاخیر پاسخ سرویس در شکل ۴، نشان داده شده‌است. همانطور که در شکل ۴ مشخص است، افزایش سایز داده‌ی ورودی منجر به افزایش تاخیر سرویس در همه‌ی روش‌ها شده‌است. بااین‌حال، روش پیشنهادی در سایزهای داده‌ی مختلف دارای عملکرد بهتری در مقایسه با سایر روش‌ها می‌باشد. دلیل این امر آن است که با توجه به اینکه یکی از اهداف در نظر گرفته شده در روش پیشنهادی، حداقل‌سازی تاخیر پاسخ سرویس می‌باشد، بنابراین، تابع زمان‌بند با تمرکز بر این هدف، منجر به کاهش میزان این پارامتر در مقایسه با سایر روش‌ها شده‌است. همچنین، از آنجا که الگوریتم NSGA-II یک الگوریتم تکاملی چند هدفه است، به جای یک راه‌حل، مجموعه‌ای از راه‌حل‌های بهینه را به دست می‌آورد و نیازی به تنظیم وزن برای اهداف مختلف نیست و سپس با توجه به اولویت کاربر برای تاخیر یا مصرف انرژی کمتر، یکی از این راه‌حل‌ها به عنوان راه‌حل بهینه انتخاب می‌شود. از طرفی، در روش پیشنهادی، برای راه‌حل‌هایی که قید مهلت زمانی را برآورده نمی‌کنند، جریمه‌ای در نظر گرفته شده‌است، در نتیجه شانس باقی ماندن آن‌ها در نسل‌های بعد کاهش می‌یابد و منجر به راه‌حل بهتر با تاخیر پاسخ کمتر می‌شود. از سوی دیگر، استفاده از روش نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد برای مقداردهی اولیه‌ی جمعیت منجر به تنوع بیشتر در راه‌حل‌ها و همگرا شدن به راه‌حل با مقدار کمتر تاخیر می‌شود.



شکل ۴: میانگین تاخیر پاسخ سرویس در مقابل سایز داده‌ی مختلف.

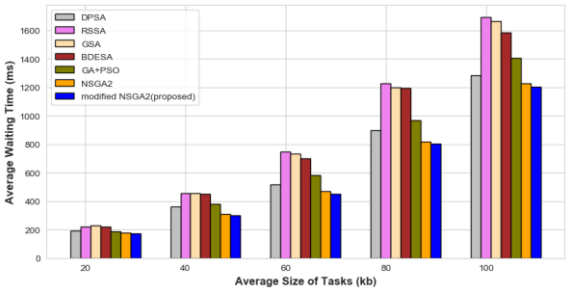
در شکل ۵، درصد وظایفی که در مهلت زمانی مورد نظر پردازش نمی‌شوند، با توجه به سایزهای مختلف وظایف نمایش داده شده‌است. همانطور که در شکل ۵ مشاهده می‌شود، با افزایش سایز داده‌ی ورودی، درصد وظایفی که مهلت زمانیشان را از دست می‌دهند، افزایش یافته است. بااین‌حال، الگوریتم پیشنهادی ما، کمترین درصد وظایفی که مهلت زمانیشان را از دست می‌دهند در مقایسه با سایر الگوریتم‌ها دارد. علت این امر آن است که در الگوریتم پیشنهادی برای وظایف مختلف با توجه به مهلت زمانیشان، اولویت‌هایی تعیین شده‌است. بنابراین، وظایف با اولویت‌های بالاتر قبل از وظایف با اولویت کمتر پردازش خواهند شد. علاوه‌براین، در روش پیشنهادی با کاهش تاخیر پاسخ سرویس، وظایف بیشتری در مهلت زمانی در نظر گرفته شده، پردازش می‌شوند. از طرفی، همانطور که قبلاً گفته شد، در روش پیشنهادی برای راه‌حل‌هایی که

قید مهلت زمانی را برآورده نمی‌کنند، جریمه در نظر گرفته شده‌است. بنابراین، این راه‌حل‌ها دارای مقدار هزینه‌ی بیشتری خواهند بود و شانس کمتری برای انتخاب شدن به عنوان راه‌حل بهینه را خواهند داشت.



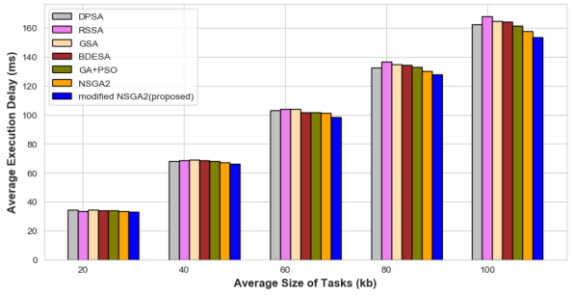
شکل ۵: میانگین درصد وظایفی که مهلت زمانیشان را از دست می‌دهند در مقابل سایز داده‌ی مختلف.

شکل ۶ میانگین زمان انتظار وظایف با توجه به سایز داده‌ی ورودی مختلف را برای چهار الگوریتم مورد بررسی در این مقاله نشان می‌دهد. همانطور که در شکل مشخص است، کمترین میانگین زمان انتظار متعلق به الگوریتم پیشنهادی است. با توجه به اینکه، در روش پیشنهادی تاخیر انتظار به عنوان بخشی از یکی از توابع هدف در نظر گرفته شده‌است، تابع زمان‌بند وظایف را به طور مناسبی در میان گره‌های محاسباتی که دارای بار کمتری هستند توزیع می‌کند که منجر به کاهش تاخیر انتظار وظایف می‌شود.



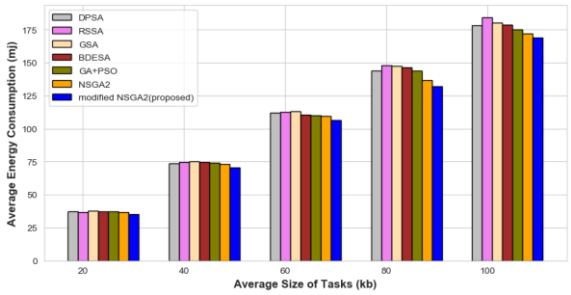
شکل ۶: میانگین زمان انتظار وظایف در مقابل سایز داده‌ی مختلف.

در شکل ۷، مقایسه‌ی عملکرد الگوریتم پیشنهادی با سایر الگوریتم‌ها از لحاظ میانگین تاخیر اجرای وظایف نمایش داده شده‌است. همانطور که ملاحظه می‌شود، کمترین تاخیر اجرای وظایف مربوط به الگوریتم پیشنهادی می‌باشد. با توجه به اینکه تاخیر پردازش وظایف، بخشی از یکی از توابع هدف این مقاله است، تابع زمان‌بند با اختصاص وظایف به گره‌های محاسباتی با ظرفیت پردازشی بیشتر منجر به کاهش تاخیر اجرای وظایف می‌شود.



شکل ۷: میانگین تاخیر اجرای وظایف در مقابل سایز داده‌ی مختلف.

در شکل ۸، عملکرد الگوریتم پیشنهادی این مقاله از لحاظ میانگین مصرف انرژی در مقابل سایز داده‌ی مختلف در مقایسه با الگوریتم‌های مبنا نمایش داده شده‌است. همانطور که در شکل ملاحظه می‌شود، الگوریتم پیشنهادی دارای کمترین میزان مصرف انرژی در مقایسه با سایر الگوریتم‌ها می‌باشد. با توجه به اینکه یکی از اهداف در این مقاله، کاهش مصرف انرژی سیستم در نظر گرفته شده‌است، زمان‌بند برای کاهش مصرف انرژی تلاش می‌کند. در نتیجه با تخصیص مناسب وظایف به گره‌های محاسباتی باعث کاهش مصرف انرژی برای پردازش و انتقال وظایف ورودی می‌گردد. از طرفی، روش پیشنهادی با تمرکز بر دو هدف در نظر گرفته‌شده، منجر به موازنه بین تاخیر سرویس و مصرف انرژی سیستم می‌شود.



شکل ۸: میانگین مصرف انرژی در مقابل سایز داده‌ی مختلف.

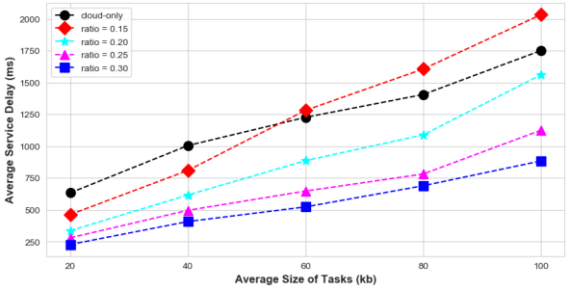
۵-۶- مقایسه‌ی پارادایم ابر و مه

در این بخش، آزمایشاتی برای ارزیابی تاخیر پاسخ سرویس توسط منابعی که فقط ویژگی‌های ابر و محاسبات مه را دارند، انجام شده‌است. به طور کلی، منابع ابر دارای قابلیت‌های پردازشی بالایی هستند، بااین‌حال، تاخیر انتقال این منابع بالا می‌باشد. در مقابل، منابع مه، قابلیت‌های پردازشی محدودی دارند، بااین‌حال، با توجه به این‌که این منابع نزدیک به کاربران نهایی قرار دارند، دارای تاخیر انتقال کمی هستند. هدف از انجام این آزمایشات این است که نشان دهیم، ابر و مه نمی‌توانند به عنوان جایگزین یکدیگر در نظر گرفته شوند و مکمل هم هستند. همچنین قصد داریم بررسی کنیم که قرار دادن منابع بسیار قدرتمند در ابر مفیدتر است یا قرار دادن

منابعی در مه با قابلیت‌های پردازشی کمتر در عین‌حال نزدیک به دستگاه‌های کاربر نهایی.

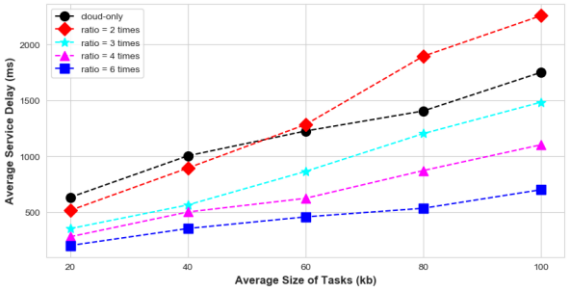
این آزمایش‌ها با در نظر گرفتن دو پارامتر نسبت قابلیت پردازشی مه به ابر و همچنین نسبت تعداد منابع مه به ابر انجام شده‌است. در هر آزمایش، یکی از پارامترها ثابت نگه داشته شده و با تغییر پارامتر دیگر، تاثیر آن بر تاخیر پاسخ سرویس، بررسی شده‌است و سپس تاخیری که با تغییر این پارامترها به دست می‌آید با سیستمی که فقط منبع ابر را در نظر می‌گیرد مقایسه شده‌است. برای سیستم با فقط منابع ابر، ۳ مرکز داده‌ی ابری با قابلیت پردازشی ۶×۱۰^{۹} هرتز در نظر گرفته شده‌است. لازم به ذکر است که، به منظور ارزیابی تاخیر سرویس، ۵۰۰ وظیفه با نرخ تولید ۰/۳ ثانیه و سایز میانگین مختلف تولید شده‌است.

در شکل ۹، تاثیر نسبت قابلیت پردازشی بر روی تاخیر سرویس ارزیابی شده‌است. در این آزمایش، تعداد گره‌های مه، ۴ برابر تعداد مراکز داده‌ی ابری و نسبت قابلیت پردازشی گره‌های مه به مراکز داده‌ی ابری متغیر و با مقادیر ۰/۱۵، ۰/۲۰، ۰/۲۵ و ۰/۳۰ در نظر گرفته شده‌است. همان‌طور که در شکل مشاهده می‌شود، کاهش نسبت قابلیت‌های منابع مه به ابر، منجر به افزایش تاخیر سرویس می‌شود تا زمانی که به نقطه‌ای می‌رسد که در آن، تاخیر مه بیشتر از تاخیر ابر است. در این مورد می‌توان نتیجه‌گیری کرد که اگرچه منابع مه در مقایسه با ابر به کاربران نهایی نزدیکتر هستند، در صورتی‌که قابلیت پردازشی کمی داشته باشند، منجر به تاخیر بالایی می‌شوند.



شکل ۹: تاخیر پاسخ سرویس مه در مقایسه با ابر با تغییر قابلیت پردازشی منابع مه.

شکل ۱۰، تاثیر نسبت تعداد گره‌های مه بر روی تاخیر سرویس را نشان می‌دهد. در این آزمایش، قابلیت‌های پردازشی گره‌های مه، ۰/۲۵ قابلیت‌های پردازشی مراکز داده‌ی ابری تنظیم شده‌است و تعداد منابع مه نسبت به ابر از ۲، ۳، ۴، ۶ برابر تغییر می‌کند. همان‌طور که در شکل مشخص است، با کاهش تعداد منابع، تاخیر پاسخ سرویس افزایش می‌یابد، تا زمانی‌که به نقطه‌ای می‌رسد که در آن، تاخیر مه بیشتر از ابر است. در این مورد می‌توان نتیجه‌گیری کرد که اگرچه منابع مه دارای قابلیت پردازشی مناسب و نزدیک به کاربران باشند، بااین‌حال، در صورتی‌که تعداد این منابع کم باشد، تاثیر منفی بر روی تاخیر سرویس و کیفیت تجربه‌ی کاربران خواهد گذاشت.



شکل ۱۰: تاخیر سرویس در مقایسه با ابر با تغییر تعداد گره‌ها در لایه‌ی مه.

۵-۷- بحث و تحلیل نتایج

در بخش قبلی به ارزیابی روش پیشنهادی پرداخته شد، همان‌طور که در نمودارها نشان داده شده‌است، روش پیشنهادی در مقایسه با سایر الگوریتم‌های زمان‌بندی عملکرد بهتری از لحاظ میانگین تاخیر پاسخ سرویس، میانگین درصد وظایفی که مهلت زمانیشان را از دست می‌دهند، میانگین زمان انتظار، میانگین تاخیر اجرای وظیفه و میانگین مصرف انرژی دارد. علت این امر این است که از آن‌جا که در مدل زمان‌بندی پیشنهادی اهداف در نظر گرفته شده تاخیر سرویس و مصرف انرژی سیستم است، زمان‌بند تلاش می‌کند تا با تخصیص مناسب وظایف به گره‌های محاسباتی، تاخیر سرویس و مصرف انرژی سیستم را کاهش دهد. در مدل زمان‌بندی پیشنهادی، تاخیر سرویس از چهار بخش تاخیر انتظار، تاخیر انتقال، تاخیر انتشار و تاخیر اجرای وظیفه تشکیل شده‌است. بنابراین زمان‌بند با تمرکز بر حداقل کردن این موارد منجر به کاهش تاخیر سرویس می‌شود. از طرفی، با توجه به این واقعیت که ما اولویت‌های مختلفی برای وظایف بر اساس مهلت زمانیشان در نظر گرفتیم، وظایفی که اولویت بالاتری دارند، قبل از وظایف با اولویت کمتر پردازش خواهند شد. همچنین وظایف با توجه به اولویتشان به گره‌های محاسباتی اختصاص داده می‌شوند. بنابراین، الگوریتم پیشنهادی وظایف با اولویت بالا را به گره‌های مه اختصاص می‌دهد و وظایف با اولویت کمتر به مراکز داده‌ی ابری ارسال می‌شوند که باعث کاهش درصد وظایفی می‌شود که مهلت زمانیشان را از دست می‌دهند.

همچنین در روش پیشنهادی به منظور مقداردهی اولیه‌ی جمعیت از ترکیب روش‌های نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد استفاده شده‌است که باعث افزایش تنوع در راه‌حل‌ها، بهبود قابلیت اکتشاف الگوریتم و قابلیت جستجوی سراسری، اجتناب از بهینه‌ی محلی و بهبود سرعت همگرایی و در نتیجه یافتن راه‌حل بهتر با

حداقل‌سازی تاخیر سرویس و مصرف انرژی سیستم می‌شود. در نهایت با آزمایشاتی که در بخش ۵-۶ انجام دادیم، اهمیت یکپارچه کردن رایانش مه در رایانش ابر را نشان دادیم. این آزمایشات نشان داد که تعداد مناسب گره‌های مه و قابلیت پردازشی مناسب آن‌ها، منجر به کاهش تاخیر سرویس و بهبود تجربه کاربر می‌شود.

از محدودیت‌ها و معایب مدل زمان‌بند پیشنهادی می‌توان به این نکته اشاره کرد که در حال حاضر تحرک کاربران در نظر گرفته نشده‌است. علاوه‌براین، یکی از معایب الگوریتم NSGA-II سرعت همگرایی کند آن است که در این کار با استفاده از روش پیشنهادی برای تولید جمعیت اولیه تا حدودی بهبود داده شد. بااین‌حال، می‌توان با در نظر گرفتن اپراتورهای تقاطع و جهش بهتر به نتایج مطلوب‌تری دست یافت. در جدول ۳ نتایج مربوط به الگوریتم‌های زمان‌بندی مختلف تحت معیارهای ارزیابی مختلف نشان داده شده‌است.

جدول ۳: مقایسه عملکرد الگوریتم‌های زمان‌بندی مختلف

معیار ارزیابی	سایز وظایف (KB)					
	روش	۲۰	۴۰	۶۰	۸۰	۱۰۰
تاخیر پاسخ سرویس (ms)	DPSA	۲۲۷/۴	۴۳۵/۰	۶۲۶/۸	۱۰۳۴/۶	۱۴۵۲/۴
	RSSA	۲۵۷/۰	۵۲۶/۷	۸۵۷/۶	۱۴۷۰/۵	۱۸۷۳/۵
	GSA	۲۶۵/۶	۵۲۶/۳	۸۴۶/۳	۱۳۴۷/۴	۱۸۴۳/۸
	BDESA	۲۵۶/۸	۵۲۱/۵	۸۱۰/۳	۱۳۳۹/۳	۱۷۶۱/۱
	GA+PSO	۲۲۴/۷	۴۵۲/۴	۶۸۹/۷	۱۱۱۲/۶	۱۵۸۵/۴
	NSGAII	۲۱۴/۷	۳۸۲/۲	۵۷۶/۷	۹۵۳/۳	۱۳۹۰/۳
	Proposed	**۲۱۰/۲**	**۳۷۰/۴**	**۵۵۴/۸**	**۹۳۸/۷**	**۱۳۶۹/۶**
درصد وظایف با مهلت از دست رفته (٪)	DPSA	۰	۰	۰/۴۰	۸/۵۳	۱۱/۳۵
	RSSA	۰/۱۹	۱/۳۷	۶/۵۷	۱۴/۳۸	۱۹/۷۶
	GSA	۰/۳۴	۲/۰۴	۵/۸۷	۱۳/۸۹	۲۰/۶۰
	BDESA	۰/۲۸	۲/۱۱	۵/۱۰	۱۲/۸۶	۱۸/۲۹
	GA+PSO	۰/۰۹	۱/۲۳	۴/۰۲	۹/۲۴	۱۴/۴۲
	NSGAII	۰	۰	۰/۱۵	۵/۵۱	۹/۵۴
	Proposed	**۰**	**۰**	**۰/۰۹**	**۳/۹۸**	**۷/۶۵**
زمان انتظار (ms)	DPSA	۱۹۱/۸	۳۶۲/۶	۵۱۸/۵	۸۹۶/۵	۱۲۸۱/۷
	RSSA	۲۲۰/۴	۴۵۳/۴	۷۴۸/۴	۱۲۲۶/۳	۱۶۹۲/۳
	GSA	۲۲۸/۶	۴۵۳/۱	۷۳۵/۰	۱۱۹۹/۹	۱۶۶۶/۳
	BDESA	۲۲۰/۷	۴۴۸/۷	۷۰۱/۴	۱۱۹۴/۱	۱۵۸۴/۰
	GA+PSO	۱۸۸/۶	۳۷۹/۶	۵۸۰/۸	۹۶۸/۰	۱۴۰۸/۳
	NSGAII	۱۷۸/۷	۳۱۰/۹	۴۷۱/۱	۸۱۶/۶	۱۲۲۵/۹
	Proposed	**۱۷۴/۲**	**۲۹۹/۱**	**۴۴۹/۲**	**۸۰۲/۰**	**۱۲۰۵/۱**
تاخیر اجرا (ms)	DPSA	۳۴/۳	۶۸/۲	۱۰۳/۰	۱۳۲/۶	۱۶۲/۴
	RSSA	۳۳/۵	۶۸/۶	۱۰۳/۹	۱۳۶/۹	۱۶۷/۹
	GSA	۳۴/۵	۶۹/۰	۱۰۳/۸	۱۳۴/۹	۱۶۴/۹
	BDESA	۳۴/۱	۶۸/۵	۱۰۱/۸	۱۳۴/۲	۱۶۴/۴
	GA+PSO	۳۳/۹	۶۸/۰	۱۰۱/۵	۱۳۴/۲	۱۶۴/۴
	NSGAII	۳۳/۴	۶۷/۱	۱۰۱/۰	۱۳۰/۳	۱۵۷/۹
	Proposed	**۳۳/۰**	**۶۶/۰**	**۹۸/۵**	**۱۲۷/۸**	**۱۵۳/۶**
مصرف انرژی (mj)	DPSA	۳۷/۱	۷۳/۷	۱۱۲/۰	۱۴۴/۱	۱۷۷/۹
	RSSA	۳۶/۶	۷۴/۷	۱۱۲/۷	۱۴۷/۹	۱۸۴/۲
	GSA	۳۷/۶	۷۵/۰	۱۱۲/۸	۱۴۷/۲	۱۸۰/۲
	BDESA	۳۷/۱	۷۴/۷	۱۱۰/۷	۱۴۶/۲	۱۷۸/۵
	GA+PSO	۳۷/۰	۷۴/۳	۱۰۹/۹	۱۴۴/۱	۱۷۵/۳
	NSGAII	۳۶/۵	۷۳/۰	۱۰۹/۵	۱۳۶/۹	۱۷۲/۳
	Proposed	**۳۵/۰**	**۷۰/۰**	**۱۰۶/۴**	**۱۳۲/۲**	**۱۶۹/۱**

نتیجه‌گیری و کارهای آتی

در این مقاله، یک مدل زمان‌بندی وظایف برای برنامه‌های کاربردی شهر هوشمند در محیط ابر-مه ارائه شد. با توجه به اینکه بسیاری از برنامه‌های کاربردی شهر هوشمند حساس به تاخیر هستند، کاهش تاخیر ارائه‌ی خدمات نقش مهمی در بهبود تجربه‌ی کاربر بازی می‌کند. از طرفی، ارائه‌دهندگان خدمات مه و ابر به دنبال کاهش مصرف انرژی سیستم می‌باشند. بنابراین، در این پژوهش بر روی این دو فاکتور به عنوان اهداف اصلی تمرکز شده و کاهش تاخیر ارائه‌ی خدمات و مصرف انرژی سیستم با در نظر گرفتن قید مهلت زمانی به عنوان توابع هدف انتخاب شده‌اند. به منظور حل این مسأله و یافتن راه‌حل مناسب، الگوریتم NSGA-II به کار گرفته شده‌است که در آن به منظور بهبود تنوع در راه‌حل‌ها از ترکیب روش نگاشت بی‌نظمی و یادگیری مبتنی بر تضاد برای مقداردهی اولیه جمعیت استفاده کردیم. از آن‌جا که، الگوریتم NSGA-II یک الگوریتم تکاملی چندهدفه است می‌تواند منجر به ایجاد موازنه بین اهداف مختلف مسأله شود. از طرفی، به منظور برآورده کردن قید مهلت زمانی، از یک رویکرد مبتنی بر جریمه استفاده شده‌است. پس از تعیین پارامترهای الگوریتم NSGA-II، به منظور ارزیابی عملکرد، آن را با شش الگوریتم زمان‌بندی DPSA، RSSA، BDESA، GSA، GA+PSO و NSGA-II (ورژن استاندارد) مقایسه نمودیم. نتایج حاصل از آزمایشات نشان می‌دهد که روش پیشنهادی با تخصیص مناسب وظایف به گره‌های محاسباتی منجر به کاهش تاخیر سرویس و مصرف انرژی می‌شود. از طرفی، از آنجا که به وظایف با توجه به مهلت زمانیشان، اولویت‌هایی اختصاص داده شده‌است، روش پیشنهادی می‌تواند منجر به کاهش درصد وظایفی شود که مهلت زمانیشان را از دست می‌دهند. در نهایت، آزمایشاتی برای نشان دادن اهمیت یکپارچه کردن رایانش مه در رایانش ابر، انجام شد. رایانش مه در مقایسه با ابر دارای تاخیر انتقال کمی است و با توجه به معماری توزیع‌شده، قادر است منطقه‌ی جغرافیایی گسترده‌ای را پوشش دهد. بنابراین، به‌کارگیری آن در رایانش ابر منجر به تاخیر سرویس کمتر و تجربه‌ی بهتر کاربران می‌شود. در کار آینده، هدف ما مطالعه‌ی بهینه‌سازی اهداف دیگر از جمله افزایش استفاده‌ی

منبع و کاهش هزینه خواهد بود. همچنین به مطالعه‌ی استفاده از الگوریتم‌های هیبریدی جهت حل مسأله‌ی زمان‌بندی وظایف خواهیم پرداخت. یکی از معایب الگوریتم NSGA-II، همگرایی کند آن است. از طرفی، الگوریتم PSO، توانایی همگرایی سریع را دارا می‌باشد. بنابراین، در کار آینده، به بهبود عملکرد NSGA-II با ترکیب آن با مزیت PSO خواهیم پرداخت.

مراجع

[1] Tang, B., Chen, Z., Hefferman, G., Pei, S., Wei, T., He, H. and Yang, Q., "Incorporating intelligence in fog computing for big data analysis in smart cities" *IEEE Transactions on Industrial informatics*, *13*(5), pp.2140-2150, 2017.

[2] Dong, Y., Guo, S., Liu, J. and Yang, Y., "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city" *IEEE Internet of Things Journal*, *6*(5), pp.7543-7554, 2019.

[3] Fiandrino, C., Anjomshoa, F., Kantarci, B., Kliazovich, D., Bouvry, P. and Matthews, J.N., "Sociability-driven framework for data acquisition in mobile crowdsensing over fog computing platforms for smart cities" *IEEE Transactions on Sustainable Computing*, *2*(4), pp.345-358, 2017.

[4] Liu, Q., Wei, Y., Leng, S. and Chen, Y., October. "Task scheduling in fog enabled Internet of Things for smart cities" In *2017 IEEE 17th International Conference on Communication Technology (ICCT)* (pp. 975-980), IEEE, 2017.

[5] Desikan, K.S., Kotagi, V.J. and Murthy, C.S.R., September. "Smart at right price: A cost efficient topology construction for fog computing enabled iot networks in smart cities" In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* (pp. 1-7), IEEE, 2018.

[6] Liao, S., Dong, M., Ota, K., Wu, J., Li, J. and Ye, T., December. "Vehicle mobility-based geographical migration of fog resource for satellite-enabled smart cities" In *2018 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE, 2018.

[7] Deng, R., Lu, R., Lai, C., Luan, T.H. and Liang, H. "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption" *IEEE internet of things journal*, *3*(6), pp.1171-1181, 2016.

[8] Mahmud, R., Kotagiri, R. and Buyya, R., "Fog computing: A taxonomy, survey and future directions" In *Internet of everything* (pp. 103-130). Springer, Singapore, 2018.

[9] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J. and Jue, J.P., "All one needs to know about fog computing and related edge computing paradigms: A complete survey" *Journal of Systems Architecture*, *98*, pp.289-330, 2019.

[10] Stojmenovic, I. and Wen, S., "The fog computing paradigm: Scenarios and security issues" In *2014 federated conference on computer science and information systems* (pp. 1-8). IEEE, 2014.

[11] Bittencourt, L.F., Diaz-Montes, J., Buyya, R., Rana, O.F. and Parashar, M., "Mobility-aware application scheduling in fog computing" *IEEE Cloud Computing*, *4*(2), pp.26-35, 2017.

[12] Pham, X.Q. and Huh, E.N., "Towards task scheduling in a cloud-fog computing system" In *2016 18th Asia-Pacific network operations and management symposium (APNOMS)* (pp. 1-4). IEEE, 2016.

[13] Jennings, B. and Stadler, R., "Resource management in clouds: Survey and research challenges" *Journal of Network and Systems Management*, *23*(3), pp.567-619, 2015.

[14] Basu, S., Karuppiah, M., Selvakumar, K., Li, K.C., Islam, S.H., Hassan, M.M. and Bhuiyan, M.Z.A., "An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment" *Future Generation Computer Systems*, *88*, pp.254-261, 2018.

[15] Aburukba, R.O., AliKarrar, M., Landolsi, T. and El-Fakih, K., "Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing" *Future Generation Computer Systems*, *111*, pp.539-551, 2020.

[16] Xu, J., Hao, Z., Zhang, R. and Sun, X., "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling" *IEEE Access*, *7*, pp.116218-116226, 2019.

[17] Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., Chung, H.S.H. and Li, Y., "Cloud computing resource scheduling and a survey of its evolutionary approaches" *ACM Computing Surveys (CSUR)*, *47*(4), pp.1-33, 2015.

[18] Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N. and Zhang, Q., "Multiobjective evolutionary algorithms: A survey of the state of the art" *Swarm and Evolutionary Computation*, *1*(1), pp.32-49, 2011.

[19] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T., "A fast and elitist multiobjective genetic algorithm: NSGA-II" *IEEE transactions on evolutionary computation*, *6*(2), pp.182-197, 2002.

[20] Xu, X., Liu, X., Xu, Z., Dai, F., Zhang, X. and Qi, L., "Trust-oriented IoT service placement for smart cities in edge computing" *IEEE Internet of Things Journal*, *7*(5), pp.4084-4091, 2019.

[21] Xu, X., Huang, Q., Yin, X., Abbasi, M., Khosravi, M.R. and Qi, L., "Intelligent offloading for collaborative smart city services in edge computing" *IEEE Internet of Things Journal*, *7*(9), pp.7919-7927, 2020.

[22] Naranjo, P.G.V., Pooranian, Z., Shojafar, M., Conti, M. and Buyya, R., "FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments" *Journal of Parallel and Distributed Computing*, *132*, pp.274-283, 2019.

[23] Tang, C., Wei, X., Zhu, C., Wang, Y. and Jia, W., "Mobile vehicles as fog nodes for latency optimization in smart cities" *IEEE Transactions on Vehicular Technology*, *69*(9), pp.9364-9375, 2020.

[24] Hosseinioun, P., Kheirabadi, M., Tabbakh, S.R.K. and Ghaemi, R., "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm" *Journal of Parallel and Distributed Computing*, *143*, pp.88-96, 2020.

[25] Hosseinzadeh, M., Masdari, M., Rahmani, A.M., Mohammadi, M., Aldalwie, A.H.M., Majeed, M.K. and Karim, S.H.T., "Improved Butterfly Optimization Algorithm for Data Placement and Scheduling in Edge Computing Environments" *Journal of Grid Computing*, *19*(2), pp.1-27, 2021.

[26] Tanha, M., Hosseini Shirvani, M. and Rahmani, A.M., 2021. A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. *Neural Computing and Applications*, *33*(24), pp.16951-16984.

[27] Ghanavati, S., Abawajy, J.H. and Izadi, D., "An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment" *IEEE Transactions on Services Computing*, 2020.

[28] Bitam, S., Zeadally, S. and Mellouk, A., "Fog computing job scheduling optimization based on bees swarm" *Enterprise Information Systems*, *12*(4), pp.373-397, 2018.

[29] Kumar, A.S. and Venkatesan, M., "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment" *Wireless Personal Communications*, *107*(4), pp.1835-1848, 2019.

[30] Shahryari, O.K., Pedram, H., Khajehvand, V. and TakhtFooladi, M.D., "Energy and task completion time trade-off for task offloading in fog-enabled IoT networks" *Pervasive and Mobile Computing*, p.101395, 2021.

[31] Huang, T., Lin, W., Xiong, C., Pan, R. and Huang, J., 2020. An ant colony optimization-based multiobjective service replicas placement strategy for fog computing. *IEEE Transactions on Cybernetics*.

[32] Memari, P., Mohammadi, S.S., Jolai, F. and Tavakkoli-Moghaddam, R., 2022. A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. *The Journal of Supercomputing*, *78*(1), pp.93-122.

[33] Zade, B.M.H., Mansouri, N. and Javidi, M.M., 2021. SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment. *Expert Systems with Applications*, *176*, p.114915.

[34] Dubey, K. and Sharma, S.C., 2021. A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. *Sustainable Computing: Informatics and Systems*, *32*, p.100605.

[35] Abd Elaziz, M., Abualigah, L. and Attiya, I., 2021. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Generation Computer Systems*.
[36] Nguyen, B.M., Thi Thanh Binh, H. and Do Son, B., "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment", *Applied Sciences*, *9*(9), p.1730, 2019.
[37] Ali, I.M., Sallam, K.M., Moustafa, N., Chakraborty, R., Ryan, M.J. and Choo, K.K.R., "An Automated Task Scheduling Model using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems" *IEEE Transactions on Cloud Computing*, 2020.
[38] Tsegaye, A. and Assefa, B.G., 2021, November. HSSIW: Hybrid Squirrel Search and Invasive Weed Based Cost-Makespan Task Scheduling for Fog-Cloud Environment. In *2021 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)* (pp. 160-165). IEEE.
[39] Wang, Y., Lang, P., Tian, D., Zhou, J., Duan, X., Cao, Y. and Zhao, D., "A game-based computation offloading method in vehicular multiaccess edge computing networks" *IEEE Internet of Things Journal*, *7*(6), pp.4987-4996, 2020.
[40] Han, S., Xu, X., Fang, S., Sun, Y., Cao, Y., Tao, X. and Zhang, P., "Energy efficient secure computation offloading in NOMA-based mMTC networks for IoT" *IEEE Internet of Things Journal*, *6*(3), pp.5674-5690, 2019.
[41] Guerrero, C., Lera, I. and Juiz, C. Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. *Future Generation Computer Systems*, *97*, pp.131-144, 2019.
[42] Li, Q., Liu, S.Y. and Yang, X.S. "Influence of initialization on the performance of metaheuristic optimizers". *Applied Soft Computing*, *91*, p.106193, 2020.
[43] Truong, Khoa H., et al. "A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems." *Applied Soft Computing* 77,567-583, (2019).
[44] Salimi, R., Motameni, H. and Omranpour, H., "Task scheduling using NSGA II with fuzzy adaptive operators for computational grids" *Journal of Parallel and Distributed Computing*, *74*(5), pp.2333-2350, 2014.
[45] Sivanandam, S.N. and Deepa, S.N., "Genetic algorithms" In *Introduction to genetic algorithms* (pp. 15-37). Springer, Berlin, Heidelberg, 2008.
[46] Du, K.L. and Swamy, M.N.S., "Particle swarm optimization" In *Search and optimization by metaheuristics* (pp. 153-173). Birkhäuser, Cham, 2016.
[47] Safe, M., Carballido, J., Ponzoni, I. and Brignole, N., September. "On stopping criteria for genetic algorithms" In *Brazilian Symposium on Artificial Intelligence* (pp. 405-413). Springer, Berlin, Heidelberg, 2004.
[48] T. SimPy, Simpy: Discrete event simulation for python, Tech. Rep. 9, 2017, URL https://simpy.readthedocs.io/en/latest/.
[49] Gazori, Pegah, Dadmehr Rahbari, and Mohsen Nickray. "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach." Future Generation Computer Systems 110, pp.1098-1115, 2020.
[50] Rajput, I.S. and Gupta, D., "A priority based round robin CPU scheduling algorithm for real time systems" *International Journal of Innovations in Engineering and Technology*, *1*(3), pp.1-11, 2012.
[51] Zhu, Q., Si, B., Yang, F. and Ma, Y., "Task offloading decision in fog computing system" *China Communications*, *14*(11), pp.59-68, 2017.
[52] Wang, L., Fu, X., Mao, Y., Menhas, M.I. and Fei, M., 2012. A novel modified binary differential evolution algorithm and its applications. *Neurocomputing*, *98*, pp.55-75.

پاورقی‌ها:

[1] Urbanization
[2] Internet of Things (IoT)
[3] Computing tasks
[4] Big data
[5] Sensing layer
[6] Paradigm
[7] Fog computing
[8] Gateways
[9] Access points
[10] Task scheduling
[11] Load balancing
[12] Exact
[13] Heuristic
[14] Near-optimal
[15] Population-based evolutionary algorithm
[16] Pareto set
[17] Non-dominated sorting genetic algorithm II (NSGA-II)
[18] Non-dominated sorting mechanism
[19] Crowding distance
[20] Elitist
[21] Crossover
[22] Mutation
[23] Chaotic map
[24] Opposition based learning
[25] Penalty function
[26] Cooperative
[27] Workload
[28] Service placement
[29] Edge computing
[30] Strength Pareto Evolutionary Algorithm II (SPEAII)
[31] Computation offloading
[32] Road Side Unit (RSU)
[33] Integer Linear Programming (ILP)
[34] Invasive Weed Optimization (IWO)
[35] Cultural Evolution Algorithm (CEA)
[36] Mobile Edge Computing (MEC)
[37] Butterfly Optimization Algorithm (BOA)
[38] Simulated Annealing (SA)
[39] Ant Mating Optimization (AMO)
[40] Bees Life Algorithm (BLA)
[41] Hybrid Genetic Algorithm-Ant Colony Optimization (HGA-ACO)
[42] Mixed-Integer Non-Linear Programming (MINLP)
[43] Particle Swarm Optimization (PSO)
[44] Tabu Search Algorithm (TSA)
[45] Fruit Fly Optimization (FOA)
[46] Approximate Nearest Neighbor (ANN)
[47] Squirrel Search Algorithm (SSA)
[48] Chemical Reaction Partial Swarm Optimization
[49] Laxity-based Priority Algorithm (LBPA)
[50] Ant Colony Optimization(ACO)
[51] Static
[52] Artificial ecosystem based optimization (AEO)
[53] Salp Swarm Algorithm (SSA)
[54] Individual
[55] Invasive weed

[56] Processing Delay
[57] Transmission Delay
[58] Propagation Delay
[59] Queuing Delay
[60] Dominance
[61] Non-dominated
[62] population
[63] front
[64] density
[65] parents
[66] offsprings
[67] Population size
[68] Fast Non-Dominated Sorting
[69] ranking
[70] Logistic map
[71] Iterative map
[72] Generation
[73] Fast non-dominated sorting mechanism
[74] integer
[75] set
[76] diversity
[77] Diversity-preservation
[78] Euclidian distance
[79] selection
[80] Random selection
[81] Tournament selection
[82] Roulette Wheel Selection
[83] Exploration operator
[84] Single point crossover
[85] Double point crossover
[86] Uniform crossover
[87] Idle
[88] Deadline-based Priority Scheduling Algorithm (DPSA)
[89] Random Selection Scheduling Algorithm (RSSA)
[90] Genetic Scheduling Algorithm (GSA)
[91] Binary Differential Evolution scheduling Algorithm (BDESA)