دو فصلنامه علمی
فناوری اطلاعات و ارتباطات ایران

سال پانزدهم، شماره ۵۷ و۵۸ ، پاییز و زمستان ۱۴۰۲
صفحات:۱۹۱ الی ۲۱۰

Improving the load balancing in Cloud computing using a rapid SFL algorithm (R-SFLA)

Kiomars Salimi[*], Mahdi Mollamotalebi[**]

[*]Department of Computer Engineering, Islamic Azad University, Buin Zahra Branch, Buin Zahra, Iran
[**]Department of Computer and Information Technology Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran

Abstract

Nowadays, Cloud computing has many applications due to various services. On the other hand, due to rapid growth, resource constraints and final costs, Cloud computing faces with several challenges such as load balancing. The purpose of load balancing is management of the load distribution among the processing nodes in order to have the best usage of resources while having minimum response time for the users' requests. Several methods for load balancing in Cloud computing have been proposed in the literature. The shuffled frog leaping algorithm for load balancing is a dynamic, evolutionary, and inspired by nature. This paper proposed a modified rapid shuffled frog leaping algorithm (R-SFLA) that converge the defective evolution of frogs rapidly. In order to evaluate the performance of R-SFLA, it is compared to Shuffled Frog Leaping Algorithm (SFLA) and Augmented Shuffled Frog Leaping Algorithm (ASFLA) by the overall execution cost, Makespan, response time, and degree of imbalance. The simulation is performed in CloudSim, and the results obtained from the experiments indicated that the proposed algorithm acts more efficient compared to other methods based on the above mentioned factors.

Keywords: Cloud computing, Load balancing, Rapid shuffled frog leaping, Resource scheduling, Response time

کیومرث سلیمی، مهدی ملامطلبی، دوفصلنامه فناوری اطلاعات و ارتیاطات ایران، سال پانزدهم، شماه های ۵۷ و۵۸، پاییز و زمستان ۱۴۰۲، صفحه ۱۹۱ الی ۲۱۰

بهبود توازن بار در رایانش ابری با استفاده از الگوریتم جهش قورباغه سریع (R-SFLA)

کیومرث سلیمی*، مهدی ملامطلبی**×

*گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد بوئین زهرا، بوئین زهرا، ایران

*گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی، واحد قزوین، قزوین، ایران

تاریخ دریافت: ۱۴۰۱/۰۵/۰۷ تاریخ پذیرش: ۱۴۰۱/۱۰/۲۱

نوع مقاله: پژوهشی

چکیده

امروزه رایانش ابری به علت ارائه خدمات متنوع، کاربردهای زیادی دارد. از سوی دیگر، به علت رشد سریع، محدودیت منابع و هزینه نهایی، چالش‌های متعددی در رایانش ابری به وجود آمده است که یکی از این چالش‌ها، توازن بار است. منظور از توازن بار، چگونگی مدیریت توزیع بار در بین گره‌های پردازشی، به‌منظور استفاده بهینه از منابع و صرف کمترین زمان جهت پاسخ به درخواست کاربر است. روش‌های متعددی در خصوص برقراری توازن بار پیشنهاد شده‌اند که یکی از آن‌ها، الگوریتم جهش قورباغه است که پویا، تکاملی و الهام گرفته از طبیعت می‌باشد. در این مقاله، بهبودی بر الگوریتم جهش قورباغه پیشنهاد شده است که باعث همگرایی سریع و بستن راه حلقه تکرار تکامل معیوب قورباغه‌ها، می‌گردد. جهت ارزیابی، الگوریتم جهش قورباغه بهبود یافته پیشنهادی $\mathcal{R}$-$\mathcal{SFLA}$ و الگوریتم $\mathcal{SFLA}$ و الگوریتم $\mathcal{ASFLA}$ در شبیه‌ساز کلودسیم تحت شرایط یکسان، مورد آزمایش قرار گرفتند. نتایج به‌دست‌آمده از آزمایشات، بیانگر آن است که روش پیشنهادی نسبت به روشهای دیگر، از نظر هزینه کلی اجرا، زمان پاسخ و درجه توازن بار، کاراتر عمل نموده است.

واژگان کلیدی: رایانش ابری، توازن بار، الگوریتم جهش قورباغه، زمانبندی منابع، زمان پاسخ

×نویسنده مسئول: مهدی ملامطلبی motalebi@gmail.com

۱. مقدمه

رایانش ابری یک از رویکردهای جدید رایانشی است که در چند سال اخیر موردتوجه قرارگرفته و به‌طور فزاینده در حال گسترش است. موسسه ملی فناوری و استاندارد (NIST) رایانش ابری را این‌گونه تعریف می‌کند: رایانش ابری مدلی برای فراهم کردن دسترسی آسان بر اساس تقاضای کاربر از طریق شبکه، به مجموعه‌ای از منابع رایانشی قابل‌تغییر و پیکربندی (مانند شبکه‌ها، سرورها، فضای ذخیره‌سازی، برنامه‌های کاربردی و سرویس‌ها) است. در این رویکرد هدف این است که دسترسی با کمترین نیاز به مدیریت منابع و نیاز به دخالت مستقیم فراهم‌کننده سرویس، به‌سرعت فراهم‌شده یا آزاد گردد.

با توجه به ساختار معماری، رایانش ابری از مجازی‌سازی و قابلیت استفاده چندگانه نیز پشتیبانی می‌کند [۱]. این سیستم رایانشی در کنار مزایایش دارای معایب و چالش‌هایی نیز می‌باشد که یکی از این چالش‌ها، توازن باراست که در این تحقیق به آن پرداخته شده است.

در محیط‌های ابری، مراکز داده سرویس‌هایی را جهت استفاده کاربران ارائه می‌دهند. درصورتی‌که چندین مرکز داده وجود داشته باشد، الگوریتم‌های متفاوتی جهت انتخاب این مراکز داده و ارسال درخواست‌های کاربران به‌سوی آن‌ها وجود دارد. مسیریابی ترافیک بین پایگاه‌های کاربر و مراکز داده، توسط کارگزار سرویس کنترل می‌شود و تصمیم‌گیری این موضوع که سرویس‌دهی به درخواست کاربر به‌عهده کدام مرکز سرویس‌دهی است توسط مجموعه کارگزار سرویس انجام می‌شود.

یکی از الگوریتم‌هایی که می‌توان از آن جهت توازن بار استفاده کرد الگوریتم توازن بار جهش قورباغه (SFL) می‌باشد. الگوریتم جهش قورباغه‌ها است که از تکامل طبیعی گروهی از قورباغه ها زمانی که به دنبال محل با بیشترین ذخیره غذایی در دسترس می‌گردند، الهام گرفته است. در الگوریتم جهش قورباغه، هر قورباغه نمایانگر یک‌راه حل قابل‌قبول در مسئله بهینه‌سازی است که دارای یک مقدار شایستگی است. قورباغه‌ها بر اساس شایستگی‌شان به‌صورت نزولی مرتب می‌شوند و بر اساس روندی خاص، به زیرمجموعه‌های مختلف تقسیم می‌شوند.

این مقاله با بررسی الگوریتم مذکور، پیشنهادهایی جهت بهبود در نواحی که وجود ضعف، محتمل است ارائه داده است. این نواحی شامل مرحله جهش محلی، جهش سراسری و تغییر در بخش سانسور قورباغه تکامل نیافته است. اهداف این تحقیق، تقسیم بهتر منابع محاسباتی جهت افزایش کارایی در سرورهای مجازی رایانش ابری، و کاهش زمان پاسخگویی به درخواست کاربران است.

در ادامه این مقاله، روشهای تامین توازن بار در رایانش ابری با ذکر توانایی‌ها و کاستی‌های آنها در بخش دوم مرور می‌شوند. در بخش سوم، الگوریتم جهش قورباغه بهبود یافته با جزئیات معرفی شده و در بخش چهارم، نتایج پیاده‌سازی و ارزیابی الگوریتم پیشنهادی ارائه می‌شود. جهت ارزیابی روش پیشنهادی، نتایج حاصل از آن با مقاله [۲] و [۳] مقایسه شده است. در پایان در بخش پنجم، تحقیق حاضر جمع‌بندی شده و پیشنهادهای تحقیقی آتی ارائه می‌گردند.

۲. کارهای مرتبط

با توجه به انواع درخواست‌هایی که به رایانش ابری می‌رسد، در حالت‌های مختلف، یک یا چند ابر وجود دارند که در آن‌ها منابعی جهت ارائه انواع سرویس به‌صورت واقعی و مجازی وجود دارد که جهت رسیدگی به انواع تقاضا از آن‌ها استفاده می‌شود. درواقع، توازن بار یک فرایند جابه‌جایی و بارگذاری فعالیت در رایانش ابری است تا گره‌ها در یک سیستم اشتراکی، استفاده بهینه از منابع داشته باشند و زمان پاسخ به درخواست بهبود یابد. نحوه تقدم و تأخیر رسیدگی و همچنین انتخاب ماشین مجازی سرویس‌دهنده به درخواست، به عهده متوازن کننده بار است.

مهمترین اهداف توازن بار در محیط رایانش ابری عبارتند از [۴]: الف) افزایش دسترس‌پذیری : سازوکارهای بهبود دسترس‌پذیری شامل سه مرحله اصلی تحمل خرابی، افزونگی و حفاظت در برابر سربار است ب) بهبود کارایی: با توجه به این که توازن صحیح بار باعث رسیدگی سریعتر به درخواستها و کاهش تأخیر می‌گردد، بنابراین بهبود قابل‌توجهی در کارایی رایانش ابری حاصل می‌شود ج) حفظ ثبات سیستم: با توجه به کاهش زمان کارکرد و فشار کاری سیستم، امکان ازکارافتادن کاهش یافته و درنتیجه، ثبات سیستم حفظ می‌گردد د) افزایش انعطاف‌پذیری سیستم: توازن بار با ارائه الگوریتم‌های زمان‌بندی خاص، باعث می‌شود گره‌ها خود را با هر تغییری سریعا انطباق دهند ه) افزایش تحمل خرابی: در صورت بروز خطا/خرابی، با توجه به وجود گره‌های رایانشی متعدد، الگوریتم‌های توازن بار می‌توانند کار را به گره‌های دیگر بسپارند و) افزایش رضایت کاربران: در صورت برقراری صحیح توازن بار، با توجه به روال پرداخت به میزان استفاده، زمان پاسخ‌گویی به

کاربـــر و به تبع آن، هزینه‌ها کـــاهش می‌یابد ز) کاهش مصرف انرژی: با توجه به سیاست استفاده حداکثری از گره‌های موجود در توازن بار، گره‌هایی که نیاز به استفاده از آنها نیست می‌توانند خاموش شوند.

در ادامه، برخی از چالش‌های اصلی که بر نحوه عملکرد الگوریتمهای توازن بار در رایانش ابری تأثیر دارند معرفی می‌شوند [۵]:

الف) توزیع فاصله گره‌های ابر: برخی از الگوریتم‌ها فقط برای شبکه اینترانت یا گره‌های نزدیک که در آنها، زمان تأخیر ارتباطی ناچیز است، طراحی‌شده‌اند. بااین‌حال، طراحی یک الگوریتم متوازن کننده بار که بتواند با هر نوع توزیع فاصله گره‌های ابر کار کند، یک چالش محسوب می‌شود. دلیل این موضوع آن است که باید عوامل دیگری مانند سرعت ارتباط شبکه بین گره‌ها، فاصله بین مشتری و گره‌های پردازش کار، و فاصله بین گره‌های درگیر ارائه خدمات، موردتوجه قرار گیرد. همچنین ایجاد یک سازوکار کنترلی برای متوازن کننده بار جهت توزیع تمام گره‌هایی که قادر به تحمل موقت تأخیر بالا باشند، ضرورت دارد.

ب) ذخیره‌سازی و تکثیر□□□□: یک الگوریتم تکثیر کامل، بهره‌وری رادر ذخیره‌سازی در نظر نمی‌گیرد چراکه داده‌های مشابه در تمام گره‌ها، به‌صورت تکثیرشده ذخیره می‌شوند. بنابراین، به دلیل آنکه ذخیره‌سازی بیشتری موردنیاز است، الگوریتم‌های تکثیر کامل باعث افزایش هزینه‌ها می‌شوند. در مقابل، الگوریتم‌های تکثیر جزئی، می‌توانند قطعات مجموعه داده‌ها را در هر گره (با سطح معینی از همپوشانی) بر اساس قابلیت‌های هر گره مانند قدرت پردازش و ظرفیت، ذخیره کنند و بنابراین، می‌توانند به استفاده بهتر منجر شوند. از سوی دیگر، این الگوریتم‌ها تلاش می‌کنند در سراسر گره‌های ابر، بخش‌های مختلف مجموعه داده در دسترس حساب کاربری باشند ولی این موضوع پیچیدگی الگوریتم‌های متوازن کننده بار را افزایش می‌دهد.

ج) پیچیدگی الگوریتم: ترجیح داده می‌شود که الگوریتم‌های متوازن کننده بار ازنظر پیاده‌سازی و عملیات، پیچیدگی کمتری داشته باشند. علاوه بر این، زمانی که الگوریتم‌ها نیاز به اطلاعات بیشتر و ارتباطات وسیعتر برای نظارت و کنترل دارند، تأخیر افزایش و بهره‌وری کاهش می‌یابد. درنتیجه، الگوریتم‌های متوازن کننده بار باید در ساده‌ترین شکل ممکن طراحی شوند.

د) نقطه شکست□□□: کنترل توازن بار و جمع‌آوری داده‌ها در مورد گره‌های مختلف باید به‌گونه‌ای طراحی شود که از داشتن یک نقطه منفرد شکست در الگوریتم جلوگیری کند. برخی از الگوریتم‌ها (متمرکز) می‌توانند یک‌راهکار کارآمد و مؤثر برای حل توازن بار در یک الگوی خاص ارائه دهند زیرا این الگوریتم‌ها دارای یک کنترل‌کننده برای کل سیستم هستند. در چنین مواردی، اگر کنترل‌کننده کارش را خوب انجام ندهد، کل سیستم شکست خواهد خورد. پس در طراحی هر الگوریتم متوازن کننده بار، باید این چالش را مد نظر قرار داد.

الگوریتمهای توازن بار ارائه شده در پیشینه تحقیق را می‌توان از جنبه‌های مختلفی دسته‌بندی نمود. یکی از تقسیم‌بندی‌ها بر اساس متمرکز□□ و غیرمتمرکز□□□ بودن است. الگوریتم‌های متمرکز به‌صورت یکپارچه توسط یک گره یا سرور کنترل می‌شوند و در صورت خرابی نقطه کنترل کننده مرکزی، کل سیستم از کار می‌افتد. مزیت این نوع الگوریتمها، سادگی و هزینه پیاده‌سازی پایین است. از سوی دیگر، نقاط ضعف این نوع الگوریتمها، ارتباطات پرحجم در نقاط کنترل مرکزی، نقطه منفرد خرابی□□□□، و مقیاس‌پذیری پایین است.

در الگوریتم‌های غیرمتمرکز یا توزیع‌شده□□□□□، مدیریت در بین گره‌ها پخش می‌شود. جهت پیاده‌سازی این نوع الگوریتم، در زمان طراحی باید حداکثر ظرفیت و مقیاس‌پذیری مشخص گردد. نقاط ضعف این نوع الگوریتمها نسبت به الگوریتم‌های متمرکز، پیچیدگی بالا و هزینه پیاده‌سازی بیشتر است. همچنین مزیت این نوع الگوریتمها، مقیاس‌پذیری بالاتر و تحمل بالای شکست به علت عدم اتکا یا کم بودن اتکا به نقطه‌ای خاص است.

تقسیم‌بندی دیگر بر اساس همگن□□□□ و ناهمگن□□□ بودن محیط پیاده‌سازی است. کلیه عناصر (شبکه، سخت‌افزار، نرم‌افزار) موجود در محیط‌های همگن، یکسان یا دارای تفاوت‌های ناچیزی باهم هستند. با توجه به محدودیت‌های حافظه و/یا نیازهای محاسباتی، فرایند توزیع مجدد شبکه باید تا جایی که ممکن است، محل داده را حفظ کند. ازآنجاکه عموماً ارتباط بین پردازنده‌ها هزینه زیادی در بر دارد، توزیع مجدد باید با حداقل مهاجرت داده‌ها انجام شود. در صورت همگن بودن محیط، بروزرسانی وضعیت شبکه در هر مرحله جهت تقسیم و توازن بار، ساده‌تر انجام می‌گیرد. ایستگاه‌های کاری در محیط ناهمگن، ممکن است ازلحاظ معماری، سیستم‌عامل، سرعت پردازنده، اندازه حافظه و فضای دیسک در دسترس، متفاوت باشند. اگرچه چنین محیط‌هایی مزایای متعددی را ارائه می‌دهند، اما حداکثر بهره‌برداری از قدرت پردازشی آن‌ها بسیار دشوار است چراکه هماهنگی بین عناصر، زمانبر است [۶].

تقسیم‌بندی دیگر بر اساس شروع‌کننده توازن بار است بطوریکه اگر توازن بار توسط فرستنده شروع شود، اصطلاحاً فرستنده آغازگر[۴] نامیده می‌شود و اگر عملیات توازن بار توسط گیرنده شروع شود، اصطلاحاً گیرنده آغازگر[۵] نامیده می‌شود و اگر ترکیبی باشد، یعنی کل عملیات توسط فرستنده و گیرنده به صورت مشترک انجام شود، اصطلاحاً متقارن[۶] نامیده می‌شود [۷].

الگوریتمهای توازن بار سرور بر اساس سخت‌افزاری بودن و نرم‌افزاری بودن نیز تقسیم‌بندی می‌شوند. در روش سخت‌افزاری، افزونگی و متوازن‌سازی بار توسط سوئیچ‌ها در لایه چهارم شبکه فراهم می‌شود و در روش نرم‌افزاری، از ابزارهای نرم‌افزاری برای محاسبه میزان استفاده از سرور و تخصیص منابع استفاده می‌گردد. صرف‌نظر از اینکه آیا در متوازن‌سازی بار از نرم‌افزار یا سخت‌افزار استفاده شده باشد، روش متوازن‌سازی بار به دو نوع پویا و ایستا نیز تقسیم‌بندی می‌شود. متوازن‌سازی بار ایستا برای حالتی که پردازش بار قابل پیش‌بینی باشد، استفاده می‌شود و متوازن‌سازی بار پویا، برای حالتی که پردازش بار غیرقابل‌پیش‌بینی باشد، مناسب است [۸]. اگر تنوع در گره‌ها کم باشد، روش ایستا قابل‌اجرا است. در محیط‌های ناهمگن، الگوریتم‌های پویا ترجیح داده می‌شوند [۹]. در ادامه، الگوریتم‌های توازن بار پویا و ایستا با جزئیات بیشتری مرور می‌شوند.

در توازن بار ایستا، تغییرات آنی سیستم در نظر گرفته نمی‌شود یعنی در صورت پایین آمدن سطح عملکرد یا ازکارافتادن سرور میزبان یا تغییر در حجم و تعداد درخواست‌ها، طراحی الگوریتم به شکلی است که استراتژی راه‌حل تغییرات پویا در آن پیش‌بینی‌نشده است. درنتیجه تصمیم جهت انتقال بار، به وضعیت حال حاضر ماشین مجازی وابسته نیست. توازن بار ایستا اغلب جهت شرایطی مناسب است که بارکاری مربوط به یک مجموعه، با تغییرات کم یا بدون تغییرات باشد. هدف الگوریتم متوازن‌سازی ایستا آن است که هزینه اجرایی را با توجه به مهلت اجرای برنامه و دسترسی به منابع، به حداقل برساند. بااین‌حال، الگوریتم‌های متوازن‌سازی ایستا قادر به یافتن راه‌حل بهینه نبوده [۳] و انعطاف‌پذیرنیستند و به دانش قبلی در مورد سیستم نیاز دارند [۱۰].

در توازن بار پویا، تغییرات آنی سیستم در نظر گرفته می‌شود؛ یعنی سیستم به وضعیت فعلی متکی است و در صورت پایین آمدن سطح عملکرد یا ازکارافتادن ماشین میزبان یا تغییر در حجم و تعداد درخواست‌ها، استراتژی راه‌حل تغییرات پویا در آن پیش‌بینی‌شده است و بار به سیستم‌هایی که، کم‌بار یا بدون بار هستند انتقال می‌یابد. در این نوع الگوریتم، پیچیدگی‌های موجود در هر گره از قبیل ویژگی‌های رایانشی مطرح است و بار به‌صورت پویا از یک گره که دارای بارکاری بالایی می‌باشد به سمت گره‌هایی حرکت می‌کند که بارکاری کمتری بر روی آنها وجود دارد؛ ازاین‌رو پیاده‌سازی به‌مراتب پیچیده‌تری دارند. این الگوریتمها به بررسی دائمی گره‌ها نیاز دارند و معمولاً پیاده‌سازی آنها دشوار است. آنها برای محیط رایانش ابری مناسب هستند زیرا در زمان اجرا، بار را توزیع می‌کنند و وزن‌های مناسب را به سرور اختصاص می‌دهند؛ بنابراین، بعد از جستجو، مناسب‌ترین سرور را در شبکه جهت تخصیص وظایف ترجیح می‌دهند [۱۱]. آنها همچنین در مقایسه با الگوریتم‌های ایستا، دارای قابلیت رقابتی بالاتری هستند [۱۲].

در ادامه، روشهای ارائه شده جهت توازن بار در محیط رایانش ابری، مرور و نقد و بررسی می‌شوند.

در الگوریتم توازن بار نوبت چرخشی (RR)[۱] [۱۱] که به صورت ایستا و متمرکز عمل می‌نماید، یک زمان ثابت به کلیه کارها اختصاص داده می‌شود. تأکید این الگوریتم بر عدالت و محدودیت زمانی است. در این الگوریتم، یک صف حلقوی جهت اختصاص منبع وجود دارد. این الگوریتم برای انجام هر یک از کارها، مدت‌زمانی برابر اختصاص می‌دهد. در الگوریتم نوبت چرخشی، بارها به‌صورت مساوی بین ماشین‌های مجازی توزیع می‌شوند.

ازجمله محدودیت‌های این الگوریتم آن است که جهت دستیابی به کارایی بالا، نباید بیش از یک اتصال مشتری در یک‌زمان شروع شود. مزایای این الگوریتم، سادگی و تقسیم مساوی بارکاری بین گره‌ها است و از معایبش، می‌توان به بار اضافی روی زمان‌بند به دلیل تصمیم‌گیری برای اندازه‌گیری تخصیص زمان اشاره نمود که در صورت یکسان نبودن قدرت پردازش ماشین مجازی، ممکن است برخی از گره‌ها پربار و برخی دیگر بیکار یا کم‌بار باشند. همچنین بارکنونی ماشین‌های مجازی را در نظر نمی‌گیرد؛ یعنی درحالی‌که ممکن است یک ماشین مجازی کاملاً آزاد باشد، درخواست را به ماشینی با تعداد درخواست‌های زیاد تخصیص می‌دهد.

الگوریتم توازن بار حداقل- حداقل[۲] [۱۳]، با مجموعه‌ای از کارها که قبلاً به هیچ‌یک از گره‌ها اختصاص نیافته است، شروع می‌شود. ابتدا حداقل زمان اجرا برای تمام کارهای موجود محاسبه می‌شود. هنگامی‌که این محاسبه تکمیل شد، کاری که دارای کمترین زمان اتمام است انتخاب‌شده و به گره مربوطه اختصاص داده می‌شود. زمان اجرای کارهای دیگر که در حال حاضر موجود

است، به‌روز می‌شود و کار اجرا شده از مجموعه کارهای موجود حذف می‌شود. این روال تا هنگامی‌که تمامی کارها به ماشین‌های معادل اختصاص داده شوند انجام می‌شود.

این الگوریتم، زمانی که کارهای کوچک بیشتر از کارهای بزرگ هستند، بهتر عمل می‌کند. یکی از معایب این الگوریتم آن است که، ایده‌آل‌ترین نتیجه برای کارهایی است که در اولین مقایسه، کمترین زمان اجرا را به دست می‌آورند. همچنین منابع به کوچک‌ترین کار تخصیص می‌یابند بطوریکه کارهای کوچک‌تر زودتر اجرا می‌شوند و این در حالی است که کارهای بزرگ‌تر در صف انتظار مانده، و درنهایت مجبور به استفاده از ماشین ضعیف می‌شوند. همچنین مزیت این الگوریتم ساده و سریع بودن آن بوده و توانایی فراهم‌آوری عملکرد بهبودیافته را دارد. در این الگوریتم، معیار بهره‌برداری از منابع، سربار، توان عملیاتی، زمان پاسخ و کارایی در نظر گرفته‌شده است و از تخصیص‌های تکراری غیرضروری جلوگیری می‌کند.

در الگوریتم توازن بار حداکثر – حداقل [۱۴]، برای اولین بار، تمام کارهای موجود به سیستم ارسال می‌شوند. پس از محاسبه حداقل زمان اتمام برای همه کارها، کاری که زمان تکمیل حداکثر را دارد انتخاب‌شده و به گره مربوطه اختصاص داده می‌شود. این الگوریتم، زمانی که تعداد کارهای بزرگ بیشتر از تعداد کارهای کوچک است بهتر از الگوریتم حداقل- حداقل عمل می‌کند. اگر در یک صف کاری، تنها یک کار بزرگ ارائه شده باشد، الگوریتم کار کوچک را همزمان با کار بزرگ اجرا می‌کند. در این الگوریتم، میزان زمان اختصاص منابع کاربر در کارهای کوتاه مدت، نسبت به کارهای بلند مدت، تقریباً شبیه الگوریتم حداقل- حداقل است به‌جز شرایطی که تعداد کارهایی که در زمان بلندمدت تکمیل می‌شود زیاد باشد.

در این الگوریتم، وظایفی که دارای زمان حداکثر تکمیل هستند، ابتدا اجرا می‌شوند و در انتها کارهایی که دارای حداقل زمان اتمام هستند، انجام می‌شوند. ایده‌آل‌ترین نتیجه برای کارهایی است که در اولین مقایسه، بیشترین زمان اجرا را به دست می‌آورند. در این الگوریتم، معیار بهره‌برداری از منابع، سربار، توان عملیاتی، زمان پاسخ و کارایی در نظر گرفته‌شده است.

الگوریتم توازن بار متوقف‌شده (TA) [۱۵]، مبتنی بر ماشین مجازی است. برای هر مشتری، ابتدا درخواست به متوازن کننده بار ارسال می‌شود و متوازن کننده بار شروع به پویش می‌کند. بعد از پویش، لیستی از ماشین‌های مجازی و وضعیت (مشغول و یا خالی بودن) آن‌ها تهیه می‌کند، تا بتواند سرویس لازم را به درخواست‌ها بدهد. متوازن کننده بار توسط این لیست برای یافتن ماشین مجازی مناسب که در دسترس باشد اقدام می‌کند، و در صورت یافت شدن، کار به ماشین مجازی اختصاص داده می‌شود. در غیر این صورت، کار در صف باقی می‌ماند و ماشین مجازی، در ادامه روند، در دسترس درخواست‌های جدید قرار می‌گیرد، تا کارهای واردشده جدید را انجام دهد. مزیت این روش، زمان پاسخ پایین برای درخواست‌های کاربران و مشکل آن، در انتظار ماندن درخواست‌ها در صف مرکز داده است. همچنین این الگوریتم در زمان تخصیص یک درخواست، بار فعلی ماشین مجازی را در نظر نمی‌گیرد که این می‌تواند باعث افزایش زمان پاسخ‌دهی شود.

الگوریتم توازن بار نظارت فعال (AMLB) [۱۶]، اطلاعاتی در مورد هر ماشین مجازی و تعداد درخواست‌های فعلی که به هر ماشین مجازی تخصیص داده‌شده را نگهداری می‌کند. زمانی که یک درخواست جدید وارد می‌شود، ماشین مجازی‌ای که کمترین بار را دارد شناسایی‌شده و درخواست ورودی را به آن ماشین مجازی اختصاص می‌دهد. به‌این‌ترتیب، کار بیشتری به ماشین‌های مجازی با توانایی پردازش زیاد، نسبت به ماشین‌های مجازی ضعیف تخصیص می‌یابد. با این کار، موقعیت‌های بحرانی را کاهش می‌دهد و زمان پاسخگویی را به شکل قابل‌توجهی کاهش می‌دهد.

ازآنجایی‌که ممکن است قوی‌ترین ماشین مجازی باقدرت پردازش بالا در اوایل به یک کار کوچک تخصیص داده‌شده باشد، پس قوی‌ترین ماشین مجازی حال حاضر می‌تواند قوی‌ترین ماشین مجازی مجموعه نباشد. توان پردازش لحظه‌ای ماشین مجازی به سیاست‌های برنامه‌ریزی در دو سطح مرتبط است. علاوه بر این، یک کار بزرگ، توان پردازشی بیشتری نسبت به یک کار کوچک نیاز دارد. بنابراین، تعداد کارهایی که در حال حاضر به ماشین مجازی اختصاص داده‌شده، قدرت واقعی لحظه‌ای ماشین مجازی را نشان نمی‌دهد و این مساله، ضعف این روش محسوب می‌شود. از مزایای این الگوریتم می‌توان به در نظر گرفتن همزمان هر دو شاخص بار و قدرت پردازشی ماشین‌های در دسترس اشاره کرد.

در الگوریتم توازن بار توزیع یکسان اجرای کنونی (ESCE) [۱۷]، متوازن کننده بار به‌طور مداوم صف کار و لیست ماشین‌های مجازی را پویش می‌کند. بعد از پویش، اگر بیش از یک ماشین مجازی در دسترس که بتواند درخواست را اداره کند یافت شود، اولین ماشین مجازی را انتخاب و کار را به آن اختصاص می‌دهد و سپس شناسه آن را به کنترل کننده مرکز داده ارسال می‌کند. اگر

تعداد زیادی ماشین مجازی وجود داشته باشد که زیر بار هستند و باید از بار آزاد شوند، متوازن کننده بار کارها را به شکلی توزیع می‌کند که کار جدید به ماشین مجازی کم‌بار اختصاص یابد تا بار به شکل تقریباً مساوی بین ماشین‌های مجازی تقسیم شود.

مزیت این الگوریتم آن است که بار کنونی هر ماشین مجازی را در نظر می‌گیرد و با توجه به آن، درخواست‌ها را تخصیص می‌دهد. بنابراین امکان آنکه یک ماشین مجازی با بار زیاد و یک ماشین با بار کم در مرکز داده به وجود آید از بین می‌رود. مشکل این الگوریتم نقطه‌ی شکست و سربار زمانی زیادی می‌باشد که باید جهت تعویض درخواست‌ها و تخصیص منابع به آن‌ها صرف شود.

الگوریتم توازن بار زنبورعسل (BA)□□□□□ [۱۸]، ازجمله الگـوریتم‌های غیرمتمرکز و فرا ابتـکاری است. در این الگوریتم سرورها نقش زنبورهای کارگر را دارند که تحت نظر سرور مجازی گروه‌بندی می‌شوند و هر سرور یکی از درخواست‌های صف خود را پردازش می‌کند و خروجی خود را محاسبه می‌کند که این عمل نظیر نشان دادن کیفیت غذا توسط زنبور است. سرورها پس از پردازش کار، می‌توانند بازده خروجی خود را با بازده خروجی کل سرورها مقایسه کنند. اگر بازده خروجی نزدیک به بازده خروجی کل سرورها باشد، سرور در مجموعه سرورهای مجازی فعلی می‌ماند، اما اگر بازده خروجی کم باشد، متوازن کننده بار، سرور موردنظر را بیکار یا کم‌کار در نظر می‌گیرد.

این الگوریتم به‌خوبی با انواع مختلف منابع کار می‌کند، اما در مقایسه با افزایش تعداد منابع، بازده مناسبی ندارد. از مزایای این الگوریتم، بهبود کیفیت سرویس‌های ارائه‌شده به مشتریان، و به حداقل رساندن زمان تکمیل اجرا است. از سوی دیگر، مقیاس‌پذیری پایین از معایب این الگوریتم است [۱۹].

الگوریتم توازن بار کلونی مورچه (ACO)□□□□□□ [۲۰]، غیرمتمرکز و فرا ابتکاری است که راه‌حل‌ها را به‌صورت متوالی ایجاد می‌کند. محدودیت اصلی این الگوریتم، انتخاب ترکیبی از ویژگی‌هایی است که در اکثر موارد به یک راه‌حل مطلوب منجر نمی‌شوند. بنابراین، این الگوریتم بر اساس تصادف و احتمال است. از مزایای این الگوریتم می‌توان به بهبود کارایی، تحمل شکست بالا و به‌کارگیری زیاد منابع اشاره کرد.

در الگوریتم توازن بار بهینه‌سازی ازدحام ذرات (PSO)□□□□□□□ [۲۱]، ذرات در فضای جستجو، حرکت می‌کنند و تغییر در موقعیت ذرات در فضای جستجو با توجه به تجربه خود و همسایگانش توسط تابعی از الگوریتم اصلی تعیین می‌شود. مناسب‌ترین ماشین‌های مجازی که کارها در آنها بارگذاری شده و یافتن بهترین طرح برنامه‌ریزی کار، با استفاده از الگوریتم ازدحام ذرات مشخص می‌گردد. بهینه‌سازی ازدحام ذرات معمولاً زمانی پایان می‌یابد که هیچ پیشرفت قابل‌توجهی واقع نشود. از مزایای این الگوریتم می‌توان به حداقل رساندن مهاجرت، سرعت همگرایی بالا و داشتن حافظه اشاره کرد. از سوی دیگر، گرفتار شدن در بهینه محلی و همگرایی زودرس و کاهش تنوع در جمعیت، از جمله کاستی‌های آن است.

الگوریتم توازن بار ژنتیک (GA)□□□□□ [۲۲]، از روش جستجو بر اساس تکامل طبیعی و ژنتیک استفاده می‌کند. اثربخشی این روش بستگی به بهره‌وری کاوش دارد. سه عملگر برای دستیابی به اکتشاف و بهره‌برداری شامل انتخاب، تقاطع و جهش است [۲۳]. از این الگوریتم دارای قابلیت جستجوی موازی هدف بوده و از مزایای الگوریتم‌های تکاملی بهره می‌برد. از معایب آن می‌توان به بالا بودن هزینه اجرایی (نگهداری تعداد زیادی از کروموزوم‌ها در چند نسل جهت یافتن بهینه‌ترین جواب) اشاره نمود.

الگوریتم توازن بار جستجوی ارگانیک سمبوتیک (SOS)□□ [۲۴]، از راهکار همیاری، هم‌سفرگی و انگلی برای به‌روزرسانی موقعیت بردار راه‌حل، در فضای جستجو استفاده می‌کند و یک فرایند تکراری برای بهینه‌سازی راه‌حل توازن بار است. این الگوریتم، جمعیتی از مجموعه اعضاء را نشان می‌دهد که یکی از راه‌حل‌های نامزد، مسئله بهینه سازی است. از اطلاعات مربوط به متغیرهای تصمیم‌گیری و ارزیابی راه‌حل به‌عنوان شاخصی از عملکرد مجموعه اعضاء، محافظت به عمل می‌آید. مجموعه مسیرهای جواب با استفاده از مراحل ارتباط همزیستی، اصلاح می‌شوند.

از مزایای این الگوریتم آن است که پارامترهای خاص را به‌صورت یکپارچه در یک گروه از راه‌حل‌های انتخابی برای رسیدن به یک راه‌حل سراسری قرار می‌دهد و برخلاف بسیاری دیگر از الگوریتم‌های تکاملی، این الگوریتم، فرزندان (راه‌حل جدید) را چندین بار تولید نمی‌کند و سازگاری آن از طریق تعاملات فردی است. در انتهای هر مرحله، از انتخاب حریصانه استفاده می‌کند تا راه‌حل قدیمی یا اصلاح‌شده را در اکوسیستم حفظ کند. این الگوریتم تنها از دو پارامتر حداکثر تعداد ارزیابی و اندازه جمعیت استفاده می‌کند در صورتیکه الگوریتم‌های دیگر تکاملی نیاز به تنظیم حداقل یک پارامتر علاوه بر این دو پارامتر دارند. همچنین این الگوریتم ریسک ناسازگاری عملکرد را به دلیل تنظیم پارامتر نامناسب از بین می‌برد و این موضوع ثبات عملکرد را افزایش می‌دهد. از سوی دیگر، عیب این الگوریتم، افزایش مضاعف هزینه با بزرگ شدن مسائل است.

در [۲۹]، شکل پیشرفته‌ای از الگوریتم بهینه‌سازی قورباغه ادغام شده که بر روی ماهیت و رفتار قورباغه‌ها در جستجوی غذا کار می‌کند. با استفاده از این تکنیک بهینه سازی، تابع هزینه واحد پردازش مرکزی و تابع تناسب محاسبه شده است که مجموع تابع هزینه بودجه و زمان شکل‌گیری است. این روش به کاهش زمان شکل‌گیری و همچنین هزینه متوسط با زمان‌بندی کارها برای ماشین‌های مجازی کمک نموده است و نتایجش حاکی از آن بود که می‌تواند وظایف را برای ماشین‌های مجازی به طور موثرتری در مقایسه با سایر روش‌ها، زمان‌بندی کند.

در [۳۰] یک متعادل کننده بار خودکار برای اطمینان از کشش سیستم ابری و متعادل کردن بار کاری کاربر در بین محفظه‌های موجود در یک محیط چند ابری پیشنهاد شده است. مفهوم چند-حلقه‌ای در این رویکرد برای فعال کردن خودمدیریتی کارآمد قبل از متعادل کردن بار استفاده شده است. وظایف با استفاده از یک الگوریتم زمان‌بندی به نام حداقل‌سازی بازه زمانی محدود مهلت برای زمان‌بندی چند کاره (DCMM-MTS□□□) در محفظه‌ها برنامه‌ریزی می‌شوند. بر اساس زمان‌بندی کار، بار در هر ظرف محاسبه شده و سپس، متعادل می‌شود. نتایج بیانگر آن است که این رویکرد از نظر قابلیت اطمینان، استفاده از CPU، زمان ساخت، استفاده از انرژی، زمان پاسخ‌دهی، هزینه اجرا، زمان بیکاری و مهاجرت وظیفه بهتر از سایر روش‌ها عمل نموده است.

در [۳۱] جهت برنامه‌ریزی وظایف مبتنی بر DWRR□□□□، یک سیستم منطق فازی طراحی شده است که زمان‌بندی وظایف را در محیط ابری بهبود می‌بخشد. یک روش فازی برای به‌روزرسانی وزن اینرسی PSO و به‌روزرسانی مسیرهای فرومونی PACO پیشنهاد نموده است. بدین ترتیب، با بهینه‌سازی کلونی مورچه‌های موازی ازدحام ذرات ترکیبی فازی در رایانش ابری، توانسته است زمان اجرا و انتظار، و توان عملیاتی سیستم را به حداقل برساند و در نتیجه، استفاده از منابع به حداکثر رسیده که زمان‌بندی کارها را بهبود داده است.

در روش [۳۲]، یک پارتیشن گره عمودی پیشنهاد شده است که از روش اکتشافی الگوریتم جهش قورباغه‌ای مختلط (SFLA) جهت خوشه‌بندی و زمان‌بندی بهینه گردش کار استفاده می‌کند. نتایج بیانگر آن است که این تکنیک همراه با روش خوشه‌بندی در مقایسه با روشهای بدون خوشه‌بندی، عملکرد بهتری (از نظر ایجاد و استفاده از منابع) دارد.

همچنین در [۳۳]، یک مدل مدیریت منابع محاسباتی ابری مبتنی بر مه برای خانه‌های هوشمند با الگوریتم بهبود یافته جهش قورباغه مختلط و الگوریتم جستجوی فاخته ارائه شده است. علاوه بر این، از روش شبکه عصبی الاستیک تطبیقی (AENN) برای مدل پیش‌بینی بار جهت بهینه‌سازی زمان پاسخ سرویس تخصیص داده شده، استفاده نموده است. نتایج به کارگیری این روش بیانگر آن است که از نظر پهنای باند شبکه، مصرف انرژی، تأخیر و زمان پاسخ، نسبت به روشهای مشابه، بهبود یافته است.

الگوریتم توازن بار جهش قورباغه (SFL) [۲]، بر پایه حرکت قورباغه‌ها به سمت غذا است. روش کار این الگوریتم به این شکل است که بعد از مشخص شدن تعداد و گروهای جمعیت اصلی (مقداردهی اولیه)، جمعیت اولیه (راه‌حل) به‌صورت تصادفی تولید می‌شود. شایستگی هر عضو جمعیت (قورباغه) سنجیده می‌شود و جمعیت یک گروه بر مبنای شایستگی به‌صورت نزولی مرتب می‌شوند. ممپلکس‌ها (گروه‌های کوچک جمعیتی) تشکیل می‌شوند و بعد، در مرحله جستجو با توجه به مشخص شدن مکان هر عضو مجموعه، جهش‌های اعضای هر ممپلکس جهت رسیدن به بهینه‌های محلی و بهینه سراسری انجام می‌گیرد. بدترین عضوهای هر ممپلکس ازلحاظ جایگاهی، جهش‌ها را انجام می‌دهند.

بهینه محلی انتزاعی از مناسب‌ترین راه‌حلهای محلی ابر جهت یافتن جواب بهینه است و بهینه سراسری انتزاعی از مناسب‌ترین راه‌حلهای سراسری ابر جهت یافتن جواب بهینه است. با توجه به رسیدن برخی از قورباغه‌ها به بهینه‌های محلی و یا سراسری (و خارج شدن از چرخه)، در هر مرحله، قورباغه جدید به‌صورت تصادفی تولید و به چرخه اضافه می‌گردد. این الگوریتم در هر مرحله فعالیت فوق را برای یک ممپلکس انجام می‌دهد. فعالیت فوق تا رسیدن اعضا به همگرایی (بهترین راه‌حل) ادامه می‌یابد.

این الگوریتم در یافتن راه‌حل، مطمئن و سریع است اما به‌شدت بر پارامترهای مدل مناسب (بهترین قورباغه) متکی است و شیوه شفافی برای انتخاب این پارامترها، وجود ندارد. تجزیه وتحلیل عملکرد این الگوریتم نشان می‌دهد که مشکلات مختلف، دارای مجموعه‌های متفاوتی از پارامترهای بهینه است. با توجه به یک مشکل خاص، یک مجموعه مشخص از پـــارامترها مناسب‌تر است. این الـــگوریتم سریع بوده، قابلیت بالایی برای جستجوی سراسری داشته، و پیاده‌سازی آن آسان است.

با توجه به مرور الگوریتم‌های توازن بار، می‌توان نتیجه گرفت که فاصله بین سرورها، سرعت شبکه ارتباطی، قدرت سخت‌افزاری سرورها، هزینه‌های ثانویه ارائه‌دهندگان سرویس، و هزینه

تمام‌شده‌ی سرویس رایانش ابری در توازن بار مؤثرند. در جدول شماره ۱، الگوریتم‌های معرفی شده با هم مقایسه شده‌اند.

با توجه به این که معمولا ناهمگن بودن محیط محتمل است و الگوریتم‌های پویا دارای زمان پاسخ بهتری در محیط ناهمگن هستند، بنابراین الگوریتم‌های پویا برای محیط ناهمگن مناسب‌ترند. از سوی دیگر، در محیط‌های کوچک با توجه به سربار کم، الگوریتم‌های ایستا مناسب‌ترند. الگوریتم‌های پویا، به علت محاسبات پیچیده، سربار اضافی دارند هرچند که در محیط‌های بزرگ، با توجه به زمان پاسخ مناسب، نسبت به الگوریتم های ایستا، این موضوع قابل‌چشم‌پوشی است. درنهایت باتوجه به این موضوع که الگوریتم جهش قورباغه توازن مناسبی میان ارتقا وجستجو دارد در بخش بعدی، روش پیشنهادی این تحقیق بر این مبنا برای توازن بار در محیط رایانش ابری با ذکر جزئیات، ارائه می‌گردد.

جدول ۱. الگوریتم‌های بررسی‌شده و مقایسه با توجه به مزایا و معایب

جدول مقایسه الگوریتم‌های بررسی شده						
مرجع	نام الگوریتم	شناسه	نوع	پیاده سازی	مزایا	معایب
[۴۰]	نوبت چرخشی	RR	ایستا	متمرکز	توزیع بار کاری برابر	زمان پردازش کار در نظر گرفته نشده است.
[۱۷]	حداقل- حداقل	Min-Min	ایستا	متمرکز	کاهش Makespan	بار منبع در نظر گرفته نمی شود، پس منجر به گرسنگی می شود.
[۲۳]	حداکثر - حداقل	Max-Min	ایستا	متمرکز	کاهش Makespan	مشاغل کوچکتر ممکن است برای مدت طولانی صبر کنند
[۴۷]	متوقف‌شده	Throttled	ایستا	متمرکز	زمان پاسخ پائین؛ در نظر گرفتن وضعیت ماشین	عدم در نظر گرفتن زمان پردازش برای هر درخواست
[۳۱]	نظارت فعال	AMLB	ایستا	متمرکز	هر دو بار و در دسترس بودن ماشین مجازی در نظر می‌گیرد	قدرت پردازش واقعی ماشین مجازی را در نظر نمی‌گیرد
[۴۳]	توزیع یکسان اجرای کنونی	ESCE	ایستا	متمرکز	زمان پاسخ و زمان پردازش کار را بهبود یافته است.	نقطه‌ی شکست و سربار زمانی زیاد
[۳۴]	الگوریتم توازن بار زنبورعسل	BA	پویا	غیر متمرکز	به خوبی تحت ناهمگونی منابع کار می کند	افزایش منابع به طور مساوی باعث افزایش کارایی نمی شود
[۴۸]	کلونی مورچه	ACO	پویا	غیر متمرکز	بروزرسانی تدریجی جدول سرور توسط اعضا مکانیزم تشخیص خوب گره های بارگذاری شده	هزینه بالا محاسباتی، پهنای باند اشغال شده به دلیل حضور مورچه ها و نقطه شکست
[۱۹]	بهینه‌سازی ازدحام ذرات	PSO	پویا	غیر متمرکز	زمان پاسخ ؛ در محیط ناهمگن مفید است	پیچیدگی بالاالگوریتم و سربار ارتباطی.
[۲۴]	ژنتیک	GA	پویا	متمرکز	زمان پاسخ ؛ استفاده در محیط ناهمگن	پیچیدگی بالا وسربار زمان محاسباتی.
[۲۸]	جستجوی ارگانیک سمبوتیک	SOS	پویا	غیر متمرکز	بهبود عملکرد با کاهش Makespan، زمان پاسخ و درجه عدم تعادل	افزایش مضاعف هزینه با بزرگ شدن مسائل
[۲۹]	جهش قورباغه	SFL	پویا	غیر متمرکز	قابلیت بالایی برای جستجوی سراسری و پیاده سازی اسان	**به‌شدت بر پارامترهای مدل متکی است**

۳. الگوریتم جهش قورباغه بهبود یافته (R-SFLA) جهت توازن بار در رایانش ابری

یکی از اهداف اصلی الگوریتم‌های توازن بار در محیط رایانش ابری، کاهش زمان پردازش است و الگوریتم‌های تکاملی و فرا ابتکاری در این زمینه، کارا عمل نموده‌اند. در این مقاله، باهدف کاهش زمان پردازش، بهبودهایی بر روی الگوریتم توازن بار جهش قورباغه انجام‌شده است. الگوریتم توازن بار جهش قورباغه یکی از الگوریتم‌های پویا، فرا ابتکاری و بر پایه جمعیت است. در ادامه، برخی مفاهیم مورداستفاده در این الگوریتم که در توصیف روش پیشنهادی از آنها استفاده می‌شود، به‌اختصار شرح داده می‌شوند. در ادامه، ابتدا الگوریتم جهش قورباغه[۲]، به اختصار معرفی شده و سپس بهبودهای مورد نظر این تحقیق با جزئیات ارائه می‌شود. الگوریتم جهش قورباغه شامل دو بخش اصلی

۱- جستجوی سراسری

۲- جستجوی محلی

می‌باشد و هر بخش نیز شامل چند مرحله است. با توجه توضیحات فوق قبل از بخش های اصلی الگوریتم لازم است با بعضی از مفاهیم به صورت اختصار توضیح داده شود.

منظور از ممپلکس، گروه‌های جمعیتی هستند که در الگوریتم جهش قورباغه به‌صورت موازی تشکیل‌شده‌اند و هر گروه، مجاز به بررسی فضا جهت تکامل است. یک دسته از الگوریتم‌های فرا ابتکاری، الگوریتم‌های ممتیکی هستند. کار این دسته از الگوریتم‌ها، بر اساس الگوی رفتاری می‌باشد. همانند جایگاه ژن‌ها در الگوریتم ژنتیک، کوچک‌ترین واحد سازنده این کلاس از الگوریتم‌ها، مم□□□□ نام دارد و یک جریان مسری اطلاعات رفتاری می‌باشد که با تأثیر بر رفتار سایرین (موجودات)، باعث تغییر رفتار آن‌ها می‌شود [۲].

همچنین همانند کروموزوم‌ها که در الگوریتم ژنتیک حافظ و ناقل اطلاعات ژنتیکی هستند، این وظیفه در این کلاس، بر عهده موتیپ□□□□ است که توانایی حفظ و انتشار اطلاعات الگوهای رفتاری را در مم به عهده دارد. در الگوریتم ممتیک و ژنتیک با توجه به معیارهای مختلف، اعضا طبقه‌بندی می‌شوند. به‌عنوان‌مثال، هر قورباغه در الگـــوریتم جهش قورباغه، نمایانگـر یک راه‌حل قابل‌قبول در مسئله بهینه‌سازی است، که دارای یـک مقـدار شایستگی□□□ جهت رسیدن به راه‌حل قابل‌قبول در مسئله بهینه‌سازی می‌باشد. حالتی که در آن، همه قورباغه‌ها به بهترین

قورباغه سراسری (بهترین راه‌حل) هدایت شوند و بهترین قورباغه سراسری نیز بدون تغییر بماند، همگرایی نامیده می‌شود.

در این تحقیق، کلیه آزمایش‌ها با ابزار شبیه سازی کلودسیم انجام شده‌اند که دارای یک کتابخانه معتبر جهت شبیه‌سازی محیط رایانش ابری است. جهت شبیه سازی روش پیشنهادی و مقایسه با روشهای مشابه، در کلود سیم، پیش‌نیازهایی لازم است که مهمترین موارد عبارتند از: ۱) تعداد شش ماشین مجازی با توانهای پردازشی متفاوت به هر یک از الگوریتمهای توازن بار اختصاص داده شده است ۲) جهت صف درخواست‌ها از مجموعه داده استاندارد LCG-2005-1 [۲۵] استفاده شده است ۳) تعداد قورباغه‌ها (راه‌حل‌ها) چهل عدد می‌باشد که به چهار ممپلکس ده عددی تقسیم می‌شوند ۴) هر قورباغه حامل شش ممتیپ می‌باشد که ارزش عددی هر ممتیب می‌تواند عددی از یک تا شش باشد.

الف) بخش جستجوی سراسری الگوریتم جهش قورباغه

این بخش شامل شش مرحله است که در زیر شرح داده شده‌اند.
مرحله ۰- انتخاب m و n به‌طوری‌که m تعداد ممپلکس‌ها و n تعداد قورباغه‌ها در هر ممپلکس است. مجموع قورباغه‌ها که در برکه زندگی می‌کنند با F نمایش داده شده و توسط رابطه (۱) نشان داده می‌شود.

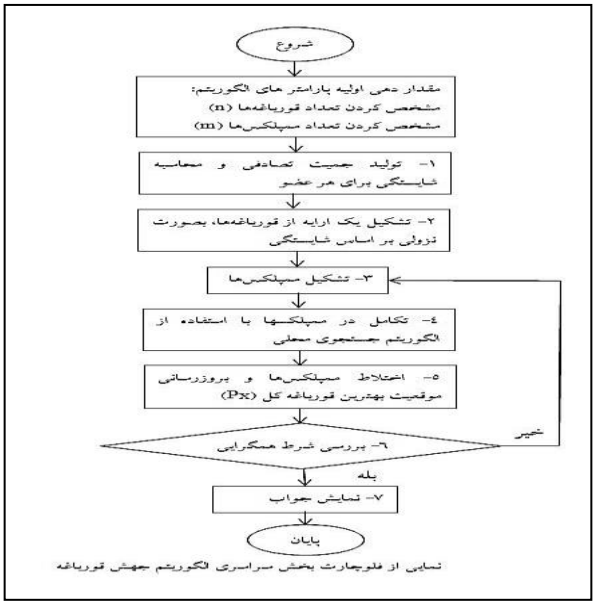
$$F = m*n \qquad (۱)$$

مرحله ۱- ایجاد یک جمعیت مجازی: قورباغه‌های مجازی $F=\{U(1),U(2),\ldots,U(F)\}$ در فضای قابل اجرای $\Omega \subset k^d$ هستند به‌طوری‌که d تعداد متغیرهای تصمیم‌گیری (مثلاً تعداد حالت‌های رفتاری (s) در یک مم که توسط یک قورباغه جابجا می‌شود) است. برای مثال، i امین قورباغه نشان‌دهنده یک بردار از مقدار متغیرهای تصمیم‌گیری $U(i) = (ui1,ui2,\ldots,uid)$می‌باشد و در مراحل بعد، مقدار شایستگی $f(i)$ برای هر قورباغه $u(i)$ محاسبه می‌شود.

مرحله ۲- رتبه‌بندی قورباغه‌ها: کل قورباغه‌ها (F) ازلحاظ مقدار شایستگی به‌صورت نزولی مرتب می‌شوند و در یک آرایه به‌صورت $X = \{U(i), f(i), i = 1, \ldots, F\}$ ذخیره می‌شوند به‌طوری‌که $i = 1$ نشان‌دهنده قورباغه با بهترین شایستگی بوده و بهترین قورباغه جمعیت، با P_X نشان داده می‌شود؛ درنتیجه $P_X = u(1)$ است.

مرحله ۳- دسته‌بندی قورباغه‌ها: در این مرحله، آرایه ذخیره‌شده X در m ممپلکس $(Y_1,Y_2,\ldots Y_m)$ دسته‌بندی می‌شود. هر ممپلکس شامل n قورباغه است. $Yk = [U(j)k, f(j)k|U(j)k = U(k + m(j - 1)), f(j)k = f(k + m(j - 1)), j = 1, \ldots, n] \quad k = 1, \ldots, m;$

برای $m = 3$ رتبه ۱ به ممپلکس ۱ می‌رود، رتبه ۲ به ممپلکس ۲ می‌رود، رتبه ۳ به ممپلکس۳ می‌رود، و رتبه ۴ مجدداً به ممپلکس ۱ می‌رود. این روند به همین شکل تا تقسیم کل قورباغه‌ها به داخل ممپلکس‌ها ادامه می‌یابد.



نمایی از فلوچارت بخش سراسری الگوریتم جهش قورباغه

فلوچارت ۱. بخش سراسری الگوریتم جهش قورباغه

مرحله ۴- تکامل در هر ممپلکس: تکامل $Yk, k = 1,\ldots, m$ بر اساس جستجوی محلی الگوریتم جهش قورباغه ارائه شده در بند بعدی (بند ب)، انجام می‌شود.

مرحله ۵- اختلاط ممپلکس‌ها: پس از تعداد مشخصی از مراحل تکامل ممتیکی در هر ممپلکس، $(Y_1,\ldots,y_m)$ داخل X جایگزین می‌شود به طوری که $X = \{Yk, k = 1, \ldots, m\}$ باشد. X به ترتیب کاهش شایستگی، رتبه‌بندی و موقعیت بهترین قورباغه برکه P_X، به‌روزرسانی می‌شود.

مرحله ۶- بررسی همگرایی: در این مرحله، همگرایی بررسی می‌شود. اگر وضعیت همگرایی مطلوب باشد، الگوریتم متوقف می‌شود و در غیر این صورت، به مرحله ۳ بازگشت می‌شود. به‌طورمعمول، تصمیم‌گیری در ارتباط با توقف، با یک شماره از پیش تعیین‌شده انجام می‌شود که شامل حلقه‌های زمانی متوالی است و این حلقه تا زمانی که حداقل یک قورباغه دارای بهترین الگوی رفتاری ممتاز بدون تغییر شود، ادامه می‌یابد. با توجه به

توضیحات ارائه شده فوق فلوچارت بخش جستجوی سراسری به شکل فوق می‌باشد.

ب) بخش جستجوی محلی الگوریتم جهش قورباغه

جزئیات جستجوی محلی برای هر ممپلکس شامل نه مرحله به شرح زیر است.

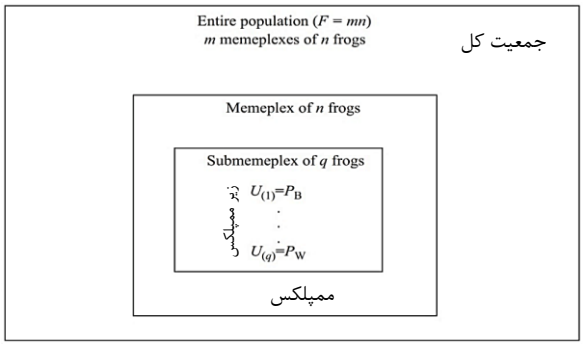
مرحله ۰- در این مرحله، im و iN برابر صفر قرار داده می‌شوند. منظور از im متغیر شمارنده ممپلکس‌ها است و درنهایت، با تعداد کل m ممپلکس برابر می‌باشد. iN تعداد مراحل تکامل را می‌شمارد و درنهایت، با N مرحله تکامل ممپلکس برابر خواهد شد.

مرحله ۱- به im یک واحد اضافه شود (im = im + 1).

مرحله ۲- به iN یک واحد اضافه شود (iN = iN + 1).

مرحله ۳- ساخت زیرممپلکس: هدف قورباغه‌ها حرکت به سمت ایده‌های مطلوب با هدف بهبود الگوهای رفتاری مم خود است. قورباغه‌ها می‌توانند از بهترین قورباغه در میان ممپلکس محلی (Y_{im}) و یا از بهترین ممپلکس سراسری، کمک بگیرند. با توجه به انتخاب بهترین قورباغه سراسری، همیشه استفاده از بهترین قورباغه سراسری مطلوب نیست؛ زیرا قورباغه‌ها می‌توانند در اطراف این قورباغه خاص تجمع کنند درحالیکه ممکن است یک بهینه محلی باشد. به همین دلیل، زیرمجموعه ممپلکس که زیرممپلکس نامیده می‌شود، در نظر گرفته می‌شود. راهبرد انتخاب زیرممپلکس به شکلی است که به قورباغه‌هایی که دارای مقادیر عملکرد بالاتر (شایستگی) هستند، وزن بالاتر و برای قورباغه‌هایی که دارای مقادیر عملکرد پایین‌تر هستند، وزن کمتر می‌دهد. وزن‌ها با توزیع احتمالی مثلثی تعیین می‌شوند.

در این مرحله، از کل قورباغه‌ها در هر ممپلکس، قورباغه‌هایی برای ایجاد آرایه زیرممپلکس Z انتخاب می‌شوند. زیرممپلکس‌ها به شکل نزولی (ازلحاظ شایستگی قورباغه‌ها) مرتب می‌شوند (iq = 1, . . . , q). بهترین موقعیت قورباغه (iq = 1) در زیرممپلکس با P_B، و بدترین موقعیت قورباغه (iq = q) در زیرممپلکس با P_W ثبت می‌شود. مفهوم یک زیرممپلکس در شکل (۱) نشان داده‌شده است.



شکل ۱. تصویری مفهومی از وضعیت کل جمعیت، ممپلکس، زیرممپلکس، بهترین (PB) و بدترین (PW) قورباغه زیرممپلکس [۲]

مرحله ۴- بهبود دادن موقعیت بدترین قورباغه: موقعیت جدید برای قورباغه با بدترین شایستگی در زیرممپلکس توسط رابطه (۲) محاسبه می‌شود:

$$\text{Step size } S = \begin{cases} \min\{\text{int}[\text{rand}(P_B - P_W)], Smax\} & \text{برای گام مثبت} \\ \max\{\text{int}[\text{rand}(P_B - P_W)], -Smax\} & \text{برای گام منفی} \end{cases} \quad (۲)$$

که در این رابطه، rand یک عدد تصادفی در محدوده [۰ ، ۱] و Smax حداکثر اندازه مجاز گام می‌باشد که یک قورباغه پس از تأثیر پذیرفتن از بهترین قورباغه محلی، می‌تواند طی کند. همچنین اندازه و ابعاد گام برابر با تعداد متغیرهای تصمیم‌گیری (مموتیپ) است. موقعیت جدید بدترین قورباغه، توسط رابطه (۳) محاسبه می‌شود.

$$U(q) = P_W + S \quad (۳)$$

اگر U(q) یا موقعیت جدید قورباغه در فضای Ω قابل‌اجرا باشد، مقدار جدید شایستگی قورباغه f(q) محاسبه می‌شود. اگر f(q) جدید بهتر از f(q) قدیمی باشد (یعنی اگر تکامل قورباغه باعث به وجود آمدن موقعیت بهتری برای قورباغه شود)، U(u) قدیمی با U(q) جدید جایگزین می‌شود و الگوریتم به مرحله ۷ جستجوی محلی می‌رود. اگر تکامل قورباغه حاصل نشود، الگوریتم به مرحله بعد می‌رود.

مرحله ۵- بهبود دادن موقعیت بدترین قورباغه به‌صورت سراسری: اگر در مرحله ۴ جستجوی محلی الگوریتم جهش قورباغه، تکامل محلی باعث به وجود آمدن موقعیت بهتر برای بدترین قورباغه محلی نشود، موقعیت جدید بدترین قورباغه، توسط رابطه (۴) محاسبه می‌شود.

$$\text{Step size } S = \begin{cases} \min\{\text{int}[\text{rand}(P_X - P_W)], Smax\} & \text{برای گام مثبت} \\ \max\{\text{int}[\text{rand}(P_X - P_W)], -Smax\} & \text{برای گام منفی} \end{cases} \quad (۴)$$

همانند مرحله قبل، اندازه و ابعاد گام برابر با تعداد متغیرهای تصمیم‌گیری مموتیپ است و موقعیت جدید با رابطه (۵) محاسبه می‌شود.

$$U(q) = P_w + S \quad (۵)$$

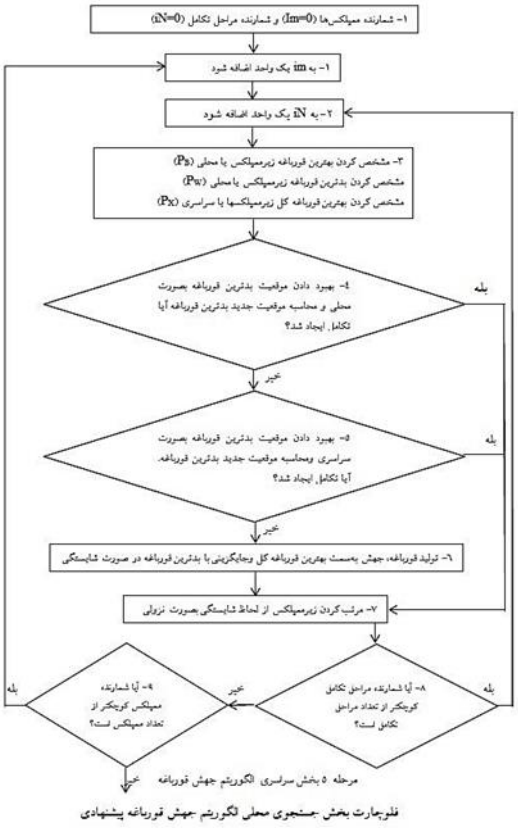
همانند حالت قبل، اگر $U(q)$ یا موقعیت جدید قورباغه در فضای Ω قابل اجرا باشد، مقدار جدید شایستگی قورباغه $f(q)$ محاسبه می‌شود. اگر $f(q)$ جدید بهتر از $f(q)$ قدیمی باشد، یعنی اگر تکامل قورباغه باعث به وجود آمدن موقعیت بهتر برای قورباغه شود، $U(u)$ قدیمی با $U(q)$ جدید جایگزین می‌شود و الگوریتم به مرحله ۷ جستجوی محلی می‌رود. اگر تکامل قورباغه حاصل نشود، الگوریتم به مرحله بعد می‌رود.

مرحله ۶- سانسور: با توجه به دو مرحله تکامل و عدم بهبود موقعیت بدترین قورباغه، گسترش مم معیوب متوقف، و به‌طور تصادفی قورباغه جدید (r) ایجاد می‌شود تا بعد از محاسبه مقدار شایستگی، جایگزین بدترین قورباغه شود.

مرحله ۷- ارتقا ممپلکس: پس از حذف یا تکامل ممتیکی بدترین قورباغه در زیرممپلکس، مجددا قورباغه‌ها به‌صورت نزولی از نظر مقدار شایستگی در هر ممپلکس مرتب می‌شوند.

مرحله ۸- اگر $iN < N$ باشد، به مرحله ۲ جستجوی محلی بازگشت می‌شود.

مرحله ۹- اگر $im < m$ باشد، به مرحله ۱ جستجوی محلی بازگشت می‌شود. در غیر این صورت، به مرحله ۵ از بخش سراسری الگوریتم جهش قورباغه بازگشت می‌شود با توجه به توضیحات ارائه شده فوق فلوچارت بخش جستجوی محلی به شکل زیر می‌باشد.



فلوچارت ۱. بخش محلی الگوریتم جهش قورباغه

در ادامه، روش جهش قورباغه بهبود یافته پیشنهادی (R-SFLA)، با ذکر جزئیات ارائه می‌شود. همچنین در بخش بعد، به منظور ارزیابی روش پیشنهادی، نتایج حاصل شده از آن با روشهای معرفی شده در [۲] و [۳] از نظر معیارهای هزینه اجرا، زمان پاسخ، و درجه عدم توازن بار، مقایسه شده است. با بررسی مقالات ادبیات تحقیق [۲۸،۲۷،۲۶،۳] که همگی بهبودهایی بر الگوریتم جهش قورباغه در کاربردهای مختلف داشته‌اند، استنتاج می‌شود که موارد زیر در بهبود این الگوریتم باید مدنظر قرار گیرند.

الف- مصالحه میان کاوش و بهره‌وری: یکی از مسائل جهت رسیدن به بهینه‌سازی مناسب در الگوریتم جهش قورباغه، مصالحه میان کاوش و بهره‌وری است. در اغلب الگوریتم‌های تکاملی که بر پایه جمعیت هستند، توجه به این نکته حائز اهمیت است چراکه اگر فقط به مسئله کاوش اهمیت داده شود، یک سربار اضافی (جستجو) به سیستم تحمیل می‌شود و با توجه به کاهش بار پردازشی در گره‌های پردازشی، بهره‌وری در کل، کاهش می‌یابد. از سوی دیگر، اگر هدف فقط بهره‌وری باشد، ممکن است سیستم در بهینه‌های محلی گرفتار شود و به دنبال یافتن بهینه سراسری (بهترین حالت ممکن) نباشد. بنابراین باید بین کاوش و بهره‌وری، تعادل برقرار باشد.

ب- دقیق سازی مسیر حرکت: یکی دیگر از مسائل جهت رسیدن به راه‌حل بهینه در الگوریتم جهش قورباغه، حرکت دقیق بدترین قورباغه و قرار گرفتن در مسیر بهترین قورباغه است [۲۸،۳]. به عبارت دیگر، بدترین قورباغه (بدترین راه‌حل) باید مسیری را جهت رسیدن به مسیر حرکت بهترین قورباغه‌ها (بهترین ممپلکس و بهترین کل) طی کند تا در بهترین حالت، خود باعث به وجود آمدن موقعیت بهتری شود یا در مسیر رسیدن به بهترین قورباغه قرار گیرد. اگر این مسیر اشتباه یا کوتاه طی شود، با عنایت به این فرض که بهترین جواب به‌احتمال زیاد در اطراف بهترین قورباغه است، احتمال هرگز نرسیدن یا دیر رسیدن به این اهداف، زیاد است.

پ- پرهیز از بن‌بست: باید از حالتی که در آن، الگوریتم بدون تغییر بماند، پرهیز شود. با فرض اینکه بهترین قورباغه سراسری (بهترین راه‌حل)، قطعی نیست و حالتی بهتر وجود دارد، با توجه به اینکه مرجع حرکت سایر قورباغه‌ها، بهترین قورباغه ممپلکس و یا بهترین قورباغه جمعیت است، احتمال اینکه کل قورباغه‌ها هیچ‌وقت به آن راه‌حل بهتر نرسند، زیاد است. بنابراین باید بهبود طوری باشد که جستجو، منحصر به یک مسیر از پیش تعیین‌شده نشود. در این صورت، احتمال رسیدن به بهترین راه‌حل، بیشتر است.

بهبود پیشنهادی اول با توجه به موارد فوق، بهبود به شرح زیر در الگوریتم جهش قورباغه ارائه می‌شود. همان‌طور که در بند (ب) مربوط به جستجوی محلی الگوریتم جهش قورباغه ذکر شد، جهت بهبود موقعیت بدترین قورباغه، جهش ممتیکی با تأثیر مستقیم از بهترین قورباغه محلی یا سراسری انجام می‌گیرد که این جهش با استفاده از رابطه‌های (۲) و (۴) انجام می‌گرفت. در این رابطه‌ها از یک عدد تصادفی (rand) استفاده می‌شود که دامنه تغییراتش از صفر تا یک است. در اینجا، در دو فاز، متغیر تصادفی rand تغییر می‌یابد. همان‌گونه که در رابطه (۲) ذکر شده است، استفاده از متغیر تصادفی rand به دلایل زیر دارای نقش تعیین‌کننده‌ای است:

الف- اگر متغیر rand در دامنه تغییراتش، نزدیک به صفر باشد و فاصله بهترین قورباغه و بدترین قورباغه زیاد باشد، جهش به سمت بهترین قورباغه با تأثیر از این متغیر، کوچک می‌شود. بر این اساس، ممکن است همگرایی چند مرحله دیرتر انجام شود. جهت روشن شدن بیشتر موضوع، مثال (۱) ارائه می‌شود: مثال (۱): اگر P_W ۱ =و P_B = ۹ و rand=۰/۱ باشد، درنتیجه با توجه به رابطه (۲)، S = ۰/۸ است. با توجه به رابطه (۳)، موقعیت جدید بدترین قورباغه زیرممپلکس $U(q)$ =۱/۸ است و با توجه به نتایج عددی به‌دست‌آمده، جهش به سمت بهترین قورباغه، کوتاه است.

ب- اگر متغیر rand در دامنه تغییراتش، نزدیک به یک باشد و فاصله بهترین قورباغه و بدترین قورباغه زیاد باشد، جهش به سمت بهترین قورباغه می‌تواند بسیار بزرگ باشد به حدی که از بهترین قورباغه دور شود. جهت روشن شدن بیشتر موضوع، مثال (۲) ارائه می‌شود: مثال (۲): اگر P_W = ۹ و P_B =۱ و rand= ۰/۹ باشد، با توجه به رابطه (۲)، S = -۷/۲ می‌شود. با توجه به رابطه (۳)، موقعیت جدید بدترین قورباغه زیرممپلکس $U(q)$ =-۶/۲ است و با توجه به نتایج عددی به‌دست‌آمده، جهش به سمت بهترین قورباغه، آن‌چنان بلند است که از هدف دور می‌شود. درنتیجه، انتظار می‌رود در صورت ارائه یک رابطه جایگزین که باعث گردد $U(q)$ یا موقعیت نهایی بدترین قورباغه جهش‌یافته، نزدیک‌تر به بهترین قورباغه باشد، همگرایی سریع‌تری به دست آید. بنابراین، در فاز اول در رابطه (۲) به جای متغیر تصادفی rand، از متغیر h با رابطه زیر استفاده می‌شود.

$$h=\frac{|P_B - P_W|}{(\text{if } (PB> PW);\text{then } PB;\text{else } PW)} \qquad (۶)$$

رابطه (۶) با هدف به دست آوردن ضریبی که درنهایت باعث جهش مناسب برای بدترین قورباغه گردد، ارائه شده است. در صورت کسر، تفاضل فاصله عددی موتیپ‌های بدترین قورباغه از بهترین قورباغه، و در مخرج کسر، بزرگ‌ترین عدد از بین موتیپ‌های بدترین قورباغه و بهترین قورباغه باهدف به دست آوردن بزرگ‌ترین ضریب کسری در محدوده صفر تا یک اعمال شده‌اند. جهت به دست آوردن عدد مثبت، صورت رابطه در قدر مطلق قرار دارد. حال جهت مقایسه اولیه، بجای متغیر تصادفی rand از ضریب h در مثال (۳) و (۴) با شرایط همسان مثال (۱) و (۲) استفاده می‌شود: مثال (۳): اگر P_W =۱ و P_B = ۹ باشد، با توجه به رابطه (۶)، h = ۸/۹=۰/۸۸ است. درنتیجه با توجه به رابطه (۲)، S = ۷/۰۴ بوده و درنهایت، با توجه به رابطه (۳)، موقعیت جدید بدترین قورباغه زیرممپلکس $U(q)$ = ۸/۰۴ است. مثال (۴): اگر P_W =۹ و P_B =۱ باشد، با توجه به رابطه (۶)، h = ۸/۹=۰/۸۸ است. درنتیجه با توجه به رابطه (۲)، S -۷/۰۴ = بوده و درنهایت، با توجه به رابطه (۳)، موقعیت جدید بدترین قورباغه زیرممپلکس $U(q)$ = -۶/۰۴ است. در جمع‌بندی مطالب عنوان‌شده بالا و مقایسه عددی مثال‌ها، مشاهده می‌شود که در مثال(۳)، فاصله بدترین قورباغه از بهترین قورباغه زیرممپلکس نسبت به مثال (۱) بسیار نزدیک‌تر و در مثال(۴)، فاصله بدترین قورباغه از بهترین قورباغه زیرممپلکس نسبت به مثال (۲)، بافاصله جزئی بهتر است. با جمع‌بندی موارد فوق، روش پیشنهادی R-SFLA، رابطه (۷) را برای جهش قورباغه‌ها در فاز اول در نظر می‌گیرد. همچنین در فاز دوم ، تمام دلایل و مثال‌ها برای رابطه (۴) نیز صادق هستند. بنابراین جهت اصلاح رابطه تکامل در الگوریتم جهش قورباغه، این مقاله رابطه (۸) را پیشنهاد می‌دهد.

$$\text{Step size } S = \min\{\text{int}\,[(\frac{|P_B - P_W|}{(\text{if } (PB> PW);\text{then } PB;\text{else } PW)})*(P_B- P_W)], Smax\} \qquad (۷)$$

برای گام مثبت

$$\max\{\text{int}\,[(\frac{|P_B - P_W|}{(\text{if } (PB> PW);\text{then } PB;\text{else } PW)})*(P_B- P_W)], -Smax\}$$

برای گام منفی

$$\text{Step size } S = \min\{\text{int}\,[(\frac{|P_x - P_W|}{(\text{if } (Px> PW);\text{ then } Px;\text{ else } PW)})*(P_x- P_W)], Smax\} \qquad (۸)$$

برای گام مثبت

$$\max\{\text{int}\,[(\frac{|P_x - P_W|}{(\text{if } (Px> PW);\text{ then } Px;\text{ else } PW)})*(P_x- P_W)], -Smax\}$$

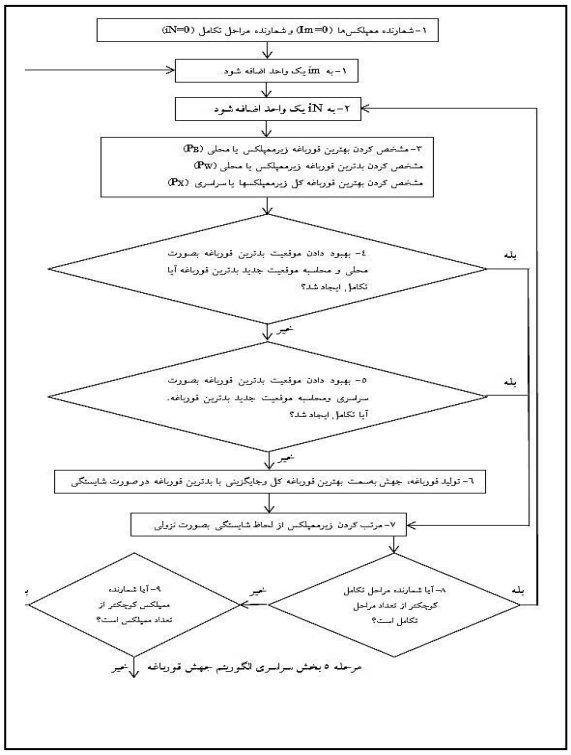
برای گام منفی

لازم به ذکر است که Smax که حداکثر اندازه گام مجاز است، در کلیه حالات، آزاد فرض شده است به این معنی که در محدوده

فضای جستجوی الگوریتم قورباغه میتوان حداکثر اندازه گام را برداشت. رابطه (۷) و(۸) که در آنها متغیر تصادفی rand تغییر یافته است، پیچیدگی محاسبات کمتری داشته و همچنین باعث حرکت به‌صورت مستقیم و منظم به سمت بهترین قورباغه می‌شوند. با توجه به گستردگی محدوده تغییرات رابطه‌های پیشنهادی، انتظار می‌رود هیچگاه به بن‌بست دچار نشده و دارای همگرایی سریع‌تر بوده و زمان پردازش را کاهش دهند.

بهبود پیشنهادی دوم همان‌طور که در مرحله ششم (سانسور) جستجوی محلی الگوریتم جهش قورباغه بیان شد، اگر تلاش‌ها برای بهبود دادن موقعیت بدترین قورباغه هر زیرممپلکس که با تأثیر از بهترین قورباغه زیرممپلکس و بهترین قورباغه کل انجام می‌گرفت، به نتیجه نرسد، به‌ناچار بدترین قورباغه حذف می‌شود و به‌طور تصادفی یک قورباغه جدید (r) ایجاد و جایگزین قورباغه‌ای در فضای جستجوی مسئله می‌شود که تکامل مناسبی نداشته است.

احتمال اینکه قورباغه تازه ایجادشده دارای موقعیت مناسب نباشد وجود دارد؛ درنتیجه موقعیت قورباغه تازه ایجادشده در نزدیک‌ترین موقعیت بهترین قورباغه کل درنظر گرفته می‌شود و در صورت داشتن شایستگی بالاتر، جایگزین قورباغه سانسور شده می‌شود. درنتیجه شایستگی f (r) از طریق رابطه P(r)= (rand ((px) محاسبه می‌شود. این رابطه جهت ایجاد موقعیت نزدیک در همسایگی بهترین قورباغه کل ارائه شده است و تمام عناصر به‌جز متغیر تصادفی rand ، تعریف شده‌اند. جهت نزدیکی به بهترین قورباغه سراسری، بازه متغیر تصادفی rand در محدوده ۰/۸ تا ۱ در نظر می‌شود و احتمال اینکه بهترین حالت در نزدیکی بهترین قورباغه کل باشد، بیشتر است. با توجه به اینکه ایجاد قورباغه با تکامل انجام‌گرفته، انتظار می‌رود همگرایی سریع باشد (زمان پردازش کاهش یابد). بنابراین فلوچارت بخش جستجوی محلی الگوریتم جهش قورباغه بهبود یافته به شکل زیر می‌باشد.



فلوچارت ۱. بخش محلی الگوریتم قورباغه بهبود یافته

۴. محیط و پارامترهای شبیه‌سازی

کلیه آزمایش‌ها در این تحقیق، با استفاده از نرم‌افزار شبیه‌ساز کلودسیم انجام می‌شود که یک ابزار معتبر جهت شبیه‌سازی سناریوهای مرتبط با رایانش ابری است و در محیط توسعه زبان جاوا NetBeans، تحت سیستم عامل ویندوز انجام شده‌اند. جهت شبیه‌سازی، شش ماشین مجازی با توانهای مختلف استفاده شده است.

هر موجودیت در کلودسیم شامل چند پارامتر است که برای پیکربندی موجودیت‌ها، باید تعدادی از این پارامترها متناسب با موضوع شبیه سازی تنظیم شود. این پارامترها عبارتند از:

الف – bw : پهنای باند را در موجودیت های مختلف (Datacenter, Vm, Host)، شبیه‌سازی می‌کند.

ب- pesNumber : تعداد CPU را در موجودیت های مختلف، شبیه‌سازی می‌کند.

پ- ram : میزان حافظه جانبی در موجودیت های مختلف، را شبیه‌سازی می‌کند.

ت- mips : توانایی پردازش بر حسب میلیون دستورالعمل در ثانیه، در عناصر پردازشی موجودیت‌های مختلف کلودسیم را شبیه‌سازی می‌کند.

ث- cloudletScheduler: بیانگر سیاست اختصاص توان پردازشی است که به دو صورت زمان اشتراکی (TimeShared) و فضا اشتراکی(SpaceShared) می‌باشد. در سیاست زمان اشتراکی، درخواستها به کل توان پردازشی با محدویت زمانی دسترسی دارند و در سیاست فضا اشتراکی، درخواستها به بخشی از توان پردازش با زمان نامحدود دسترسی دارند.

ج- Cloudletlengh: تعداد دستورالعمل‌های یک درخواست، که توسط ماشین مجازی پردازش می‌شود را شبیه‌سازی می‌کند.

چ- vmm : سیستم ناظر بر ماشین مجازی را شبیه‌سازی می‌کند.

ح- cloudletFileSize :حجم فایل درخواست کاربر را قبل از اجرا، شبیه‌سازی می‌کند.

خ- cloudletOutputSize : حجم فایل خروجی درخواست کاربر را بعد از اجرا، شبیه‌سازی می‌کند.

د- Pe : عناصر پردازشی ماشین میزبان را، شبیه‌سازی می‌کند.

ذ- CloudletList : صف درخواست کاربر را شبیه‌سازی می‌کند.

ر- VMList : صف ماشین‌های مجازی را شبیه‌سازی می‌کند.

جهت هریک از سه الگوریتم (R-SFLA, ASFLA, SFLA) مورد مقایسه، شش ماشین مجازی استفاده شده است و با توجه به ماهیت شبیه سازی، این شش ماشین دارای توانهای مختلفی هستند. مقادیر جدول زیر در شرایط یکسان برای هر سه الگوریتم در شبیه ساز کلود سیم تعریف شده است.

جدول ۲. تنظیمات مربوط به پارامترهای ماشینهای مجازی جهت هر سه الگوریتم

مشخصات تنظیمات ماشین‌های مجازی								
algorithm	vmid	brokerId	mips	pesNumber	ram	bw	vmm	Cloudlet Scheduler
SFLA	1	brokerId1	130	1	2048	1000	zen	TimeShared
	2	brokerId2	260	1	2048	1000	zen	TimeShared
	3	brokerId3	440	1	2048	1000	zen	TimeShared
	4	brokerId4	1000	1	2048	1000	zen	TimeShared
	5	brokerId5	5300	1	2048	1000	zen	TimeShared
	6	brokerId6	12200	1	2048	1000	zen	TimeShared
ASFLA	7	brokerId7	130	1	2048	1000	zen	TimeShared
	8	brokerId8	260	1	2048	1000	zen	TimeShared
	9	brokerId9	440	1	2048	1000	zen	TimeShared
	10	brokerId10	1000	1	2048	1000	zen	TimeShared
	11	brokerId11	5300	1	2048	1000	zen	TimeShared
	12	brokerId12	12200	1	2048	1000	zen	TimeShared
R-SFLA	13	brokerId13	130	1	2048	1000	zen	TimeShared
	14	brokerId14	260	1	2048	1000	zen	TimeShared
	15	brokerId15	440	1	2048	1000	zen	TimeShared
	16	brokerId16	1000	1	2048	1000	zen	TimeShared
	17	brokerId17	5300	1	2048	1000	zen	TimeShared
	18	brokerId18	12200	1	2048	1000	zen	TimeShared

در ادامه جهت ارزیابی عملکرد جداول مرتبط با تنظیمهای متغیرهای الگوریتم جهت شبیه سازی با حدود و دامنه یکسان بشرح زیر می‌باشد.

جدول ۳. تنظیمهای پارامترهای صف درخواست کاربر

مشخصات صف درخواست					
CloudletList	Cloudletid	Cloudletlengh	pesNumber	fileSize	outputSize
LCG-2005-1	var	var (17-300000)	1	300	300

جدول ۴. تنظیمهای پارامترهای ماشین میزبان

مشخصات ماشین میزبان							
number	hostId	mips	pe	ram	bw	storage	VmScheduler
1	1	1000000	18	204800	100000	1000000	SpaceShared

جدول ۵. تنظیمهای پارامترهای دیتاسنتر

مشخصات دیتا سنتر					
number	arch	os	vmm	time_zone	cost
1	x64	Linux	Xen	10.0	var

جدول ۶. هزینه اجرایی بر مبنای هر ساعت استفاده

مجموع هزینه‌های استفاده از هر ماشین مجازی						
number vmid	1,7,13	2,8,14	3,9,15	4,10,16	5,11,16	6,12,18
OEC ($)	4,050	4,590	4,671	5,508	5,589	5,670

۵. ارزیابی عملکرد روش پیشنهادی (R-SFLA)

در این بخش، نتایج پیاده‌سازی روش پیشنهادی ارائه شده و مقایسه‌ای بین الگوریتم جهش قورباغه بهبودیافته این مقاله (R-SFLA) الگوریتم ASFLA [۳] یک نوع دیگر ازالگوریتم جهش قورباغه بهبود یافته که با الگوریتم پرندگان والگوریتم جهش قورباغه مقایسه شده و الگوریتم SFLA [۲] که همان الگوریتم جهش قورباغه انجام می‌گیرد. دلیل انتخاب این سه الگورریتم جهت مقایسه، نشان دادن بهبود یافتگی در این الگوریتم نسبت به کارهای گذشته و همچنین بدست اوردن نتایج بهتر در استفاده از الگوریتم جهش قورباغه بهبودیافته در توازن بار میباشد نتایج شبیه‌سازی بدست آمده از نظر معیارهای زمان پاسخ، درجه توازن بار و هزینه کلی اجرا، مورد تجزیه و تحلیل قرار می‌گیرند.

جهت شبیه‌سازی، شش ماشین مجازی با توانهای مختلف استفاده شده است. جهـــت هر سه الگوریتم R-SFLA، ASFLA ، و SFLA، جهت صف درخواست‌ها از دیتاست استاندارد LCG-

2005-1استفاده می‌شود [۴۰] که تداعی حالت استـــاندارد از درخواست‌های رایانش ابری می‌باشد. تنظیمهای مربوط به متغیرهای الگوریتم جهش قورباغه و مولفه های اصلی در آزمایشات انجام شده به شرح زیر است:

الف- تعداد قورباغه: تعداد قورباغه‌ها چهل عدد است که به چهار ممپلکس ده عددی تقسیم شده‌اند. این چهل عدد قورباغه به‌صورت تصادفی تولید شده‌اند و هر سه الگوریتم، جهت مقایسه از یک نسخه واحد استفاده می‌کنند.

ب- تعداد iteration : همان متغیر iN است که در مرحله صفر بند ب (جستجوی محلی الگوریتم جهش قورباغه) معرفی شده است. منظور از آن، طی دوره تکامل در الگوریتم جهش قورباغه است که در این مقاله، ۱۰۰۰ در نظر گرفته شده است.

پ- شایستگی: مقادیر شایستگی قورباغه‌ها از طریق رابطه (۹) محاسبه می‌شود:

$$\text{fitness}=\frac{(\text{mipsvmi}*\text{pesNumbervmi})}{\text{Cloudletlengh}} \qquad (۹)$$

در این رابطه، صورت کسر قدرت پردازش ماشین مجازی و مخرج کسر تعداد دستورالعمل‌های یک درخواست که باید پردازش شود، است. شایستگی هرقورباغه از طریق مجموع شایستگی‌های هر شش مموتیپ یک قورباغه، محاسبه می‌شود.

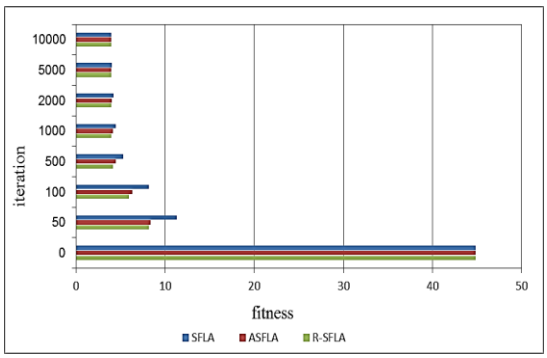
ت - درجه عدم توازن: جهت محاسبه این معیار، حداقل زمان کارکرد از حداکثر زمان کارکرد در بین هر شش ماشین مجازی مربوط به شبیه سازی هر کدام از این الگوریتم‌ها، کسر می‌گردد. سپس عدد بدست آمده بر میانگین زمان کارکرد تقسیم می‌شود. حاصل این عملیات، درجه عدم توازن بار است که این عدد هر اندازه کوچکتر باشد، توازن بار مناسبتر و هر اندازه بزرگتر باشد نشانه عدم توازن بار است.

ث- همگرایی : در شبیه‌سازی تحقیق حاضر، دو فاکتور (رسیدن به میانگین $\frac{1}{15}$ شایستگی قورباغه‌های تصادفی اولیه، سپری کردن ۱۰۰۰ دوره تکامل) جهت همگرایی تعریف شده است و محقق شدن هر کدام از این شرایط باعث توقف بهبودسازی موقعیت قورباغه‌ها می‌شود.

در این تحقیق از چهار سناریو جهت مقایسه استفاده شده است. آزمایش با هرکدام از بارهای کاری (چهار بار کاری ۵۰،۱۰۰،۱۵۰ و۲۰۰ درخواست کاربر)، بعنوان یک سناریو مطرح می‌گردد و سه الگوریتم نامبرده شده از نظر معیارهای ۱) کلیه هزینه‌های اجرا (OEC) ۲) حداکثر زمان تکمیل اجرا (Makespan) ۳) زمان پاسخ (Response time) ۴) درجه عدم توازن بار(degree of imbalance) تحت ارزیابی قرار می‌گیرند.

همگرایی سریع و به حداقل رساندن حلقه‌های معیوب در تکامل در هر سه الگوریتم، با وضعیت همگرایی و درجه شایستگی قورباغه‌ها نسبت مستقیم دارد چرا که، سپردن درخواست‌های کاربر به ماشین‌های مجازی، وضعیت کل نتایج را مشخص می‌نماید. همچنین با توجه به اینکه مم هر قورباغه تعیین می‌کند درخواست کاربر به‌کدام ماشین مجازی سپرده شود، قبل از هر گونه بررسی نتایج، وضعیت مم قورباغه‌ها با پارامتر متوسط شایستگی در دوره‌های مختلف تکامل سنجیده می‌شود تا مناسب‌ترین تکرار دوره تکامل جهت شروع شبیه سازی انتخاب شود.

شکل (۲) نمودار نتایج بدست آمده از متوسط شایستگی مم چهل عدد قورباغه، در هشت مرحله با تعداد تکرار مختلف (۰ تا ۱۰،۰۰۰) را نشان می‌دهد.

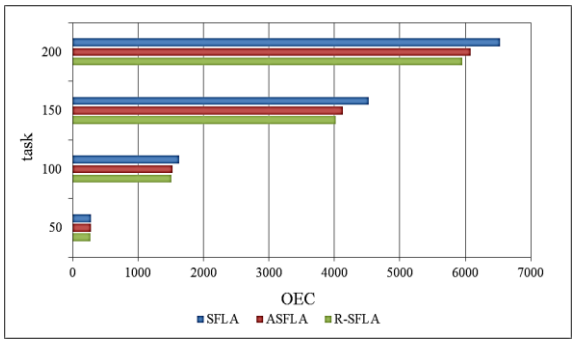


شکل ۲. نمودار مقایسه متوسط شایستگی در دوره های مختلف تکامل

در شکل (۲)، محورهای عمودی و افقی به‌ترتیب نشان‌دهنده تکرار دوره تکامل و شایستگی هستند. شایستگی قورباغه‌های تصادفی در شروع برابر با عدد ۴۴/۸ است و با بالا رفتن تکرار دوره تکامل، الگوریتم R-SFLA زودتر از الگوریتمهای ASFLA و SFLA به همگرایی می‌رسد. دلیل این امر آن است که الگوریتم R-SFLA، از گسترش یک حلقه تکرار معیوب جلوگیری می‌کند. همچنین بعد از تکرار دوره تکامل ۱۰۰۰، رشد شایستگی هر سه الگوریتم، به کمترین حد ممکن می‌رسد. با توجه به نتایج، الگوریتم R-SFLA به‌طور متوسط ازلحاظ رسیدن به همگرایی نسبت به الگوریتم‌های SFLA و ASFLA، به ترتیب ۲۸ و ۵ درصد بهبود عملکرد نشان می‌دهد.

یکی از مهمترین ویژگی‌های هر الگوریتم توازن بار در حوزه رایانش ابری، کلیه هزینه‌های اجرا برای سرویس‌دهنده است. توازن بار مناسب در رایانش ابری، در مرحله اول باعث پائین آمدن کلیه هزینه‌های اجرا برای سرویس‌دهنده و متعاقبا برای سرویس‌گیرنده

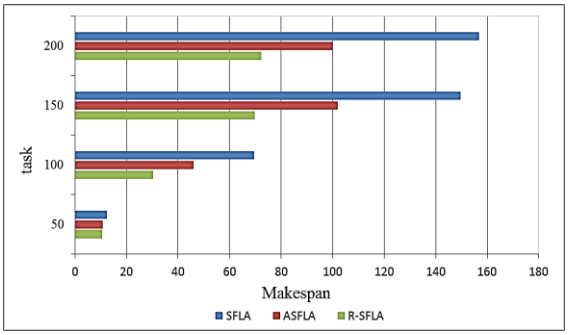
خواهد شد. تقسیم مناسب منابع پردازشی و سپردن درخواست کاربر به ماشین متناسب با این درخواست، باعث پائین آمدن این پارامتر در رایانش ابری می‌شود. شکل (۳) نتایج بدست آمده از الگوریتم‌های R-SFLA, ASFLA, SFLA در اندازه‌گیری معیـار کلـیه هزینه‌هـــای اجرا را نشان می‌دهد.



شکل ۳. نمودار مقایسه کلیه هزینه‌های اجرا در شرایط مختلف کاری

در شکل (۳)، محور عمودی نشان‌دهنده تعداد درخواست کاربر و محور افقی نشان دهنده کلیه هزینه‌های اجرا بر حسب دلار است. با بررسی نمودار مقایسه کلیه هزینه‌های اجرا، ملاحظه می‌شود که الـگوریتم‌های R-SFLA و ASFLA با بالا رفتن تعداد درخواست کاربر نسبت به الگوریتم SFLA، هزینه‌های اجرای پائین‌تری دارند و هزینه‌های اجرا نسبت به تعداد درخواست کاربر شیب تصاعدی دارد. دلیل این امر آن است که، قدرت پردازشی در ماشین‌های مجازی این شبیه‌سازی به‌صورت زمان اشتراکی است و با افزایش درخواست کاربر، سرعت پردازش کاهش می‌یابد. بنابراین، زمان پردازش درخواست کاربر بیشتر شده و کلیه هزینه-های اجرا، افزایش می‌یابد. بنابراین الگوریتم R-SFLA به‌طور متوسط ازلحاظ هزینه‌های کلی اجرا نسبت به الگوریتم‌های SFLA و ASFLA به ترتیب ۸ و ۲ درصد بهبود عملکرد نشان می‌دهد.

یکی دیگر از پارامترهای ارزیابی الگوریتم‌های توازن بار، حداکثر زمان تکمیل اجرای درخواستهای کاربر است. در ادامه، میزان حـــداکثر زمان تکمیل الگوریتم‌های R-SFLA,ASFLA SFLA, مورد ارزیابی قرار می‌گیرد. شکل (۴) نتایج بدست آمده از این سه الگوریتم با معیار حداکثر زمان تکمیل را نشان می‌دهد.

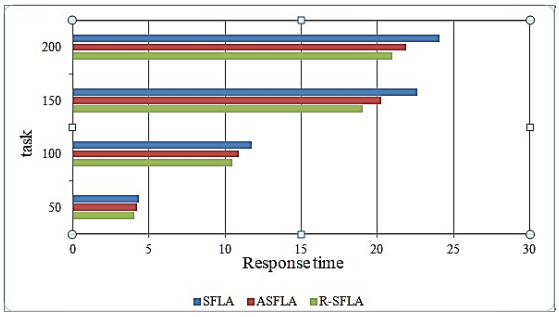


شکل ۴. نمودار مقایسه حداکثر زمان تکمیل در شرایط مختلف کاری

در شکل (۴)، محورهای عمودی و افقی به‌ترتیب نشان‌دهنده تعداد درخواست کاربر و حداکثر زمان تکمیل بر حسب ثانیه هستند. این نمودار نتایج شبیه‌سازی الگوریتم‌های R-SFLA, ASFLA, SFLA را در چهار حالت کاری، با پارامتر حداکثر زمان تکمیل نمایش می‌دهد. الگوریتم R-SFLA با بالا رفتن تعداد درخواست کاربر، نسبت به الگوریتم ASFLA و SFLA حداکثر زمان تکمیل اجرا کمتری دارد و این پیشرفت تصاعدی بدلیل رسیدن به شرایط ارزیابی صحیح است.

حداکثر زمان تکمیل بعد از ۱۵۰ درخواست کاربر در هر سه الگوریتم نزدیک به حالت ثبات است و تغییرات چندانی ندارد؛ دلیل این امر آن است که حداکثر زمان تکمیل به حالت بهینه نزدیک شده است. بنابراین، الگوریتم R-SFLA به‌طور متوسط ازنظر حداکثر زمان تکمیل نسبت به الـگوریتم‌های SFLA و ASFLA به ترتیب ۹۴ و ۳۵ درصـد بهبود عملکرد نشان می‌دهد.

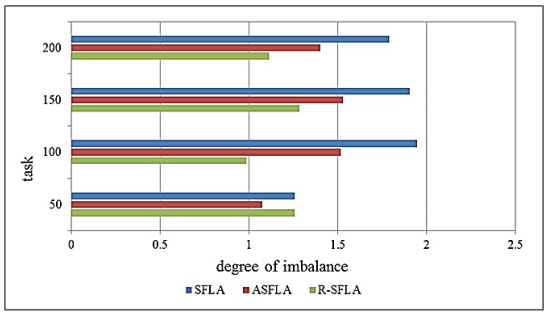
معیار دیگری که در رایانش ابری مد نظر کاربران قرار می‌گیرد و کلیه سرویس‌گیرندگان رایانش ابری با آن در ارتباط مستقیم هستند، زمان پاسخ است. در ادامه، میزان متوسط زمان پاسخ، که از نتایج شبیه سازی الگوریتم‌های R-SFLA, ASFLA, SFLA به دست آمده است مورد ارزیابی قرار می‌گیرد. شکل (۵) نمودار نتایج به دست آمده از معیار حداکثر زمان تکمیل اجرای درخواستهای کاربر را نشان می‌دهد.



شکل ۵. نمودار مقایسه میانگین زمان پاسخ در شرایط مختلف کاری

در شکل (۵)، محور عمودی نشان‌دهنده تعداد درخواست کاربر و محور افقی نشان دهنده میانگین زمان پاسخ بر حسب ثانیه است. نتایج شبیه‌سازی در چهار حالت کاری، با پارامتر میانگین زمان پاسخ نمایش داده شده‌اند. ملایم شدن شیب پیشرفت نمودار بعد از ۱۵۰ درخواست کاربر بعلت نزدیک شدن به حالت بهینه زمان پاسخ است. با توجه به نتایج کسب شده، می‌توان نتیجه گرفت که الگوریتم R-SFLA به‌طور متوسط از نظر زمان پاسخ نسبت به الگوریتم‌های SFLA و ASFLA به ترتیب ۱۳ و ۴ درصد بهبود عملکرد دارد.

در ادامه، درجه عدم توازن بار روش پیشنهادی بررسی می‌شود که توسط آن، وضعیت تقسیم مناسب درخواست کاربران بین منابع رایانشی ارزیابی می‌گردد. در صورت تقسیم مناسب وظایف، این درجه به صفر نزدیک می‌شود. شکل (۶) نتایج مقایسه الگوریتم‌های SFLA ، ASFLA و R-SFLA از نظر معیار درجه عدم توازن بار را نشان می‌دهد.



شکل ۶. نمودار مقایسه درجه عدم توازن بار در شرایط مختلف کاری

در شکل (۶)، محور عمودی نشان‌دهنده تعداد درخواست کاربر و محور افقی نشان دهنده درجه عدم توازن بار است. درجه عدم توازن بار در الگوریتم R-SFLA بین ۱ تا ۱/۲۵ می‌باشد که نشان دهنده اختلاف کم درجه عدم توازن بار در شرایط مختلف کاری نسبت به دو الگوریتم دیگر است. بنابراین، می‌توان نتیجه گرفت الگوریتم R-SFLA در شرایط مختلف تعداد درخواست کاربر، منابع رایانشی را عادلانه تقسیم می‌کند که این تقسیم مناسب درخواست کاربر مابین ماشین‌های مجازی درنهایت منجر به کاهش میانگین زمان پاسخ و حداکثر زمان تکمیل اجرا می‌شود. الگوریتم R-SFLA به‌طور متوسط از نظر درجه عدم توازن بار نسبت به الگوریتم‌های SFLA و ASFLA به ترتیب ۵۱ و ۲۱ درصد بهبود عملکرد دارد.

بررسی آزمایش‌های انجام شده نشان می‌دهد که الگوریتم R-SFLA نسبت به دو الگوریتم مورد مقایسه، دارای عملکرد مناسب‌تری است چراکه همگرایی آن سریعتر بوده و حلقه تکامل معیوب را زودتر مسدود می‌نماید.

۵. نتیجه‌گیری

این مقاله، زمان رسیدن به همگرایی در الگوریتم جهش قورباغه را که وظیفه تقسیم منابع رایانشی را در متوازن کننده بار سرورهای مجازی رایانش ابری بر عهده دارد، بهبود داده است. در راستای این هدف، تقسیم بهتر منابع محاسباتی و کاهش زمان پاسخ‌گویی به درخواست کاربر در توازن بار تأمین گردید. کاهش زمان رسیدن به همگرایی در الگوریتم جهش قورباغه باعث شد که معیارهای ارزیابی (زمان پاسخ، درجه عدم توازن بار و هزینه کلی اجرا) بهبود یابند چراکه زمان پاسخ‌گویی به درخواست کاربر، تأثیر مستقیم بر این فاکتورها دارد.

در مقایسه بین الگوریتم‌های R-SFLA، ASFLA، و SFLA در چهار حالت از تعداد درخواست کاربر (۵۰،۱۰۰،۱۵۰،۲۰۰)، نتایج زیر بدست آمد: ۱) الگوریتم R-SFLA از نظر رسیدن به همگرایی نسبت به الگوریتم‌های ASFLA و SFLA ، به ترتیب ۲۸ و ۵ درصد بهبود عملکرد نشان داد ۲) الگوریتم R-SFLA از نظر هزینه‌های کلی اجرا نسبت به الگوریتم‌های ASFLA و SFLA ، به ترتیب ۸ و ۲ درصد بهبود عملکرد نشان داد ۳) الگوریتم R-SFLA از نظر حداکثر زمان تکمیل اجرا نسبت به الگوریتم‌های ASFLA و SFLA، به ترتیب ۹۴ و ۳۵ درصد بهبود عملکرد نشان داد ۴) الگوریتم R-SFLA از نظر متوسط زمان پاسخ نسبت به الگوریتم‌های ASFLA و SFLA ، به ترتیب ۱۳ و ۴ درصد بهبود عملکرد نشان داد ۵) الگوریتم R-SFLA از نظر درجه عدم توازن بار نسبت به الگوریتم‌های ASFLA و SFLA ، به ترتیب ۵۱ و ۲۱ درصد بهبود عملکرد نشان داد.

با توجه به این که در این تحقیق کلیه ماشینها و توان پردازشی یکسان و محدود بود پیشنهاد می‌شود در کارهای آینده،

ماشین‌های مجازی نامحدود با توانهای پردازشی متفاوت مورد بررسی قرار گرفته و کارایی سیستم بر اساس معیارهای ذکر شده در این تحقیق، مورد آزمون قرار گیرد. همچنین می‌توان کلیه پارامترهای سرورهای ابری را متفاوت در نظر گرفت و درخواست‌های کاربران که نیازهایی به جز پردازش دارند، را ارزیابی نمود.

مراجع

[1] Mell. P , T. Grance.2011.The Nist Definition Of Cloud Computing. Pp.1-2-3

[2] Muzaffar Eusuff , Kevin Lansey & Fayzul Pasha (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Engineering Optimization, 38:2.

[3] Parmeet Kaur, S. M. (2017). Resource Provisioning and Work flow Scheduling in Clouds using Augmented Shuffled. Journal of Parallel and Distributed Computing-Elsevier .

[4] Mina Nabi, M. T. 2015. Availability In The Cloud: State Of The Art. Journal Of Network And Computer Applications .

[5] Klaithem Al Nuaimi, N. M. J. 2012. A Survey Of Load Balancing In Cloud Computing: Challenges And Algorithms. IEEE Second Symposium On Network Cloud Computing And Applications .p138

[6] N¨Ageli, T. R.-H. 2002. Heterogeneous Dynamic Load Balancing. Springer .

[7] Jaiswal, R. M. 2012. Ant Colony Optimization: A Solution of Load Balancing In Cloud. International Journal of Web & Semantic Technology .

[8] Shang-Liang Chen, Y.-Y. C.-H. 2016. Clb: A Novel Load Balancing Architecture and Algorithm for Cloud Services. Computers and Electrical Engineering.p2 .

[9] Jagadev, B. S. 2018. Cloud Computing For Optimization: Foundations, Applications, and Challenges. Springer .

[10] Einollah Jafarnejad Ghomi, A. M. 2017. Load-Balancing Algorithms in Cloud Computing: A. Yjnca, 62-63 .

[11] Shah, S. A. 2015. Load Balancing Algorithms In Cloud Computing: A Survey Of Modern Techniques. National Software Engineering ConferenceIEEE.p31 .

[12] Minxian Xu, W. T. 2017. A Survey On Load Balancing Algorithms For Virtual Machines Placement In Cloud Computing. Wiley.p6

[13] Geethu Gopinath P P1, S. K. 2015. An In-Depth Analysis and Study of Load Balancing Techniques in the Cloud Computing Environment. Procedia Computer Science-Elsevier, 428

[14] Kumar, D. V. 2015. An Assessment On Various Load Balancing Techniques In Cloud Computing. Ijaict .

[15] Violetta N. Volkova1, L. V. 2018. Load Balancing In Cloud Computing. IEEE.p378 .

[16] Nguyen Khac Chien, N. H. 2016. Load Balancing Algorithm Based On Estimating Finish Time Of Services In Cloud Computing.18Th International Conference on Advanced Communication Technology Icact. IEEE

[17] Singh, S. B. 2014. A Survey On Scheduling And Load Balancing Techniques In Cloud Computing Environment. 5Th International Conference on Computer and Communication Technology Iccct IEEE.p89 .

[18] Samuel, K. R. 2016. Enhanced Bee Colony Algorithm For Efficient Load Balancing And Scheduling In Cloud. Springer International Publishing Switzerland.p67-77

[19] Alireza Sadeghi Milani, N. J. 2016. Load Balancing Mechanisms and Techniques in the Cloud Environments: Systematic Literature Review and Future Trends. Journal of Network and Computer Applications, 9 .

[20] Rami N. Khushaba, A. A.-A.-J. 2008. A Combined Ant Colony and Differential Evolution Feature Selection Algorithm. Springer-Verlag Berlin Heidelbergp2-11 .

[21] Hussain, F. R. 2013. Task-Based System Load Balancing In Cloudcomputing Using Particle Swarm Optimization. Springer, Non Page .

[22] Kushwah, S. S. 2016. A Genetic Based Improved Load Balanced Min-Min Task Scheduling Algorithm For Load Balancing In Cloud Computing. 8Th International Conference on Computational Intelligence and Communication Networks .p678-679 .

[23] Albert Y. Zomaya, S. M.-H. 2001. Observations on Using Genetic Algorithms for Dynamic Load-Balancing. IEEE

Transactions on Parallel and Distributed Systems, 900 .

[24] Mohammed Abdullahi, M. A. 2014. Symbiotic Organism Search Optimization Based Task Scheduling In Cloud Computing Environment. Future Generation Computer Systems.p4-3 .

[25] http://www.cs.huji.ac.il/labs/parallel/workload/l_lcg/index.html

[26] Babak Amiri & Mohammad Fathian & Ali Maroosi. (2007). Application of shuffled frog-leaping algorithm on clustering. The International Journal of Advanced Manufacturing Technology, 199-208 .

[27] Emad Elbeltagiy, T. H. (2007). A modified shuffled frog-leaping optimization algorithm: applications to project management. Structure and Infrastructure Engineering, 53-60 .

[28] S. Sarathambekai, K. U.-G. (2015). Shuffled Frog Leaping Algorithm in Distributed System. International Conference on Innovations in Computing Techniques (ICICT 2015). International Journal of Computer Applications.

[29] Kumar, D., Mandal, N. and Kumar, Y., 2023. Cloud-Based Advanced Shuffled Frog Leaping Algorithm for Tasks Scheduling. Big Data.

[30] Saif, M.A.N., Niranjan, S.K., Murshed, B.A.H., Ghanem, F.A. and Ahmed, A.A.Q., 2023. CSO-ILB: chicken swarm optimized inter-cloud load balancer for elastic containerized multi-cloud environment. The Journal of Supercomputing, 79(1), pp.1111-1155.

[31] Rajakumari, K., Kumar, M.V., Verma, G., Balu, S., Sharma, D.K. and Sengan, S., 2022. Fuzzy Based Ant Colony Optimization Scheduling in Cloud Computing. Computer Systems Science & Engineering, 40(2).

[32] Karpagam, M., Geetha, K. and Rajan, C., 2020. A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques. Soft Computing, 24(1), pp.637-646.

[33] Sudha, R., Indirani, G. and Selvamuthukumaran, S., 2020. Hybridization of Improved Shuffled Frog Leaping Algorithm with Elastic Neural Network for Fog Enabled Cloud based Intelligent Resource Management. Journal of Green Engineering, 10, pp.7602-7620.

[i] Rapid-Shuffled Frog Leaping Algorithm
[ii] Cloud Computing
[iii] NIST: National Institute Of Standards And Technology
[iv] SFL = Shuffled Frog Leaping
[v] Nodes
[vi] Virtual Machine
[vii] Load Balancer
[viii] Availability
[ix] Mechanisms
[x] Fault Tolerance
[xi] Redundancy
[xii] Overload Protection
[xiii] Performance
[xiv] Delay
[xv] Down
[xvi] Flexibility
[xvii] Scheduling Algorithm
[xviii] Replication
[xix] Point of failure
[xx] Centralized
[xxi] Decentralized
[xxii] Single point of failure
[xxiii] Distributed
[xxiv] Homogeneous
[xxv] Heterogeneous
[xxvi] Sender Initiated
[xxvii] Receiver Initiated
[xxviii] Symmetric
[xxix] RR=Round Robin
[xxx] Min-Min
[xxxi] Max-Min
[xxxii] TA =Throttled
[xxxiii] AMLB =Active Monitoring Load Balancing
[xxxiv] منظور از دو سطح، نحوه توازن بار است که در سطح ماشین میزبان یا در ماشین مجازی انجام گردد.
[xxxv] ESCE =Equally spread current execution
[xxxvi] BA=Bee Algorithm
[xxxvii] ACO =Ant Colony Optimization
[xxxviii] PSO =Particle Swarm Optimization
[xxxix] GA =Genetic Algorithm
[xl] SOS=Symbiotic Organism Search
[xli] DCMM-MTS=Deadline-Constrained Make-span Minimization for Multi-Task Scheduling
[xlii] Dynamic Weighted Round-Robin algorithm
[xliii] Meme
[xliv] Memotype
[xlv] Fitness
[xlvi] CloudSim