DOR :20.1001.1.2538161.1402.8.2.1.5

تابستان ۱۴۰۲، دوره ۸، شماره ۲، پیاپی ۲۹

دریافت مقاله: ۱۴۰۲/۰۲/۲۷

پذیرش مقاله: ۱۴۰۲/۰۵/۰۹

نوع مقاله: پژوهشی

ارائه رویکردی به‌منظور زمان‌بندی منابع در محیط‌های بدون سرویس‌دهنده

سنا رایحی

دانشکده کامپیوتر- دانشگاه علم و صنعت- تهران- ایران

پست الکترونیکی: sana_rayehi@comp.iust.ac.ir

مهرداد آشتیانی*

دانشکده کامپیوتر- دانشگاه علم و صنعت- تهران- ایران

پست الکترونیکی: m_ashtiani@iust.ac.ir

چکیده

با توجه به پیشرفت فناوری و نیاز روزافزون کاربران، تولیدکنندگان باید برنامه‌های کاربردی خود را به‌سرعت توسعه دهند. محاسبات ابری، با توجه به مزایایی که دارد توجه تولیدکنندگان را برای پیاده‌سازی برنامه‌های کاربردی به خود جلب کرده است. ابر، مدل‌های سرویس‌دهی مختلفی مانند زیرساخت به‌عنوان سرویس، بستر به‌عنوان سرویس، نرم‌افزار به‌عنوان سرویس و تابع به‌عنوان سرویس را دارد. در این پژوهش تمرکز اصلی بر روی مدل سرویس‌دهی تابع به‌عنوان سرویس است. این مدل کاربران را قادر می‌سازد تا توابع ابری را بر روی بستری از منابع اجرا کنند بدون این که نگرانی درباره مدیریت زیرساخت آن داشته باشند. این کار هزینه کمتری برای آن‌ها خواهد داشت. یکی از مهم‌ترین چالش‌های این حوزه، مسئله زمان‌بندی توابع است. ارائه‌دهندگان سرویس، از الگوریتم‌های زمان‌بندی برای نگاشت درخواست‌های ورودی خود، به منابع محاسباتی استفاده می‌کنند. این نگاشت باید از جنبه‌های مختلفی که بر عملکرد سیستم تأثیر دارند، بهینه باشد. زمان‌بند، وظیفهٔ مدیریت منابع را بر عهده دارد. اگرچه راه‌حل‌های مختلفی برای مدیریت تأمین منابع ارائه شده است، اما برای مدیریت مؤثرتر منابع نیاز به روش‌های جدیدتری هست. بر این اساس، این کار پژوهشی یک الگوریتم ترکیبی جدید را برای بهبود عملکرد تأمین منابع ارائه می‌کند. در راه‌حل پیشنهادی، یک الگوریتم ترکیبی را بر اساس الگوریتم‌های ژنتیک توسعه داده شده و شبیه‌سازی تبرید معرفی می‌کنیم. در این روش با استفاده از ترکیب دو الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید، از مزایای هر دو الگوریتم استفاده کردیم. این دو الگوریتم مکمل یکدیگر شده و نتایج مقایسه رهیافت پیشنهادی با روش‌های دیگر نشان‌دهنده عملکرد مثبت این روش در کاهش زمان تکمیل درخواست‌ها است. به‌منظور پیاده‌سازی و بررسی عملکرد این روش، یک موتور شبیه‌سازی توسعه داده شده است. طراحی و پیاده‌سازی رهیافت پیشنهادی با استفاده از زبان متلب انجام شده است. محیط انجام آزمایش شامل یک مرکز داده و تعدادی میزبان هست. برای روش ارائه شده معیارهای ارزیابی مختلفی نیز ارائه شده و رهیافت پیشنهادی به دو صورت پیاده‌سازی شده است. یکی به این صورت که الگوریتم تبرید برای تک‌به‌تک نمونه جمعیت تولید شده توسط الگوریتم ژنتیک توسعه داده‌شده، اجرا شود. روش دیگر که نتایج بهتری به همراه دارد به این

* نویسندهٔ مسئول

صورت است که الگوریتم تبرید تنها بر روی بهترین نمونه از جمعیت اجرا شود. نتایج تجربی نشان می‌دهند رهیافت پیشنهادی در مقایسه با پیاده‌سازی‌های غیر اکتشافی ۷۰درصد، در مقایسه با الگوریتم ژنتیک ۳۰ درصد و همچنین در مقایسه با الگوریتم تبرید ۳۰ درصد زمان تکمیل اجرای درخواست کمتری دارد و درنتیجه عملکرد بهتری دارد.

واژه‌های کلیدی: رایانش بدون سرویس‌دهنده، تابع به‌عنوان سرویس، زمان‌بندی، الگوریتم ترکیبی.

۱. مقدمه

در زمینه رایانش ابری، بدون سرویس‌دهنده معمولاً به مدل‌هایی اطلاق می‌شود که تولیدکنندگان نیازی به نگرانی در مورد استقرار سرویس‌دهنده ندارند. بااین‌حال، در سال‌های اخیر، اصطلاح محاسبات بدون سرویس‌دهنده بیشتر به راه‌حل‌هایی اشاره دارد که حول توابع متمرکز شده‌اند و نحوه انجام مقیاس‌بندی را پنهان می‌کنند. محاسبات بدون سرویس‌دهنده با مفاهیم محاسبات ابری سنتی متفاوت است. به این معنا که زیرساخت و بن‌سازه‌هایی که سرویس‌ها در آن اجرا می‌شوند از مشتریان پنهان هستند [۱]. یکی از مزیت‌های مهم محاسبات بدون سرویس‌دهنده این است که بار هماهنگی جریان کار را به ارائه‌دهنده خدمات منتقل می‌کند. علاوه بر این، عملکردهای بدون سرویس‌دهنده با مزایای مالی همراه هستند. زیرا کاربران فقط برای زمان استفاده از منابع واقعی (یعنی زمان صرف شده برای پردازش درخواست‌های کاربر) هزینه می‌کنند. چالشی که در این چشم‌انداز محاسباتی متصور هستیم، افزایش هزینه زیرساخت موردنیاز برای مدیریت ترافیک رو به رشد بدون سرویس‌دهنده است. مشکل خاصی که در این مقاله در نظر می‌گیریم حول محور زمان‌بندی است. چگونه باید عملکردهای کاربر برنامه‌ریزی شود تا هزینه‌های ارائه‌دهنده در مقیاس به حداقل برسد و درعین‌حال تأخیرهای قابل‌قبولی ارائه شود. باید توجه داشت که پرداختن به این چالش هم نیازمند قرار دادن کارآمد بارکاری ورودی برای به حداقل رساندن هزینه‌های سرمایه‌ای ارائه‌دهنده است. مفهوم اصلی پشت مدل اجرای بدون سرویس‌دهنده، تغییر پیچیدگی‌های مدیریت منابع برنامه از تولیدکننده به ارائه‌دهنده ابر است. مدل بدون سرویس‌دهنده به ارائه‌دهنده نیاز دارد تا تخصیص منابع به توابع را در زمان واقعی مدیریت کند. برخلاف یک سناریوی قرار دادن سرویس تحت یک مدل سرویس‌دهنده، که در آن کاربر قبل از اجرای برنامه، محیط را با منابع موردنیاز پیکربندی می‌کند. بنابراین، هر سیاست تخصیص منبع دلخواه می‌تواند منجر به اختلاف منابع بعدی برای برنامه‌ها، در طول زمان اجرا شود که منجر به نقض قرارداد سطح سرویس برای کاربر می‌شود[۲]. ازآنجایی‌که ارائه‌دهنده ابر مسئول مدیریت منابع در بن‌سازه‌های بدون سرویس‌دهنده است، اهداف کاربر در تجزیه‌وتحلیل بدون سرویس‌دهنده با استقرار سرویس‌دهنده محور متفاوت است. به‌ویژه، درحالی‌که به حداقل رساندن هزینه اجرا و زمان تکمیل کار[1] هنوز یک هدف اصلی است، معیارهایی مانند استفاده از منبع و جداسازی اکنون بر عهده ارائه‌دهنده ابر است [٤]. از آنجایی که بن‌سازه‌های تابع به‌عنوان سرویس[2] تخصیص منابع محاسباتی را در بین مشاغل مدیریت می‌کنند. ارائه‌دهندگان دیگر نیازی به اهداف مرسوم به حداکثر رساندن استفاده از منابع خوشه‌ای و اجرای عدالت در بین مشاغل از طریق سیاست‌های زمان‌بندی بین شغلی ندارند. در عوض، تحت مدل صورت‌حساب[3] یعنی تابع به‌عنوان سرویس، ارائه‌دهندگان باید اکنون هزینه اجرای هر کار را که متناسب با زمان‌های اجرا جمع‌آوری شده در سراسر وظایف جزء آن است، در نظر بگیرند. این امر نیاز به سیاست‌های زمان‌بندی بین وظایف را برای کارهای تحلیلی بدون سرویس‌دهنده نشان می‌دهد. به‌منظور زمان‌بندی کارآمد و مؤثر وظایف یک گردش کار در محیط‌های رایانش ابری، برنامه‌ریزان برنامه‌ها دارای

1- Job Completion Time
2- Function as a Service(FaaS)
3- Billing model

خط‌مشی‌های مختلف کیفیت خدمات[4] هستند که شامل به حداقل رساندن کل زمان اجرا، به حداقل رساندن هزینه کل اجرا[5]، رعایت محدودیت‌های مهلت زمانی[6] وظایف هست. الگوریتم پیشنهادی بر روی یک رویکرد ترکیبی تمرکز دارد. رهیافت پیشنهاد شده، از یک روش ترکیبی استفاده می‌کند که ترکیبی از ژنتیک[7] توسعه داده شده و شبیه‌سازی تبرید[8] به دو شیوه است. در مسئله مورد بررسی، یک گردش کار از وظایف با وابستگی‌های از قبل ناشناخته وجود دارد. در مورد هریک، اندازه متوسط ورودی و خروجی آشکار است. مجموعه‌ای از ماشین‌های مجازی[9] ابری رزرو شده و درخواستی وجود دارد و هزینه استفاده از آن‌ها مشخص است. الگوریتم به‌گونه‌ای است که هزینه به حداقل برسد. نوآوری و دستاوردهای این کار پژوهش شامل مواردی مانند زیر است:

۱. بهینه‌سازی سراسری و استفاده بهینه از منابع.
۲. بهره‌مندی از الگوریتم‌های اکتشافی.
۳. پویایی مضاعف در مراحل مختلف الگوریتم.
٤. کاهش یافتن زمان تکمیل درخواست‌ها.

استفاده از الگوریتم زمان‌بندی ترکیبی، اهدافی همچون رعایت توافق‌نامه سطح خدمات (میان کاربر و ارائه‌دهنده سرویس)، تأمین کیفیت خدمات، کمینه کردن زمان اجرا و به‌طورکلی بهینه کردن برنامه از هر نظر را، بدون اتلاف منبع و هزینهٔ زیاد، به‌صورت پویا محقق می‌کند. در بخش دوم به معرفی و توضیح ادبیات و مفاهیم اولیه مورد استفاده در این کار پژوهشی مانند ادبیات رایانش بدون سرویس‌دهنده پرداخته می‌شود. در بخش سوم تعدادی از کارها و پژوهش‌های اخیر انجام شده در حوزه زمان‌بندی منابع معرفی می‌شوند و نقاط مثبت و منفی هر کدام توضیح داده می‌شود. در بخش چهارم، جزئیات رهیافت پیشنهادی تشریح می‌شود و الگوریتم لازم ارائه می‌شود. در بخش پنجم، ارزیابی رهیافت پیشنهادی در مقایسه با روش‌های دیگر انجام می‌شود. در بخش ششم، به جمع‌بندی و نتیجه‌گیری از پژوهش انجام شده و ارائه پیشنهاد برای کارهای آینده، پرداخته می‌شود.

۲. مفاهیم اولیه

در بخش قبل، اهمیت تخصیص مناسب منابع برای رسیدن به کیفیت بالا در برنامه‌های کاربردی و به تعریف مسئله زمان‌بندی پرداخته شد. در این بخش، ادبیات و مفاهیم پایه‌ای مرتبط با پژوهش انجام شده، بیان می‌شود.

۱,۲- رایانش ابری

رایانش ابری یک اصطلاح کلی برای هر نوع خدمتی است که شامل ارائه خدمات میزبانی شده از طریق اینترنت است. به‌عبارتی‌دیگر، رایانش ابری، در دسترس بودن منابع سیستم رایانه‌ای، به‌ویژه ذخیره داده و قدرت محاسبات، بدون مدیریت مستقیم کاربران است. رایانش ابری با ارائه منابع محاسبات خود به‌عنوان خدمت، از درگیر کردن کاربران با موضوعاتی همچون خرید، مالکیت و نگه‌داری از مراکز داده و سرویس‌دهنده‌های فیزیکی جلوگیری می‌کند که این موضوع باعث کاهش هزینه‌های کاربران می‌شود [۵]. در رایانش ابری، محاسبات و پردازش‌ها با سرویس‌دهنده‌هایی که با یکدیگر شبکه هستند، انجام می‌شود و ذخیره‌سازی داده‌ها به‌صورت متمرکز و دسترسی به سرویس‌ها و منابع، به‌صورت بر خط و از طریق اینترنت است؛ در عمل به‌جای این که اطلاعات را بر روی دیسک سخت خود نگه‌دارید و یا برنامه‌های کاربردی موردنیازتان را به‌طور مستمر به‌روزرسانی کنید، از سرویسی بر روی اینترنت برای برآوردن نیازهای خود استفاده می‌کنید[٦]. برای دسترسی به منابع پردازشی که در حجم زیادی فراهم است، از مجازی‌سازی استفاده می‌شود، که با تجمیع منابع و ایجاد یک سیستم یکپارچه انجام می‌شود. مشتری این سرویس‌های ابری، بر اساس مقدار و مدت استفاده از منابع، هزینه پرداخت می‌کند. با این

4- Quality of service (QoS)
5- execution
6- Deadline constraints
7- genetic
8- simulated annealing
9- Virtual machines(VM)

فنّاوری، هزینـــه پردازش‌ها، میزبانی برنامه‌های کاربردی، ذخیره‌ســـازی داده‌ها و ارائه سرویس به‌طور قابل‌توجهی کاهش‌یافته اســـت[۷]. یکی از مهم‌ترین ویژگی‌های رایانش ابـــری در دســـترس بودن منابع محاســـباتی متناســـب با نیازهای درخواســـتی هست. ارائه‌دهندگان خدمات رایانش ابری، خدمات خود را بر اســـاس مدل‌هـــای متفاوتی ارائه می‌کننـــد[۸]. در ادامـــه به معرفـــی انواع ســـرویس‌ها در محیط‌های رایانش ابری می‌پردازیم. زیرســـاخت به‌عنوان سرویس[۱۰]، دسترســـی به منابع اساسی مانند ماشین‌های فیزیکی، ماشـــین‌های مجـــازی و ذخیره‌ســـازی مجازی را فراهـــم می‌کنـــد. همچنین ایـــن مدل امکاناتـــی همچون ذخیره‌سازی، دیسک ماشین مجازی، ایجاد و دسترسی به شبکه محلی مجازی[۱۱]، تعادل در بار ورودی، فراهم کردن آدرس‌های IP و بســـته‌های نرم‌افـــزاری را فراهم می‌کند. مدل زیرساخت به‌عنوان ســـرویس به ارائه‌دهنده خدمات ابری این امـــکان را می‌دهد که زیرســـاخت‌ها را از طریق شـــبکه و به‌صورت مقرون‌به‌صرفه تعیین کنند. بن‌ســـازه به‌عنوان ســـرویس[۱۲] محیط اجرا را بـــرای برنامه‌ها فراهم می‌کند. همچنین ابزارهای توســـعه و اســـتقرار موردنیاز برای توســـعه برنامه‌ها را ارائه می‌دهد. بن‌سازه به‌عنوان سرویس دارای ویژگی‌هایی است که به غیر تولیدکنندگان، امکان ایجاد برنامه‌هـــای وب را می‌دهد. نرم‌افزار به‌عنوان ســـرویس[۱۳]، نرم‌افزار را به‌عنوان یک ســـرویس نهایی به کاربـــران ارائه می‌دهـــد. در حقیقت ســـرویس نهایی ارائه شده نرم‌افزاری است که در سرویس میزبان مستقر شده و از طریق اینترنت قابل‌دسترســـی است. برنامه‌های ارائه شـــده در این سرویس در دسترس هســـتند و در صورت تقاضا می‌تـــوان منابع موردنیـــاز نرم‌افزارها را افزایش و یا کاهش داد. نرم‌افزارهای ارائه شـــده به‌صورت خودکار به‌روزرســـانی شـــده و ارتقا پیدا می‌کنند. تابـــع به‌عنوان سرویس[۱۴]، این مدل کاربران را قادر می‌سازد تا توابع ابری را بر روی بســـتری از منابع اجرا کنند بدون این که نگرانی درباره مدیریت زیرســـاخت آن داشته باشند. در معماری تابع به‌عنوان ســـرویس، یک برنامه کاربردی به واحدهایی مستقل به نام تابع تجزیه می‌شوند. مهم‌ترین ویژگی مثبت مدل تابع به‌عنوان ســـرویس ویژگی مطابق با درخواست بودن آن هســـت به این معنی که زیرساخت‌های پشتیبانی تنها در حالتی که موردنیاز می‌باشـــند اســـتفاده می‌شوند و در صورت عدم اســـتفاده از آن‌ها مشـــتری هزینه‌ای را متحمل نمی‌شود[۹].

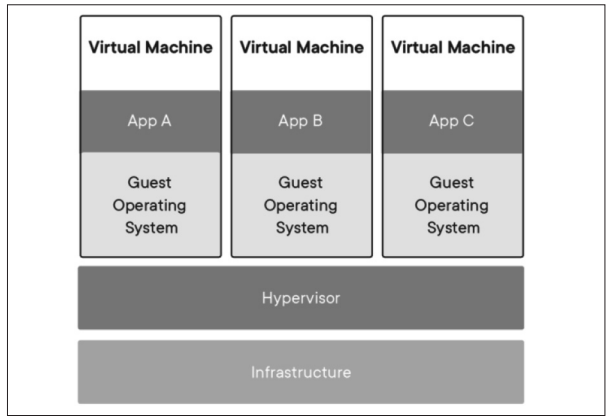
۲,۲- ماشین مجازی

ماشین مجازی یک دستگاه مجازی و نرم‌افزاری شبیه به یک کامپیوتر واقعی است که امکان اجرای سیستم‌عامل‌ها و برنامه‌های مختلف روی یک دستگاه فیزیکی مثل کامپیوتر رومیزی یا سرویس‌دهنده فیزیکی را فراهم می‌کند. ماشین مجازی یک نمونه مجازی از رایانه است که می‌تواند تقریباً تمامی عملکردهای یک کامپیوتر را ازجمله اجرای برنامه‌ها و سیســـتم‌عامل‌ها انجام دهد. ماشین‌های مجازی بر روی یک ماشـــین فیزیکی کار می‌کنند و از طریق نرم‌افزاری به نام ناظر[۱۵] به منابع محاسباتی دسترسی دارند. ناظر، منابع ماشین فیزیکی را به مجموعه‌ای منتقل می‌کند که می‌تواند در صورت نیاز تهیه و توزیع شود و چندین ماشین مجازی را قادر می‌سازد تا در یک ماشین فیزیکی واحد اجرا شوند. ناظر منابع موجود را مدیریت می‌کند و آن‌ها را به ماشین یا ماشین‌های مجازی در حال کار روی دستگاه اختصاص می‌دهـــد. در ضمـــن این نرم‌افـــزار در زمان‌بنـــدی تعیین روش اختصاص منابع بر اســـاس نحوه پیکربندی ناظر و ماشین‌های مجازی را نیز بر عهده دارد و قادر است منابع اختصاص داده شده به هر یک از ماشین‌های مجازی را بر اســـاس تغییر نیازهای آن‌ها تغییر دهد[۱۰]. شکل معماری ماشین مجازی در شکل (۱) آورده شده است.

۳,۲- کانتینر

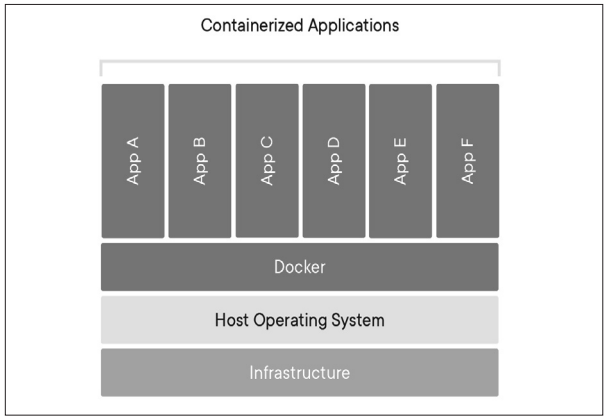
کانتینرهـــا یک انتـــزاع در لایه برنامه هســـتند که کد و

10- Infrastructure-as-a-Service(IaaS)
11- V AN
12- Platform-as-a-Service(PaaS)
13- Software-as–a-Service (SaaS)
14- Function-as-a-Service(FaaS)
15- hypervisor

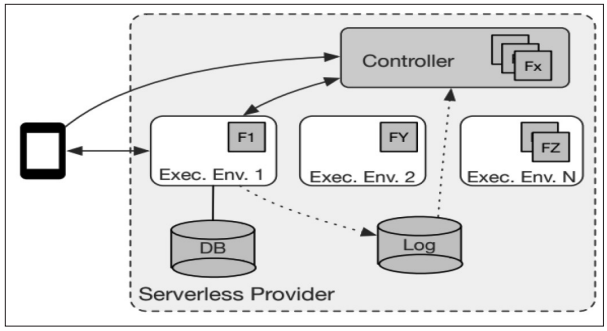


شکل (۱) : معماری ماشین‌های مجازی[۱۱].

وابستگی‌ها را باهم بسته‌بندی می‌کنند. چندین کانتینر می‌توانند روی یک ماشین اجرا شوند و هسته سیستم‌عامل را با سایر کانتینرها به اشتراک بگذارند که هرکدام به‌عنوان فرآیندهای ایزوله در فضای کاربر اجرا می‌شوند. کانتینرها نسبت به ماشین‌های مجازی فضای کمتری را اشغال می‌کنند (تصاویر کانتینر معمولاً ده‌ها مگابایت اندازه دارند)، می‌توانند برنامه‌های بیشتری را مدیریت کنند و به ماشین‌های مجازی و سیستم عامل کمتری نیاز دارند. کانتینرها و ماشین‌های مجازی مزایای جداسازی و تخصیص منابع مشابهی دارند، اما عملکرد متفاوتی دارند زیرا کانتینرها به‌جای سخت‌افزار، سیستم عامل را مجازی می‌کنند. کانتینرها قابل‌حمل و کارآمدتر هستند. شکل معماری کانتینر در شکل (۲) آورده شده است.



شکل (۲) : معماری کانتینر[۱۱].



شکل (۳) : نمونه‌ای از فراخوانی دریک ارائه‌دهنده بدون سرویس‌دهنده[۱۲].

۴،۲- رایانش بدون سرویس‌دهنده

رایانش بدون سرویس‌دهنده یک الگوی نوظهور برای اجرای توابع مشخص شده توسط کاربر بر روی منابع ارائه‌دهنده با مقیاس‌پذیری تقریباً نامحدود است[۳]. در رایانش بدون سرویس‌دهنده، کاربر مسئول نوشتن کد و بسته‌بندی آن است و ارائه‌دهنده ابر مسئول تهیه و نگهداری زیرساخت‌ها، ازجمله سرویس‌دهنده‌های میزبان، موردنیاز برای اجرای کد و برنامه کاربر است. رایانش بدون سرویس‌دهنده همچنین می‌تواند شامل چارچوب‌هایی باشد که نیازمندی‌های برنامه‌های کاربردی خاص مانند پیشنهادهای پسانه به‌عنوان سرویس[۱۶] را برآورده می‌کنند[۳]. در حالی که مجموعه برنامه‌های کاربردی پشتیبانی شده توسط محاسبات بدون سرویس‌دهنده هنوز در حال تکامل است، درجامعه محاسباتی اتفاق‌نظر وجود دارد که چندین رده از برنامه‌های کاربردی، ازجمله چارچوب‌های مبتنی بر MapReduce، در نهایت به‌طور یکپارچه بر روی بن‌سازه‌های بدون سرویس‌دهنده اجرا می‌شوند[۱۲]. نمونه‌ای از فراخوانی در یک ارائه‌دهنده بدون سرویس‌دهنده در شکل (۳) نشان داده شده است.

تولیدکنندگانی که از تابع به‌عنوان سرویس استفاده می‌کنند، در واقع به بن‌سازه دسترسی دارند که به آن‌ها اجازه می‌دهد تا کد برنامه‌های خود را اجرا کنند. مهم‌ترین ویژگی تابع به‌عنوان سرویس سادگی است. برای این که تولیدکنندگان بیشترین بهره را از تابع به‌عنوان سرویس ببرند، باید مطمئن شوند که هر ویژگی فقط یک عملیات را انجام می‌دهد. در نظر داشته باشید که دامنه اجرا باید

16- Backend-as-a-Service(BaaS)

محدود و درعین‌حال مؤثر باشد. درخواست از یک تابع برای فراخوانی یک تابع دیگر باعث کاهش سرعت برنامه در طول زمان و افزایش هزینه‌ها می‌شود [۱۳]. رایانش بدون سرویس‌دهنده دارای مزایایی است که در ادامه به آن‌ها می‌پردازیم:

۱. **هزینه پایین**: رایانش بدون سرویس‌دهنده به‌طورمعمول بسیار مقرون‌به‌صرفه هست. قابل‌ذکر است ارائه‌دهندگان خدمات سنتی مبتنی بر ابر غالباً مجبور به پرداخت هزینه‌های فضای استفاده نشده کاربر و یا زمان‌هایی که از CPU استفاده نمی‌شود، هستند.

۲. **بی‌نیاز از مدیریت سرویس‌دهنده**: با این که رایانش بدون سرویس‌دهنده در واقع در سرویس‌دهنده‌ها انجام می‌شوند، اما ارائه‌دهندگان خدمات مربوطه مجبور نیستند که با سرویس‌دهنده‌ها سروکار داشته باشند. در واقع می‌توان گفت آن‌ها توسط فروشندگان اداره می‌شوند. این امر منجر به کاهش سرمایه‌گذاری در DevOps می‌گردد که علاوه بر کاهش هزینه‌ها، به ارائه‌دهندگان خدمات این امکان را می‌دهد که برنامه‌های خود را بدون محدودیت در ظرفیت سرویس‌دهنده، ایجاد نموده و گسترش دهند.

۳. **مقیاس‌پذیری ساده**: ارائه‌دهندگان خدمات رایانش بدون سرویس‌دهنده، با استفاده از ساختار بدون سرویس‌دهنده دیگر نگران خط‌مشی‌های خود نسبت به افزایش کد نیستند. بنابراین، یک برنامهٔ بدون سرویس‌دهنده قادر به مدیریت تعداد غیرمعمول درخواست‌ها خواهد بود، درست همان‌طور که به یک درخواست واحد از یک کاربر رسیدگی می‌کند. یک برنامهٔ سنتی با مقدار ثابت فضای سرویس‌دهنده به‌محض این که با افزایش ناگهانی ترافیک روبرو شود، از کنترل خارج می‌گردد.

۴. **ساده‌سازی کد پشته**: ازآنجا که برنامه در یک سرویس‌دهنده مبدأ میزبانی نمی‌شود، کد آن از هر جایی قابل اجرا است. بنابراین بسته به فروشنده، امکان اجرای برنامه بر روی سرویس‌دهنده‌های نزدیک به کاربر وجود دارد. این امر منجر به کاهش زمان تأخیر شده زیرا، درخواست‌های کاربر مجبور نیستند کل مسیر را تا سرویس‌دهنده مبدأ طی کنند.

مانند هر راه‌حل نرم‌افزاری، رایانش بدون سرویس‌دهنده نیز دارای معایبی است. در ادامه به بررسی برخی از مهم‌ترین معایب و نقاط ضعف رایانش ابری بدون سرویس‌دهنده می‌پردازیم.

۱. محدودیت منابع: به‌طورکلی مدیریت منابع ابری به دلیل محدودیت آن‌ها چالشی است که در همه سطوح محاسبات ابری دیده می‌شود، اما در محاسبات بدون سرویس‌دهنده عدم مدیریت صحیح منابع محدود به دلیل امکان فعال شدن مکرر توابع و فرض نامحدود بودن منابع چالشی بزرگ است.

۲. تأخیر: هنگام اجرای یک عملکرد، ارائه‌دهندگان بدون سرویس‌دهنده به‌طور خودکار منابع لازم را در هر فراخوانی به کار می‌گیرند. نسبت به حجم کار، کانتینرها معمولاً در میلی‌ثانیه تهیه می‌شوند، اما حتی می‌توانند چند ثانیه طول بکشد.

۳. مدت اجرا: زمان اجرای یک عملکرد بدون سرویس‌دهنده محدود است و پس از یک دوره معین لغو می‌شود. این امر معمولاً ۵ دقیقه پس از فراخوان است، اما در بین ارائه‌دهندگان متفاوت است. محدودیت‌های اجرا یک اشکال عمده برای برنامه‌هایی است که فرآیندهای طولانی‌مدت را آغاز می‌کنند. کاهش این مشکل با تقسیم‌بندی کد به قطعات کوچک‌تر و اجرای آن‌ها به‌عنوان میکروسرویس‌ها امکان‌پذیر است.

٤. قفل فروشنده: ارائه‌دهندگان معمولاً از فناوری‌های اختصاصی برای فعال کردن سرویس‌های بدون سرویس‌دهنده خود استفاده می‌کنند. این امر ممکن است مشکلاتی را برای کاربرانی ایجاد کند که می‌خواهند تغییر حجم کار خود را به بن‌سازهٔ دیگری منتقل کنند. هنگام انتقال به ارائه‌دهنده دیگر، تغییر در کد و معماری برنامه اجتناب‌ناپذیر است. دستگاه‌های مبتنی بر تابع به‌عنوان سرویس منجر به قفل شدن فروشنده می‌شوند، یعنی هیچ

امکانی برای انتقال (سیستم یا برخی از قطعات آن) به سایر ارائه‌دهندگان ابری ندارند. این بدان معنی است که کاربران نمی‌توانند از تفاوت‌های ارائه‌دهندگان ابری استفاده کنند و مزایای هر یک را ترکیب کنند.

۲-۴-۱. معرفی بن‌سازه‌ها

در جدول (۱) برخی از معروف‌ترین ارائه‌کنندگان بن‌سازه‌های رایانش بدون سرویس‌دهنده معرفی‌شده‌اند. گفتنی است که همه‌ی این ارائه‌کنندگان، خدمات خود را به‌صورت متن‌باز ارائه نمی‌دهند و اختصاصی هستند.

۵,۲- دسته‌بندی الگوریتم‌های زمان‌بندی

زمان‌بندی به معنی انتخاب بهترین منبع برای یک وظیفه است یا اختصاص ماشین‌های مجازی به وظایف به‌گونه‌ای که زمان اجرا به حداقل برسد. به‌طورکلی در الگوریتم‌های زمان‌بندی فهرستی از وظایف وجود دارد که به ترتیب اولویت برای همهٔ وظایف ساخته شده است. وظایف بر اساس اولویت انتخاب می‌شوند و به پردازنده‌ها تخصیص داده شده که تابع تناسب آن‌ها از پیش تعیین شده است. روش‌های مختلفی برای مشکل زمان‌بندی در سیستم‌های توزیع شده وجود دارد [۱٤]. الگوریتم‌های زمان‌بندی سعی دارند تا زمان‌بندی بهینه را جهت افزایش بهره‌وری از منابع و درنتیجه افزایش توان محاسباتی، ارائه نمایند. در این بخش به بیان برخی از اصطلاحات پیرامون محاسبات ابری و مدل محاسباتی بدون سرویس‌دهنده پرداختیم [۱۵].

۳- پژوهش‌های مرتبط

تاکنون روش‌های متفاوتی در زمینه زمان‌بندی پیشنهاد شده است. این روش‌های درگذشته بیشتر با استفاده از رویکردهای قدیمی مانند، الگوریتم‌های ابتکاری پیاده‌سازی می‌شدند اما امروز بیشتر روش‌ها به رویکردهای فعال روی آورده‌اند و اغلب با استفاده از روش‌های فرا اکتشافی، یادگیری ماشین و روش‌های دیگر در زمینه هوش مصنوعی، پیاده‌سازی می‌شوند.

۱,۳- روش‌های مبتنی بر الگوریتم‌های زمان‌بندی ابتکاری

الگوریتم‌های ابتکاری معیارهایی همچون نرخ بهره‌وری استفاده از منابع، توازن بار و لزوم پاسخ‌دهی سریع به درخواست‌ها را در نظر نمی‌گیرند. برای نمونه، الگوریتم FCFS، در این الگوریتم وظایف بر اساس ترتیب ورود سرویس‌دهی می‌شوند، به‌نحوی که کارهایی که زودتر وارد می‌شوند، زودتر از سایرین برای سرویس‌دهی در اختیار ماشین‌ها قرار می‌گیرد. از این الگوریتم به‌ندرت استفاده می‌شود و یا به عنوان الگوریتم ثانویه به همراه دیگر الگوریتم‌ها استفاده می‌شود. لاکرا و همکاران [۱٦]، الگوریتم زمان‌بندی وظایف چندهدفه را طراحی کرد که فهرستی از وظایف را با توجه به اولویت‌های اختصاص داده شده تشکیل می‌داد و زمان اجرای رویکرد پیشنهادی بهتر از FCFS بود.

روش پیشنهاد شده توسط آقای سعید پارسا و رضا ملکی [۱۷]، یک رویکرد حداکثر حداقل بهبودیافته برای یافتن راه‌حل زمان‌بندی کار با در نظر گرفتن زمان تکمیل همه کارها به‌جای زمان اجرای وظیفه فعلی ارائه شد، یعنی هدف آن تغییر الگوریتم Min-Max با انتخاب منبعی بود که تکمیل کلی را به حداقل می‌رساند. در مقایسه با الگوریتم Min-Max این طرح بهتر از الگوریتم اصلی عمل کرد. الگوریتم Greedy-R در روش حریصانه رسیدن به هدف در هر گام مستقل از گام قبلی و بعدی است. یعنی در هر مرحله برای رسیدن به هدف نهایی، مستقل از این که در مراحل قبلی چه انتخاب‌هایی صورت گرفته و انتخاب فعلی ممکن است چه انتخاب‌هایی در پی داشته باشد، انتخابی که در ظاهر بهترین انتخاب ممکن است صورت می‌پذیرد. به همین دلیل است که به این روش، روش حریصانه گفته می‌شود. بنابراین، در این الگوریتم وظایف با اولین زمان اجرای سریع به قوی‌ترین منبع ابر در دسترس برای به حداکثر رساندن زمان پاسخ سیستم اختصاص می‌یابد [۱۸].

جدول (۱) : انواع بن‌سازه‌های بدون سرویس‌دهنده و ارائه‌کنندگان آن‌ها.

نام بن‌سازه	صاحب	سایت	نوع ارائه
Amazon Lambda	Amazon	https://aws.amazon.com/lambda	تجاری
Azure Functions	Microsoft	https://azure.microsoft.com/en-us/services/functions	تجاری
GCP Functions	Google	https://cloud.google.com/functions	تجاری
IBM Cloud Functions	IBM	https://www.ibm.com/uk-en/cloud/functions	تجاری
AlibabaCloud Function Compute	Alibaba	https://www.alibabacloud.com/products/function-compute	تجاری
kubeless	kubeless	https://kubeless.io	متن‌باز
Open FaaS	OpenFaaS Ltd	https://www.openfaas.com	متن‌باز
Open Whisk	Apache	https://openwhisk.apache.org	متن‌باز
Fission	Platform9	https://fission.io	متن‌باز
IronFunctions	Iron	https://open.iron.io	متن‌باز
Fn Project	Oracle	https://fnproject.io	متن‌باز
Open-lambda	OpenLambda	https://github.com/open-lambda	متن‌باز
Knative	Google	https://knative.dev	متن‌باز

۲،۳- روش‌های مبتنی بـــر الگوریتم‌های زمان‌بندی اکتشافی

الگوریتـــم بهینه‌ســـازی اجتمـــاع ذرات (PSO) بـــرای زمان‌بنـــدی برنامه کاربـــردی به منابع محاســـباتی ارائه شـــده که هم هزینه انتقـــال داده و هم هزینه محاســـبه را در نظر می‌گیرد، ایـــن الگوریتم برای برنامه‌های کاربردی جریان کاری از طریق متفـــاوت کردن هزینه‌های ارتباطی و محاســـباتی مورداستفاده قرار می‌گیرد. در این الگوریتم هدف بهبود جریان کاری اســـت، نتایج آزمایش‌ها نشـــان می‌دهد که نگاشـــت وظیفه به منبع بر اساس این الگوریتم در هزینه صرفه‌جویی می‌کند و علاوه بر این تعادل خوبی از بـــار کاری روی منابـــع محاســـباتی را از طریق توزیع وظایف به منابع در دســـترس فراهـــم می‌کند [۱۹]. به‌طور مثال در کار تحقیقاتی انجام شـــده توسط متپو و همکاران ایشـــان [۲۰]، الگوریتم بهینه‌سازی ازدحام ذرات دودویی برای زمان‌بندی وظایف معرفی شـــد. آن‌ها نسخهٔ دودویی الگوریتـــم PSO را با پیچیدگی و هزینه کم برای زمان‌بندی وظایف و به‌منظـــور حداقل کردن زمان انتظار، درجه عدم توازن و درعین‌حال به حداقل رساندن زمان و هزینه اجرا و به حداکثر رساندن استفاده از منابع، معرفی کردند.

۳،۳- روش‌های مبتنی بـــر الگوریتم‌های زمان‌بندی ترکیبی

روش‌های زمان‌بندی ترکیبی از ترکیب چندین روش با یکدیگر تولید می‌شـــوند. نقطه مثبت روش‌های ترکیبی این است که درصورتی‌که روش‌های مناسب با یکدیگر ترکیب شـــوند، می‌توانند مکمل یکدیگر باشند و هر روش ایرادات روش دیگـــر را جبران کند. به‌طـــور مثال در کار تحقیقاتی انجام شـــده توسط چاودری و همکاران [۲۱] یک الگوریتم ترکیبـــی مبتنی بر GSA بـــرای برنامه‌ریزی گردش کار دو هدفه معرفی شده است. آن‌ها یک الگوریتم فرااکتشافی برای زمان‌بندی جریان کار پیشنهاد دادند که مدت‌زمان و هزینه را در نظـــر می‌گرفت. این الگوریتم ترکیبی از الگوریتم‌های GSA و HETF بود. آن‌ها با استفاده از آزمون‌های آماری و آنالیز واریانس اثربخشی الگوریتم پیشنهادی را نسبت به الگوریتم‌های GSA و HETF نشـــان دادند به‌طوری‌که نتایج حاکی از آن بود که الگوریتم پیشـــنهادی از این الگوریتم‌ها بهتر عمل می‌کند. کار تحقیقاتی و روش پیشنهادی توسط هنتش و همکاران[۲۲]، نیز یک روش ترکیبی هســـت که به

جدول (۲): خلاصه مقایسه کارهای مرتبط بررسی شده.

مقاله	نوع	نوع منابع	معیار	تکنیک	روش	معایب
لاکرا و همکاران[۱۶]	پیشگیرانه	همگن	چند معیاره (منابع، هزینه)	تصمیم‌گیری چند معیاره	آگاه از منابع	عدم توجه به بار ورودی و منابع همگن
سعید پارسا و رضا ملکی[۱۷]	منفعلانه	ناهمگن	استفاده از پردازنده و حافظه	بر اساس درخت بیشینه-کمینه	آگاه از منابع	عدم توجه به بار ورودی و سربار محاسباتی
متپو و همکاران [۲۰]	پیشگیرانه	همگن	استفاده از پردازنده و حافظه	الگوریتم PSO	آگاه از منابع	امکان قرار گرفتن در بهینه‌های موضعی
آقای چاودری و همکاران [۲۱]	منفعلانه	همگن	زمان پاسخگویی و هزینه	الگوریتم مبتنی بر GSA	آگاه از منابع و SLA	عدم توجه به بار ورودی و سربار محاسباتی
هنتش و همکاران[۲۲]	پیشگیرانه	همگن	زمان پاسخگویی و هزینه	الگوریتم ترکیبی ذوب فلزات و جستجوی ممنوعه	آگاه از منابع و SLA	عدم توجه به بار ورودی و منابع همگن
رامان و وهاب [۲۳]	پیشگیرانه	ناهمگن	بارکاری	شبکه‌های عصبی	آگاه از هزینه / SLA	سربار محاسباتی

ارائه یک مدل ترکیبی بر اساس الگوریتم ذوب فلزات و جستجوی ممنوعه پرداخته است. در این تحقیق، یک روش ترکیبی جدید بر پایه الگوریتم‌های ذوب فلزات و لیست ممنوعه ارائه شده است که تعادل بار را در هنگام تخصیص کارهای وارد شده به سیستم و منابع برقرار می‌کند. هدف اصلی روش پیشنهادی اختصاص دادن درخواست‌های ارسالی به منابع با حفظ تعادل با در نظر گرفتن توان محاسباتی منابع و همچنین طول کار درخواستی هست. روش پیشنهادی با شبیه‌ساز CloudSim ارزیابی شده است و نتایج با نتایج روش‌های دیگر تحت شرایط یکسان مقایسه شده است. نتایج نشان می‌دهد که روش پیشنهادی کارایی در معیارهای زمان اجرای کل، هزینه اجرا را بهبود می‌دهد.

۴،۳- روش‌های مبتنی بر شبکه عصبی

رامان و وهاب [۲۳]، به محاسبه زمان‌بندی گردش کار با استفاده از شبکه عصبی پرداخته‌اند. برای اندازه‌گیری کارایی زمان‌بندی گردش کار، تعیین زمان ساخت و هزینه اجرا ضروری است. از آنجایی که برآورد زمان و هزینه در محیط ابر دشوار است، طراحی یک محاسبه کارآمد زمان‌بندی گردش کار همچنان یک چالش است. بر اساس تجربه سیستم، این مقاله الگوریتم زمان‌بندی ترکیبی شبکه عصبی پس‌پراکندگی پس‌پرسازی مبتنی بر اولویت (PBF-NN) را برای اندازه‌گیری دقیق زمان ساخت و هزینه اجرا پیشنهاد می‌کند. الگوریتم backfill برای زمان‌بندی وظایف به منابع موجود استفاده می‌شود. درصد مهاجرت با استفاده از این الگوریتم در مقایسه با الگوریتم به ترتیب ورود کاهش می‌یابد. سپس از مدل برای اندازه‌گیری عادلانه تخصیص منابع استفاده می‌شود. سیستم تخصیص مجدد کار را بر اساس ارزش عادلانه تصمیم می‌گیرد. شبکه عصبی پس‌انتشار فرآیند قرار دادن ماشین مجازی را با آموزش و آزمایش‌های لازم انجام می‌دهد. الگوریتم پیشنهادی به‌صورت پویا وظایف را تخصیص می‌دهد و استفاده از منابع را کاهش می‌دهد. خلاصه‌ای از مقایسه کارهای مرتبط بررسی شده در جدول (۲) نشان داده شده است.

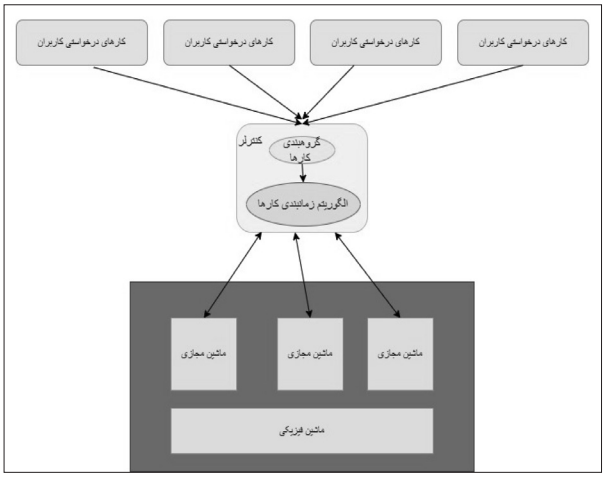
مشکل روش‌های فعال نیاز آنها به حجم زیاد داده، قدرت محاسباتی به نسبت بالا و زمان طولانی برای اجرا هست. از طرفی در بسیاری از روش‌های کنونی، الگوریتم‌هایی برای مسئله زمان‌بندی در سیستم‌های همگن ارائه شده است. سیستم‌های جدید با توجه به نیازهای جدید اکثراً ناهمگن هستند. بنابراین، در این مقاله، به مسئله زمان‌بندی کارها در سیستم‌های ناهمگن پرداخته شده است. به‌طورکلی پیدا کردن زمان‌بندی بهینه برای چنین محیطی با استفاده از روش‌های صرفاً مبتنی بر الگوریتم‌های تکاملی یا

الگوریتم‌های مبتنی بر یادگیری ماشین سربار هزینه بالایی دارد یا نمی‌تواند دقت کافی در تصمیم‌گیری را فراهم کند. از این جهت، روند تحقیقات کنونی بیشتر از پیش به سمت رویکردهای ترکیبی است.

در ایـن بخش راه‌حل‌های ارائه شـده برای زمان‌بندی منابع بر اسـاس الگوریتم‌های اکتشـافی و غیر اکتشافی بررسی شد. بررسی‌ها بیانگر این موضوع است که بیشتر تحقیقات روی الگوریتم‌های اکتشافی کار کرده‌اند. نکته‌ای کـه قابل‌توجه هسـت این اسـت که این روش‌هـا یا دقت مناسبی ندارند و یا این که ازنظر هزینه نهایی، مانند قدرت محاسباتی و یا مدت‌زمان موردنیاز برای اجرای الگوریتم صرفاً برای برخی از مسـئله‌های ساده‌تر توجیه استفاده دارند. برای نمونه روش الگوریتم اکتشـافی شبیه‌سـازی تبرید به دلیل پذیرش پاسخ‌های غیر بهبوددهنده تابع هدف، برخالف سایر روش‌های جسـت‌وجوی موضعی به نقطه شروع وابسته نیست و می‌تواند از دام بهینه‌های موضعی تا حد زیادی رهایی یابد. اما زمان اجرای الگوریتم را بهبود نمی‌دهد. روش ارائه شـده در این پژوهـش نیز مبتنی بر اسـتفاده از الگوریتم‌های اکتشافی اسـت. در این روش با استفاده از ترکیب دو الگوریتم ژنتیک پیشنهادی و الگوریتم شبیه‌سازی تبرید، از مزایای هر دو الگوریتم استفاده شده اسـت. به عبارت بهتر با ترکیب این دو روش هم از مزیت دقت بالاتر پیش‌بینی اسـتفاده شـده اسـت و هم اطمینان حاصل شده است که نتایج در دام بهینه‌های موضعی نیفتند.

۴- روش پیشنهادی

در این پژوهش، تمرکز اصلی تشخیص شرایط محیط و پیدا کردن مقدار مناسب متغیرها با توجه به شرایط حال حاضر سـامانه و در نتیجه تخصیص بهینه منابع هست. روش پیشـنهاد شـده در ابتدا تلاش می‌کند تا با بررسی برخـی از ویژگی‌های محیط، متغیرهای مـورد نیاز مانند تعداد ماشـین‌های مجازی که باید اختصاص داده شـوند را تعیین کند. رهیافت پیشـنهاد شده، از یک روش ترکیبی



شکل (۴) : معماری محیط رایانش ابری در رهیافت پیشنهادی.

اسـتفاده می‌کند که ترکیبـی از الگوریتم‌هـای ژنتیک[17] و شبیه‌سازی تبرید[18] است. محیط رایانش ابری در رهیافت پیشنهاد شده شامل قسـمت های متفاوتی است. معماری کلی محیط رایانش ابری در شکل (٤) نشان داده شده است.

۱,٤- ساختار الگوریتم ژنتیک توسعه داده‌شده

۴-۱-۱. جمعیت‌دهی اولیه

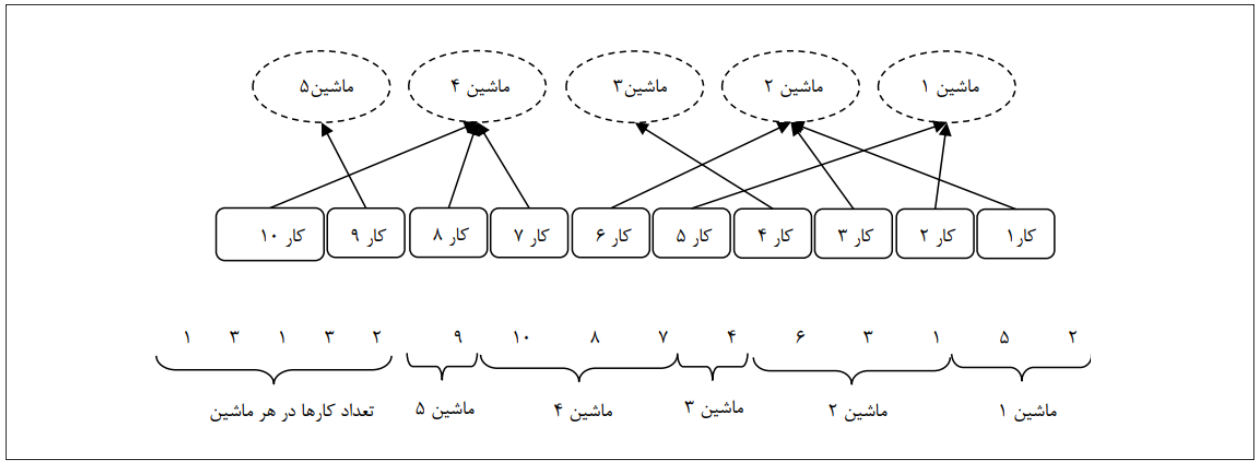
این بخش شامل دو مرحله است :

در مرحلـه اول، در الگوریتم پیشـنهادی از کدگذاری جایگشـتی استفاده شده است. تشکیل این کدگذاری شامل دو مرحله هسـت: ۱) بـرای هر ماشـین m_i، دنباله S_i از کارهایی که به آن ماشـین تخصیص داده شـده‌اند، ایجاد می‌شود، ۲) دنباله‌های کارهای ایجاد شده در مرحله اول به یکدیگر الحاق می‌شوند. نتیجه این دو مرحله، جایگشتی از کارها هست که به ماشین‌ها تخصیص داده شده‌اند. شکل (۵) مثالی از این نوع کدگذاری را برای ۱۰ کار و ۵ ماشـین نشان می‌دهد. این کدگذاری از دو بخش تشکیل شده است: ۱) تخصیص کارها و ۲) سـاختمان داده مربوط به نمایش تعداد کارهای تخصیص‌یافته به هر ماشین. در مرحله دوم، یک جمعیت اولیه تصادفی از کروموزوم‌ها تولید شـده و سپس جواب مرحله اول به این جمعیت اضافه می‌شود.

۴-۱-۲. محاسبه تابع تناسب (Fitness)

پس از انجام مرحله اول و تشـکیل جمعیت اولیه، تابع

17- genetic
18- simulated annealing



شکل (۵) : مثالی از کدگذاری جایگشتی.

تناسب برای بهینه‌سازی استفاده می‌شود. این تابع با بررسی مقدار زمان تکمیل اجرای درخواست‌ها در تمام منابع پردازشی به‌روزرسانی می‌شود. جواب انتهایی مسئله کمترین مقدار به‌دست آمده از تابع تناسب هست.

$$sum\_time = \sum_{j=1}^{n} Time(i,j) \qquad (۱-۴)$$

$\mathbf{Time(i,j)}$ مدت‌زمان اجرای درخواست j روی ماشین $\mathbf{i}$ امست. همچنین محاسبه زمان تکمیل اجرای تمام درخواست‌ها با فرمول (۴-۲) محاسبه می‌شود.

$$complete\_time = \max\left(sum_{time(i)}\right) \qquad (۲-۴)$$

درنهایت مقدار برازندگی هر جواب به‌صورت زیر محاسبه می‌گردد:

$$\text{complete\_cost} = \sum_{i=1}^{M} \text{sume\_time(i)} \times \text{Cpu(i)} \qquad (۳-۴)$$

$$\text{Cmax} = \max(\text{MCT}) \qquad (۴-۴)$$

۴-۱-۳. عملگرها

پس از محاسبه تابع تناسب، در مرحله بعدی روی جمعیت اولیه عملیات تقاطع انجام می‌شود. به این صورت که با روش چرخ رولت ۲ والد انتخاب می‌شوند و با ترکیب آن‌ها دو فرزند جدید ساخته می‌شود. این کار به تعداد مشخصی که با نرخ تقاطع از آن یاد می‌شود روی جمعیت انجام می‌شود. روی جمعیت اولیه عملیات جهش انجام می‌شود. به این صورت که به‌صورت تصادفی یکی

Algorithm 1: Developed genetic

```
BEGIN-PROCEDURE
Input: children_task = []
Output: best_population
For (parent1, parent2 in parents)
    child_task = crossover(parent1, parent2)
    For (n=1:2:ncross)
        i1 <-- find(rand<=f,1,'first')
        i2 <-- find(rand<=f,1,'first')
        [crosspop(n),crosspop(n+1)]=UniformCrossOver(pop(i1),pop(i2))
        function [y1,y2]=UniformCrossOver(x1,x2)
    END-For
    For (j=1:J)
        For ( i=L{j})
            k=find(L{j}==i)
            If (k==1)
                ST(i )<-- 0
            else
            PreviousJob=L{j}(k-1)
            ST(i) <-- FT(PreviousJob) + s(PreviousJob,i,j)
            PT(i) <-- p(i,j)
            FT(i) <--ST(i)+PT(i)
            If (not isempty(L{j})
            MCT(j)=FT(L{j}(end))
        END-For
    END-For
    cmax <-- max(MCT)
    child <-- mutate(child_task, mutation_rate)
    cost <-- cost_function(child)
    children.append(child, cost)
END-PROCEDURE
```

شکل (۶) : شبه کد الگوریتم ژنتیک توسعه داده شده.

از وظایف در یک راه‌حل جابه‌جا می‌شود. این کار به تعداد مشخصی که با نرخ جهش از آن یاد می‌شود، روی جمعیت انجام می‌شود. مقدار برازندگی تمام فرزندان تولید شده به روش‌های تقاطع و جهش محاسبه می‌شود و بهترین نسل از فرزندان بر مبنای مقدار برازندگی انتخاب می‌شوند تا به نسل بعد راه پیدا کنند. عملیات مهاجرت در هر خانواده انجام می‌شود. به این صورت که بهترین فرد هر خانواده به‌جای بدترین فرد خانواده کناری قرار می‌گیرد تا تمام خانواده‌ها به سمت بهتر شدن پیش روند. شبه کد الگوریتم ژنتیک توسعه داده شده در شکل (۶) نشان داده شده است.

۲،٤- الگوریتم تبرید در روش پیشنهادی

شروع این الگوریتم با تولید راه‌حل اولیه به‌وسیله ذوب کردن مواد همانند تولید کروموزوم‌ها ایجاد می‌شود. سپس دما در طی فرآیند جستجو کاهش می‌یابد. بنابراین در شروع جستجو احتمال پذیرش حرکت در سربالایی بیشتر است و سپس به‌تدریج کاهش می‌یابد.

۱. تعریف یک دمای اولیه برای الگوریتم.

۲. انتخاب یک نمونه به‌صورت تصادفی به‌عنوان نمونه شروع.

۳. تغییر دما به‌صورت تدریجی.

٤. تولید یک نمونه جدید به‌صورت تصادفی.

٥. ارزیابی نمونه جدید با محاسبه زمان تکمیل اجرای تمامی درخواست‌ها با استفاده از ترکیب زمان اجرای هر درخواست و شماره منبعی که آن درخواست بر روی آن انجام شده است.

٦. بررسی تغییرات امتیاز و قبول یا رد کردن نقطه جدید بر اساس احتمال قابل قبولی که برای آن تعیین می‌شود.

۷. تکرار مراحل ٤ تا ٦، تا رسیدن به یک شرط توقف مانند کاهش دمای الگوریتم به مقدار معین یا بهبود نسبی امتیاز.

در الگوریتم تبرید برای حل مسئله زمان‌بندی منابع، می‌توان از یک تابع تناسب استفاده کرد. در اینجا تابع تناسب احتمال انتقال از یک حالت فعلی، مثلاً s به حالت نامزد جدید مانند s′ با یک تابع احتمال پذیرش $P(e,e',T)$ مشخص می‌شود که در آن، $e = E(s)$ و $e' = E(s')$ است. بنابراین، هدف آن است که انرژی سیستم کمینه باشد، پس حالتی که در آن $E(s)$ کمتر باشد، بهتر از حالتی است که در آن مقدار $E(s)$ بیشتر باشد. در واقع، در صورتی‌که $E(s) \geq E(s')$ باشد، s' پذیرفته می‌شود و اگر s' بدتر از s باشد، s' با یک احتمال پذیرفته می‌شود. تابع احتمال به‌صورت زیر است:

$$p = e^{\frac{E(s')-E(s)}{\text{Temperature}}} \quad (۵-۴)$$

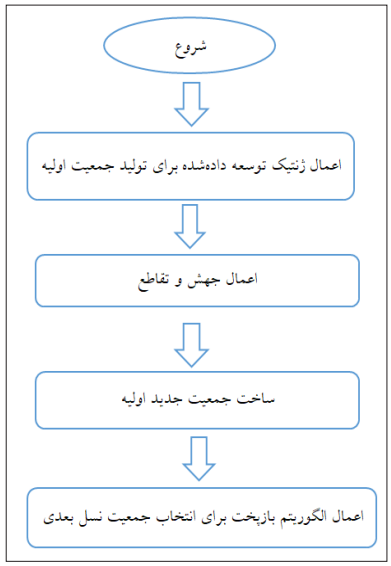
در این الگوریتم، دما در هر مرحله به یک مقدار ثابت کاهش پیدا می‌کند. این مقدار کاهش دما، ρ به‌عنوان یک پارامتر کلیدی در الگوریتم محسوب می‌شود و باید به‌گونه‌ای تعیین شود که بهبود جواب بهینه را تضمین کند.

$$\text{Temperature}_{t+1} = \rho \text{Temperature}_t \quad (۶-۴)$$

در صورت کاهش دما و میل کردن آن به سمت صفر، احتمال نیز باید کاهش پیدا کند و یا به صفر و یا یک عدد مثبت میل کند بنابراین دما در تغییرات زمان‌بندی نقش مهمی دارد. الگوریتم ابتدا در فضای بزرگی از راه‌حل‌ها، صرف‌نظر از انرژی داخلی سیستم به دنبال پاسخ می‌گردد و به‌تدریج، به سمت مناطق دارای انرژی کمتر جابه‌جا می‌شود. این منطقه، به‌تدریج کوچک‌تر می‌شود و این کار، تا زمانی که بهینه سراسری یافته شود، ادامه پیدا می‌کند.

۳،٤- ساختار الگوریتم تبرید ژنتیک پیشنهادی

در روش پیشنهادی با بررسی بهترین فرد هر تکرار الگوریتم ژنتیک به‌عنوان جواب اولیه الگوریتم تبرید در نظر گرفته می‌شود. این الگوریتم به‌صورت تک‌مسیری عمل می‌کند و در هر مرحله بهینه‌سازی تابع هزینه را با توجه به دمای فعلی و وضعیت جاری بهبود می‌بخشد. با ترکیب این دو الگوریتم، می‌توانیم یک روش بهینه‌سازی کامل برای مسائل بهینه‌سازی داشته باشیم. در این روش، ابتدا الگوریتم ژنتیک برای تولید راه‌حل‌های اولیه استفاده می‌شود. سپس الگوریتم تبرید به هر یک از راه‌حل‌ها اعمال می‌شود تا بهینه‌ترین جواب ممکن به‌دست آید. در این روش، الگوریتم ژنتیک به‌عنوان یک الگوریتم بهینه‌سازی جهت یافتن یک راه‌حل خوب به کار گرفته می‌شود. سپس، با استفاده از الگوریتم تبرید، این راه‌حل به‌دست آمده بهبود می‌یابد. حلقه داخلی الگوریتم تبرید با ساختن همسایه‌ای برای راه‌حل موجود که به‌صورت تصادفی هست شروع می‌شود. به این صورت که یکی از ۳ طریق، جابه‌جایی تعدادی از وظایف، برعکس کردن صف ورود وظایف و جابه‌جایی یک وظیفه به یک منبع دیگر انجام می‌پذیرد. راه‌حل‌های تولید شده (همسایه‌ها) با جواب اولیه مقایسه می‌شوند و اگر بهتر بودند به‌عنوان بهترین راه‌حل یعنی کمترین زمان تکمیل درخواست‌ها قرار



شکل (۷) : گام‌های روش پیشنهادی.

```
Algorithm 2: Proposed algorithm
BEGIN-PROCEDURE
Input: population_genetic
Output: best_population
1   For (population_genetic)
2       current_solution ⟵ random_solution()
3       current_temperture ⟵ initial_temperature
4       While (current_temperture > final_temperture)
5           new_solution = generate_neighbor(current_solution)
6           delta  ⟵ evaluate_cost(new_solution) – evaluate_cost(new_solution)
7           If (delta > 0)
8               current_solution = new_solution
9           else
10              probability = exp (-delta / current_ temperature)
11          If (random() < probability)
12              current_solution = new_solution()
13              current_temperture = update_temperature (current_temperature)
14      END-While
15  END-FOR
16  END-PROCEDURE
```

شکل (۸) : شبه کد الگوریتم پیشنهادی.

می‌گیرند و اگر بهتر نبودند بر اساس دمای فعلی یک فرصت برای پردازش خواهد داشت. به‌طورکلی، با ترکیب این دو الگوریتم فعالیت‌های مختلفی می‌توان در این روش برای بهبود عملکرد آن انجام داد. به‌عنوان مثال، می‌توان با تغییر میزان احتمال جهش در الگوریتم ژنتیک یا تغییر دمای اولیه در الگوریتم تبرید، عملکرد بهتری از الگوریتم‌ها به دست آورد. همچنین، می‌توان تعداد تکرارهای هر الگوریتم را نیز تنظیم کرد تا عملکرد بهتری از آن‌ها حاصل شود. هنگام اعمال الگوریتم‌های مبتنی بر جمعیت مانند الگوریتم ژنتیک به یک مسئله، دو حالت ممکن است رخ دهد، اول این که الگوریتم به سمت جواب بهینه سراسری همگرا شود، در این حالت کارایی الگوریتم باگذشت زمان بهتر شده و جواب‌های قابل قبولی را تولید خواهد کرد. دوم اینکه الگوریتم به سمت بهینه‌های محلی همگرا شود، در این حالت کارایی الگوریتم به‌شدت تخریب شده و جواب‌های تولیدی، قابل‌اتکا نخواهند بود. رهیافت پیشنهادی الگوریتم را به سمت جستجوی سراسری هدایت می‌کند. گام‌های روش پیشنهادی در شکل (۷) و الگوریتم کلی روش پیشنهادی در شکل (۸) نشان داده شده است.

۵. پیاده سازی و ارزیابی

در این بخش سعی می‌شود با پیاده‌سازی و شبیه‌سازی الگوریتم، عملکرد آن ارزیابی و بررسی شود. با توجه به مشکلات پیاده‌سازی الگوریتم در محیط‌های واقعی رایانش ابری، در این کار پژوهشی از شبیه‌سازی برای ارزیابی روش پیشنهادی استفاده شده است. ابزار مورد استفاده پیاده‌سازی این روش یک موتور شبیه‌سازی با استفاده از زبان متلب[۱۹] است.

۱،۵- محیط آزمایش

طراحی و پیاده‌سازی رهیافت پیشنهادی با استفاده از زبان متلب انجام شده است. محیط انجام آزمایش شامل یک مرکز داده و تعدادی میزبان یا ماشین فیزیکی هست. هر ماشین فیزیکی دارای منابع محاسباتی هست که در اختیار ماشین‌های مجازی برای اجرای درخواست کاربران قرار می‌گیرد. در این مرحله ابتدا داده‌ها به‌صورت تصادفی و با توجه به نیاز مسئله به تعداد مشخص تولید می‌شوند. این داده‌ها مانند تعداد درخواست‌ها، تعداد منابع پردازش، زمان آماده‌سازی هر درخواست بعد از یک درخواست مشخص در یک منبع مشخص و همچنین میزان حداقل و حداکثرهایی برای مقادیر سرعت پردازنده‌ها، حافظه‌ها و سرعت شبکه‌ها هستند.

19- Matlab

۲,۵ – مفروضات ارزیابی الگوریتم پیشنهادی

در این رهیافت، یک مجموعه از وظایف و یک مجموعه از منابع در نظر گرفته می‌شود. هر وظیفه باید بر روی یکی از منابع اجرا شود و هر منبع هم می‌تواند به چندین وظیفه اختصاص داده شود. هر وظیفه دارای زمان شروع و پایانی مشخص است که باید در آن زمان اجرا شود. هدف این است که زمان اجرای کلیه وظایف بهینه و کمینه شود .در ادامه هر منبع دارای پردازنده و مقدار مشخصی حافظه است. همچنین، ورود وظایف بر اساس نرخ تبادل شبکه نیاز به زمان بارگذاری دارد و حافظه هر منبع پردازش با توجه به پردازنده نیاز به زمانی جهت خارج کردن وظیفه قبلی وارد شده دارد.

۵-۲-۱. تولید درخواست و منابع

با اجرای موتور شبیه‌سازی ابتدا نیاز است که درخواست‌ها در اختیار موتور قرار گیرد. بدین منظور اولین بخشی که در موتور فعال می‌شود بخش تولید درخواست است. در مسئله‌های مختلف، بار ورودی و درخواست‌های کاربران می‌تواند از الگوی خاصی پیروی کند و یا این که به‌صورت تصادفی باشد. در ارزیابی انجام شده، بار ورودی توسط کاربران دارای هیچ الگوی مشخصی نیست و به‌صورت تصادفی، درخواست‌های کاربران به واسط ارسال می‌شود. درخواست‌های ارسالی از طرف کاربران دارای ویژگی‌های برای اجرا در محیط رایانش ابری می‌باشند. این ویژگی تا شامل موارد زیر هست.

۱. هر درخواست دارای اندازه مشخصی هست که نشان‌دهنده تعداد دستورات مورد نیاز برای انجام این درخواست است. در ارزیابی انجام شده، طول و اندازه درخواست‌های کاربران به‌صورت تصادفی انتخاب می‌شوند. حجم درخواست بین ٤٠ تا ١٠٠٠ مگابایت است.

۲. هر ماشین فیزیکی تعدادی ماشین مجازی را میزبانی می‌کند. وظیفه اجرای درخواست‌های کاربران بر عهده ماشین‌های مجازی هست. هر ماشین مجازی منابع خود را از ماشین فیزیکی خود تأمین می‌کند و در نتیجه همانند ماشین‌های فیزیکی یک ماشین مجازی دارای ویژگی‌های محاسباتی هست. هر ماشین مجازی دارای ویژگی‌هایی است که در زیر قابل‌مشاهده است:

۱. مقدار حافظه اصلی بین ۱ تا ٤ گیگ.

۲. سرعت پردازنده بین ۲,۳ تا ۵,۸ گیگاهرتز.

۳. زمان بارگذاری هر درخواست بعد از یک وظیفه روی یک منبع پردازشی بین ۱ تا ۱۰ واحد زمانی تولید شده است.

۵-۲-۲. مقادیر پارامترهای الگوریتم ترکیبی

۱. تعداد تکرار حلقه تولید و بررسی برازندگی فرزندان که در آزمایش‌ها برابر ۱۰۰ فرض شده است [٢٤ و ٢٥].

۲. تعداد تکرار حلقه داخلی الگوریتم تبرید که در آزمایش‌ها برابر ۱۰۰ فرض شده است [٢٤ و ٢٥].

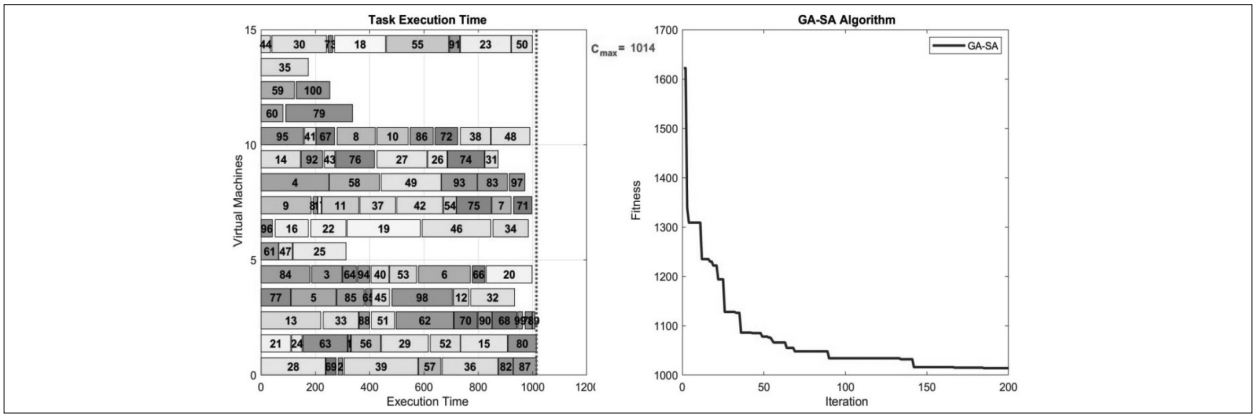
۳. تعداد جمعیت اولیه برای الگوریتم ژنتیک توسعه داده شده برابر ۱۰۰۰ فرض شده است [٢٤ و ٢٥]..

٤. نرخ کاهش دما مورد ارزیابی قرار گرفته و مقدار ۰,٤٨ بهترین نتیجه را در آزمایش‌ها نشان داده است.

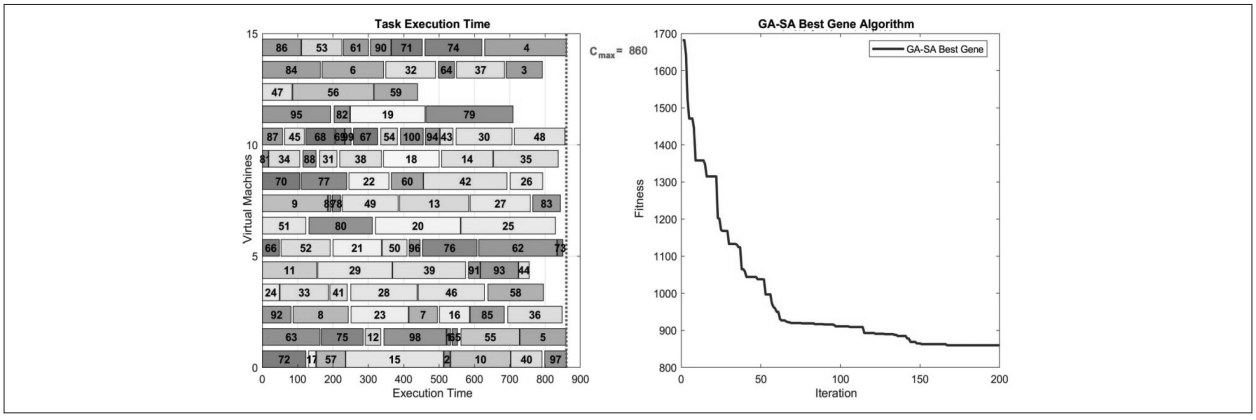
۵. تابع تناسب که مقادیر خروجی آن در آزمایش بخش ۵-۳-۷ مورد ارزیابی قرار گرفته است.

۳,۵ – ارزیابی

در این قسمت به بررسی عملکرد الگوریتم پیشنهادی، با توجه به محیط و شرایط گفته شده، پرداخته می‌شود. با توجه به شرایطی که در قسمت‌های قبل گفته شد، نحوه ارزیابی الگوریتم به این صورت است که در ابتدا محیط رایانش ابری شامل ۱۵ ماشین مجازی است. ماشین‌های مجازی با مقادیر حافظه اصلی و پردازنده بین ۱ تا ٤ گیگ و ۲,۳تا ۵,۸ گیگاهرتز در نظر گرفته شده‌اند. تعداد درخواست‌های ورودی ۱۰۰ بوده و همچنین، حجم درخواست‌های ورودی بین ٤٠ تا ١٠٠٠ مگابایت است و به‌صورت تصادفی انتخاب می‌شوند. زمان بارگذاری هر وظیفه بعد از یک وظیفه روی یک منبع پردازشی بین ۱ تا ۱۰ واحد زمانی تولید شده است. برای ارزیابی روش ارائه شده دو آزمایش طراحی شده است که در این بخش



شکل (۹) : نمودار رهیافت پیشنهادی با بررسی تمامی نمونه‌های منتخب.



شکل (۱۰) : نمودار رهیافت پیشنهادی با بررسی بهترین نمونه منتخب.

شـــرایط آن‌ها و اطلاعات به دســـت آمده از آن‌ها را ارائه می‌کنیم. سپس، اطلاعات به‌دســـت‌آمده از این دو آزمایش را بررســـی و معیارهای ارزیابی را محاســـبه می‌کنیم. در این قسمت از پژوهش، برای یافتن جواب بهینه به دو روش پیاده‌سازی انجام می‌شود. یکی به این صورت که الگوریتم تبرید برای تک‌به‌تک نمونه‌ها جمعیت تولید شـــده توســـط الگوریتم ژنتیک توســـعه داده‌شده، اجرا شود. روش دیگر که نتایج بهتری به همراه به این صورت است که الگوریتم تبرید تنها بر روی بهترین فرد از جمعیت اجرا گردد.

۵-۳-۱. روش اول

طبق روش اول، نمودار شـــکل (۹) پاسخ نهایی رهیافت پیشـــنهادی با بررســـی تمامی نمونه‌های منتخب الگوریتم ژنتیک (ســـمت چپ) و مقدار برازندگـــی بهترین جواب هر تکرار (ســـمت راســـت) را نمایش می‌دهد. تابع تناســـب با بررســـی مقدار زمـــان پایانی تکمیل کل درخواســـت‌ها در تمام منابع پردازشی به‌روزرسانی می‌شود. جواب انتهایی مسئله کمترین مقدار به‌دســـت آمده از تابع تناسب هست. در تکرارهای مختلف بر روی نســـل‌ها میزان تابع تناسب متفـــاوت خواهد بـــود. همان‌طور که در نمودار مشـــخص اســـت Cmax زمان تکمیل اجرای درخواســـت‌ها بر روی منابع اســـت. هر چه Cmax مقدار کمتر داشـــته باشد یعنی زمان تکمیل درخواست‌ها کمینه شود، الگوریتم پیشنهادی عملکرد بهتری در زمان‌بندی منابع دارد.

۵-۳-۲. روش دوم

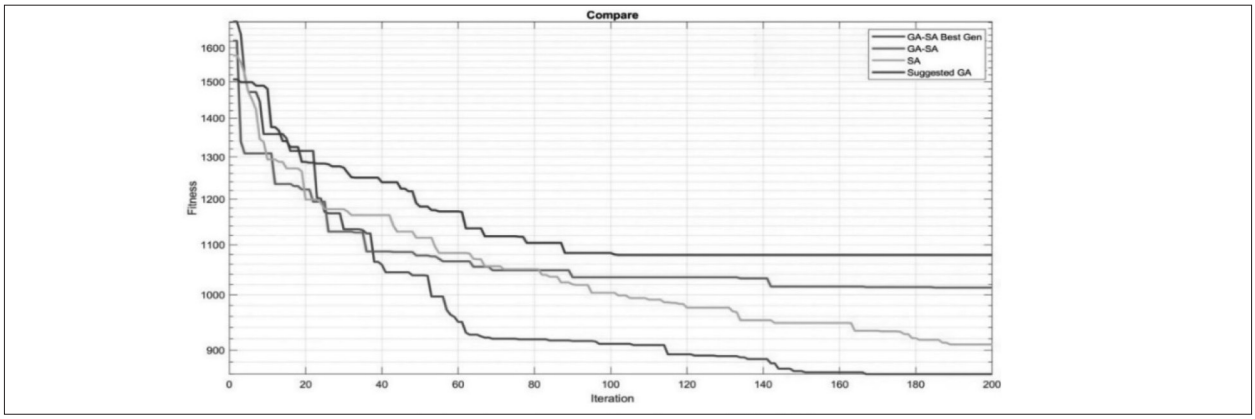
مطابق با روش دوم، نمودار شـــکل (۱۰) پاسخ رهیافت پیشـــنهادی با بررسی بهترین فرد منتخب الگوریتم ژنتیک (سمت چپ)و مقدار برازندگی بهترین جواب هر تکرار (سمت راست) را نمایش می‌دهد. طبق نمودار Cmax مقداری برابر با ۸۶۰ دارد. پس در روش پیشـــنهادی با بررسی بهترین فرد منتخب الگوریتم نتیجه‌ی بهتری حاصل می‌شود.

۵-۳-۳. مقایسه الگوریتم با روش‌های دیگر

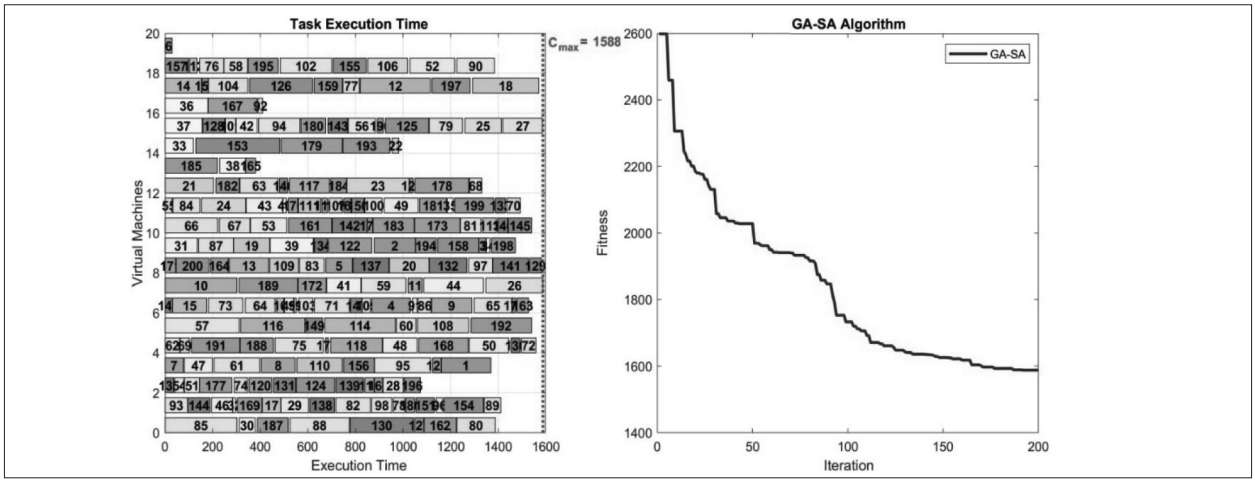
در ایـــن قســـمت تلاش می‌شـــود تا عملکـــرد رهیافت

جدول (۱): مقادیر ثبت شده زمان تکمیل اجرای درخواست‌ها برای۱۰۰درخواست.

ژنتیک توسعه داده شده	تبرید	ژنتیک- تبرید	ژنتیک- تبرید بهترین نمونه	تعداد تکرار	ماشین مجازی	درخواست
۱۰۷۹	۱۰۸۰	۱۰۱۴	۸۶۰	۱۰۰	۱۵	۱۰۰



شکل (۱۱) : نمودار مقایسه برای۱۰۰ درخواست.



شکل (۱۲) : نمودار رهیافت پیشنهادی با بررسی تمام نمونه‌های منتخب.

پیشنهادی، با الگوریتم‌های اکتشـــافی دیگر مقایسه شود. به این منظور، رهیافت پیشـــنهادی هردو روش با الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید مقایسه می‌شود. نتایج به‌دست آمده از اجرای این چهار الگوریتم به‌صورت زیر بوده که نشان‌دهنده‌ی میزان تابع تناسب در ۱۰۰ دور تکرار است.

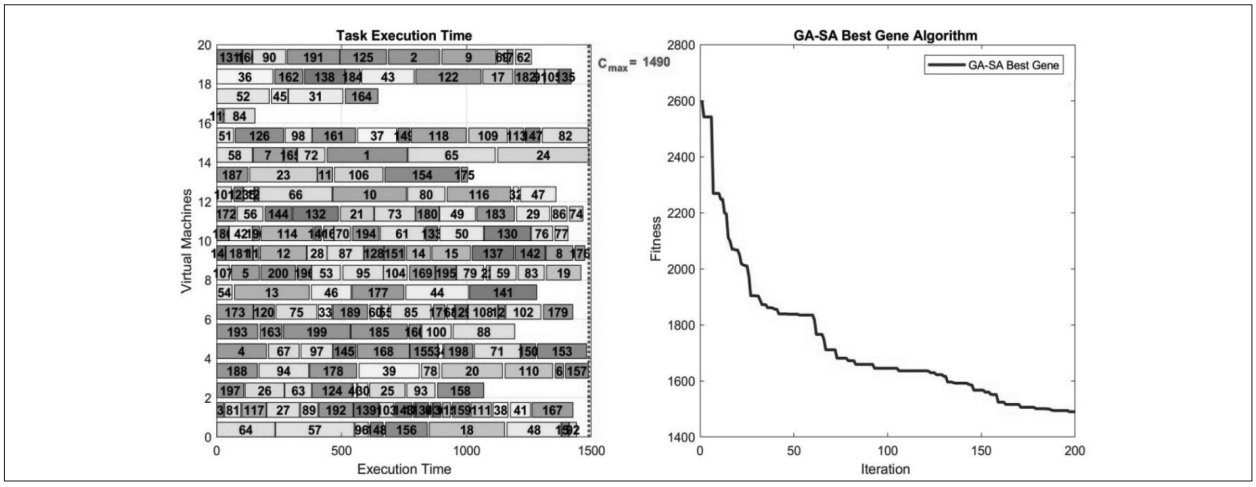
برای الگوریتم پیشنهادی طبق نمودار شکل (۱۱) میزان تابع تناسب نسبت به بقیه الگوریتم‌ها بیشتراست. در جدول (۳) زمان تکمیـــل اجرای تمامی درخواســـت‌ها برای ۱۰۰ درخواســـت و ۱۵ منبع اندازه‌گیری شـــده است و رهیافت پیشنهادی در هر دو روش با الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید مقایسه می‌شود.

۵-۳-۴. انجام آزمایش با نرخ ورودی بیشتر

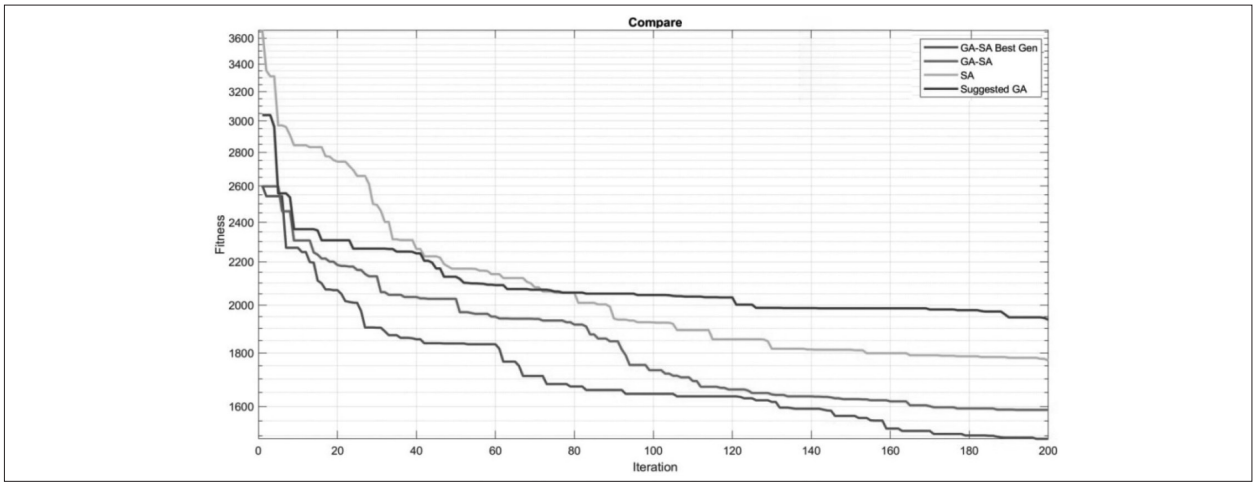
در این آزمایش هم ماننـــد آزمایش اول نحوه ارزیابی الگوریتم به این صورت هست که ۲۰ ماشین مجازی داریم و ماشین‌های مجازی با مقادیر حافظه و پردازنده بین ۱ تا ٤ گیگ و ۲,۳ تا ۵,۸ گیگاهرتز اســـت. تعداد درخواست‌های ورودی ۲۰۰ بوده و همچنین حجم درخواست‌های ورودی بیـــن ٤۰ تا ۱۰۰۰ مگابایت اســـت که به‌صـــورت تصادفی انتخاب می‌شوند.

۵-۳-۵. روش اول

همانند آزمایش اول، نمودار شـــکل (۱۲) پاســـخ نهایی



شکل (۱۳) : نمودار رهیافت پیشنهادی با بررسی بهترین نمونه منتخب.



شکل (۱۴) : نمودار مقایسه برای ۲۰۰ درخواست.

رهیافت پیشنهادی با بررسی تمامی نمونه‌های منتخب الگوریتم ژنتیک و مقدار برازندگی بهترین جواب هر تکرار را نمایش می‌دهد. تعداد حلقه‌های تکرار روی جمعیت اولیه به‌گونه‌ای در نظر گرفتیم که جمعیت به‌درستی همگرا شود. طبق نمودار تعداد حلقه‌های اجرا برابر ۲۰۰ در نظر می‌گیریم. طبق نمودار Cmax مقداری برابر با ۱۵۸۸ دارد.

۵-۳-۶. روش دوم

مطابق با روش دوم، نمودار شکل (۱۳) پاسخ رهیافت پیشنهادی با بررسی بهترین نمونه منتخب الگوریتم ژنتیک (سمت چپ) و مقدار برازندگی بهترین جواب هر تکرار (سمت راست) را نمایش می‌دهد.

۵-۳-۷. مقایسه الگوریتم با روش‌های دیگر

همانند آزمایش اول، رهیافت پیشنهادی هردو روش با الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید مقایسه می‌شود. نتایج به‌دست آمده از اجرای این الگوریتم‌ها به‌صورت زیر هست. نمودار شکل (۱٤) مقایسه ٤ الگوریتم اجرا شده در این تحقیق را نشان می‌دهد.

در جدول (٤) زمان تکمیل اجرای تمامی درخواست‌ها برای ۲۰۰ درخواست و ۲۰ منبع اندازه‌گیری شده است و رهیافت پیشنهادی در هر دو روش با الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید مقایسه می‌شود.

۶- جمع‌بندی و کارهای آینده

در این پژوهش، رایانش بدون سرویس‌دهنده موردبررسی قرار داده شد. در این مدل سرویس که یکی از مدل‌های سرویس در رایانش ابری است، تولیدکنندگان

جدول (۲): مقادیر ثبت شده زمان تکمیل اجرای درخواست‌ها برای ۲۰۰ درخواست.

ژنتیک توسعه داده شده	تبرید	ژنتیک- تبرید	ژنتیک- تبرید بهترین نمونه	تعداد تکرار	ماشین	درخواست
۱۹۳۷	۱۷۷۰	۱۵۸۸	۱۴۹۰	۲۰۰	۲۰	۲۰۰

نیازی به نگرانی در مورد چگونگی توسعه و نگه‌داری زیرساخت‌های ابری و خدمت‌دهندگان را ندارند و فقط روی توسعه کد تمرکز می‌کنند. برخلاف معنای ظاهری، این مدل سرویس به معنای نیاز نداشتن به سرویس‌دهنده نیست، بلکه به معنای نداشتن نگرانی در مورد پیکربندی و مدیریت سرویس‌دهنده توسط تولیدکنندهٔ برنامه کاربردی است. همان‌طور که دیدیم، دو اصطلاح «رایانش بدون سرویس‌دهنده»، «تابع به‌عنوان سرویس» در این پایان‌نامه و به‌طورکلی در ادبیات رایانش ابری به‌جای یکدیگر به‌کار رفته‌اند. در معماری تابع به‌عنوان سرویس، یک برنامه کاربردی به واحدهایی مستقل به نام تابع تجزیه می‌شوند. طبق این مدل، تولیدکننده فقط بر منطق برنامه و نوشتن توابع تمرکز می‌کند و مدیریت کامل پشته نرم‌افزار بر عهده ارائه‌دهنده ابر است. همان‌طور که اشاره شد، یکی از چالش‌ها، زمان‌بندی منابع در این محیط است. بدین منظور تاکنون روش‌ها متعددی برای حل مسئله زمان‌بندی و استفاده بهینه از منابع، ارائه شده‌اند. در روش پیشنهادی با استفاده از ترکیب دو الگوریتم ژنتیک توسعه داده شده و الگوریتم تبرید، از مزایای هر دو الگوریتم استفاده کردیم. این دو الگوریتم مکمل یکدیگر شده و عملکرد بهتری نسبت به روش‌های دیگر دارند. برای روش ارائه شده معیارهای ارزیابی مختلفی نیز ارائه کردیم و رهیافت پیشنهادی را به دو صورت پیاده‌سازی کردیم. یکی به این صورت که الگوریتم تبرید برای تک‌تک نمونه‌های جمعیت تولید شده توسط الگوریتم ژنتیک توسعه داده‌شده، اجرا شود. روش دیگر که نتایج بهتری به همراه دارد و عملکرد بهتری در زمان تکمیل اجرای درخواست‌ها دارد به این صورت است که الگوریتم تبرید تنها بر روی بهترین فرد از جمعیت اجرا شود. در بخش آخر، روش ارائه شده با الگوریتم‌های دیگر مقایسه شد. این مقایسه به کمک یک شبیه‌ساز که با زبان متلب توسعه داده شده است، انجام شده است. در ادامه، پیشنهادهایی برای بهبود عملکرد زمان‌بندی معرفی می‌شوند.

۱. یکی از کارهایی که پیشنهاد می‌شود معرفی پارامترهای جدیدتر به‌صورت چندبعدی با در نظر گرفتن منافع کاربران، سرویس‌دهندگان، کاهش انرژی و کاهش ترافیک شبکه است. همچنین، می‌توان مهلت را به‌عنوان پارامتر اصلی کیفیت خدمات در زمان‌بندی در نظر گرفت. یک درخواست که دارای مهلت زمانی است موجب افزایش هزینه برای کاربر می‌شود.

۲. برای بهبود عملکرد الگوریتم می‌توان مؤلفه‌های بیشتری مانند مقدار حافظه، پهنای باند شبکه و ویژگی زمانی حال حاضر (به‌طور مثال روزهای آخر هفته یا ساعات اوج درخواست کاربران) را برای بررسی شرایط سیستم در نظر گرفت.

۳. برای تولید جمعیت اولیه الگوریتم می‌توانیم به‌جای استفاده از توزیع تصادفی از یک ساختار پویا استفاده کنیم و همچنین می‌توانیم تعداد دورهای تکرار را روی نسل‌های مختلف کروموزوم‌ها بیشتر کنیم تا همگرایی زودتر حاصل شود.

٤. استفاده از یک سازوکار محاسبه ضریب شباهت مبتنی بر مقدار آستانه، دو وضعیت ناهمپوشان را برای جمعیت تعریف می‌کند: جمعیت همگرا و جمعیت غیرهمگرا. به‌عنوان مثال اگر مقدار حد آستانه(β)، ۰/۸ در نظر گرفته شود، در این صورت جمعیت با ضریب شباهت متوسط ۰/۷۹، یک جمعیت غیرهمگرا و جمعیت با ضریب شباهت متوسط ۰/۸۱ یک جمعیت همگرا در نظر گرفته می‌شود، درحالی‌که این دو جمعیت تفاوت چندانی با یکدیگر ندارند. در مطالعات آینده جهت حل این مشکل، می‌توان از روش‌های غیر آستانه‌ای مانند روش‌های فازی، استفاده نمود.

۵. تحمل خطاپذیـــری را درروش پیشـــنهادی در نظر بگیریـــم به‌گونـــه‌ای که بـــا افزایش خطا و گیـــر افتادن در بهینه‌های محلی زمان اجرای الگوریتم طولانی نشـــود. این عامل تأثیر زیادی بر عملکرد روش پیشنهادی دارد.

٦. یکی از کارهایی که پیشـــنهاد می‌شود افزایش تعداد تکرارها روی نســـل‌های به وجود آمده از الگوریتم ژنتیک است تا همگرایی به بهترین صورت ممکن انجام گیرد.

مراجع

[1] O. Alqaryoutia, N. Siyamb, "Serverless Computing and Scheduling Tasks on Cloud: A," American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), vol. 40, pp. 235-247, 2018.

[2] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, NJ. Yadwadkar, RA. Popa, JE. Gonzalez, I. Stoica, DA. Patterson, "What serverless computing is and should become: The next phase of cloud computing," Communications of the ACM, vol. 64, pp. 76-84, 2021.

[3] M. Masdari, S. ValiKardan, Z. Shahi, SI. Azar, "Towards workflow scheduling in cloud computing: a comprehensive analysis," Journal of Network and Computer Applications, vol. 66 pp. 64-82, 2016.

[4] T. Dillon, C. Wu, E. Chang, "Cloud computing: issues and challenges," in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, April-2010, Perth, WA, Australia, pp. 27-33.

[5] H. Akbar, M. Zubair, MS. Malik, " The Security Issues and challenges in Cloud Computing," International Journal for Electronic Crime Investigation, vol. 7, pp. 13-32, 2023.

[6] Q. Zhang, L. Cheng, R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, pp. 7-18, 2010.

[7] "cloud computing", Faceprep, URL:http://www.faceprep.in/skillboard/cloud-computing/, Access Date:10 October 2022.

[8] V. Saratchandran, "Cloud service models saas, iaas, paas-choose the right one for your business," Fingent, Jan 28, 2018.

[9] A. Huth, J. Cebula, "The basics of cloud computing," United States Computer, pp. 1-4, 2011.

[10] M. Masdari, SS .Nabavi, V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," Journal of Network and Computer Applications, vol. 66, pp. 106-127, 2016.

[11] "The state of virtual machines in 2020", Gitconnected, URL: http:// levelup.gitconnected.com/the-state-of-virtual-machines-in-2020-22f5d6c8a40d/, Access Date: 11 October 2022.

[12] A. Suresh, A. Gandhi, "Fnsched: An efficient scheduler for serverless functions," in Proceedings of the 5th International Workshop on Serverless Computing, December-2019, New York, United States, pp .19-24.

[13] H. Shafiei, A. Khonsari, P. Mousavi, "Serverless computing: a survey of opportunities, challenges, and applications," ACM Computing Surveys, vol. 54, pp. 1-32, 2022.

[14] L. Mao, Y. Li, G. Peng, X. Xu, W. Lin, "A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds," Sustainable Computing: Informatics and Systems, vol. 19, pp. 233-241, 2018.

[15] J. Chen, P. Han, Y. Liu, X. Du, "Scheduling independent tasks in cloud environment based on modified differential evolution," Concurrency and Computation: Practice and Experience, pp. e6256, 2021.

[16] BT. Rao, LS. Reddy, "Survey on improved scheduling in Hadoop MapReduce in cloud environments," arXiv preprint arXiv:1207.0780. Jul 3, 2012.

[17] OM. Elzeki, MZ. Reshad, MA. Elsoud, "Improved max-min algorithm in cloud computing," International Journal of Computer Applications, vol. 50, no. 12, 2012.

[18] P. Slavík, "Improved performance of the greedy algorithm for partial cover," Information Processing Letters, vol. 64 , no. 5, pp. 251-254, 2010.

[19] A. Omidi, AM. Rahmani, "Multiprocessor independent tasks scheduling using a novel heuristic PSO algorithm," in Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, August-2009, Beijing, China, pp. 369-373.

[20] JP. Mapetu, Z. Chen, L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," Applied Intelligence, vol. 49, pp. 3308-3330, 2019.

[21] A. Choudhary, I. Gupta, V. Singh, PK. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," Future Generation Computer Systems, vol. 83, pp. 14-26, 2018.

[22] K. Hentsch, P. Köchel, "Job scheduling with forbidden setups and two objectives using genetic algorithms and penalties: A case study at a continuous casting plant," Central European Journal of Operations Research, vol. 19, pp. 285-298, 2011.

[23] N. Raman, AB. Wahab, S. Chandrasekaran, "Computation of workflow scheduling using backpropagation neural network in cloud computing: A virtual machine placement approach," The Journal of Supercomputing, vol. 77, pp. 9454-9473, 2021.

[24] M. Valter Rogério, et al. "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure," Neural Computing and Applications, vol. 27, pp. 2383-2406, 2016.

[25] A. Nikravesh, Y. Samuel A. Ajila, and C. Lung. "Using genetic algorithms to find optimal solution in a search space for a cloud predictive cost-driven decision maker," Journal of Cloud Computing, vol. 7 pp. 1-21, 2018.