Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 13, Number 4, Winter 2025, pp. 65-79
Received: 08/01/2024, Revised: 03/05/2025, Accepted: 03/10/2025

Scheduling IoT Tasks in Cloud Computing Using Equilibrium Optimizer Algorithm

Mobin Yousefi[1], Jamal Zarepour-Ahmadabadi[2*] and Sepehr Ebrahimi Mood[3]
1- Department of Computer Science, Yazd University, Yazd, Iran.
2*- Department of Computer Science, Yazd University, Yazd, Iran.
3- Department of Computer Science, Yazd University, Yazd, Iran.
[1] Mobinyousefi.cs@gmail.com, [2*] Zarepourjamal@yazd.ac.ir, and [3] S.ebrahimi@yazd.ac.ir

Corresponding author's address: Jamal Zarepour-Ahmadabadi, Yazd, Safaiyeh, University Blvd, Yazd University, Department of Computer Science.

Abstract- The Internet of Things (IoT) describes the global and universal connection of objects to the Internet, enabling them to function as smart devices capable of sensing their environment, transmitting data, processing it, and providing feedback. With the expansion of IoT-based applications, the volume of processing requests has increased significantly, posing new challenges for managing and processing this data. Cloud computing enables the execution of complex computations in a short time and at an affordable cost. Efficient resource management and cost reduction can be achieved through metaheuristic algorithms such as the Equilibrium Optimizer (EO), which is inspired by physical principles. In this paper, a multi-objective optimization problem is formulated, encompassing criteria such as makespan, delay, load balancing, stability, energy consumption, and processing cost in an integrated cloud environment. The problem is solved using the EO algorithm, and the results are compared with prominent metaheuristic algorithms. Simulation results demonstrate significant improvements achieved by the proposed algorithm. Specifically, in terms of makespan, the proposed algorithm outperforms PSO, GSA, GWO, and FA by 9%, 6.2%, 9%, and 11%, respectively. Regarding delay reduction, it achieves 8%, 6.7%, 9.4%, and 9% better performance compared to the same algorithms. The algorithm also shows improvements of 1.3% to 2% in processing cost and reductions of 2% to 3.6% in energy consumption. Overall, for simultaneous optimization of all objectives, the proposed algorithm achieves 5.6% to 10% better performance than its competitors. These results highlight the superiority of the proposed algorithm in optimizing multiple objectives and enhancing the efficiency of cloud computing systems.

Keywords- Meta-heuristics algorithms, Internet of things, Equilibrium optimizer, Task scheduling, Makespan, Quality of service, Cloud computing.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۳، شماره ۴، زمستان ۱۴۰۳ ، صفحه ۶۵-۷۹
دریافت: ۱۴۰۳/۰۵/۱۱، بازنگری: ۱۴۰۳/۱۲/۱۵، پذیرش: ۱۴۰۳/۱۲/۲۰

زمان‌بندی وظایف اینترنت اشیا در محاسبات ابری با استفاده از الگوریتم بهینه‌ساز تعادل

مبین یوسفی[۱]، جمال زارع‌پور احمدآبادی*[۲]، سپهر ابراهیمی مود[۳]
۱- گروه علوم کامپیوتر، دانشگاه یزد، یزد، ایران.
*۲- گروه علوم کامپیوتر، دانشگاه یزد، یزد، ایران.
۳- گروه علوم کامپیوتر، دانشگاه یزد، یزد، ایران.
[1] mobinyousefi.cs@gmail.com, [2*] zarepourjamal@yazd.ac.ir, [3] s.ebrahimi@yazd.ac.ir

* نشانی نویسنده مسئول: جمال زارع‌پور احمدآبادی، یزد، صفاییه، بلوار دانشگاه، دانشگاه یزد، گروه علوم کامپیوتر.

چکیده- اینترنت اشیاء ارتباط جهانی و همگانی اشیاء را با شبکه اینترنت توصیف می‌کند و امکان تبدیل اشیاء به دستگاه‌های متصل به اینترنت را فراهم می‌سازد. این اشیاء قادر به درک محیط، انتقال داده‌ها، پردازش آن‌ها و ارائه بازخورد به محیط هستند. با گسترش برنامه‌های مبتنی بر اینترنت اشیاء، حجم درخواست‌های پردازشی به‌شدت افزایش یافته و چالش‌های جدیدی برای مدیریت و پردازش این داده‌ها ایجاد کرده است. محاسبات ابری قابلیت انجام محاسبات سنگین در زمان کم و هزینه مناسب را ایجاد می‌کند. برای مدیریت کارآمد منابع و کاهش هزینه‌ها، استفاده از الگوریتم‌های فراابتکاری مانند الگوریتم بهینه‌ساز تعادل، که بر پایه قوانین فیزیکی طراحی شده است، می‌تواند بسیار مؤثر باشد. در این مقاله، یک مسئله بهینه‌سازی چندهدفه شامل معیارهایی همچون زمان خاتمه آخرین وظیفه، تأخیر، تعادل بار، پایداری، مصرف انرژی و هزینه پردازش در محیط یکپارچه ابر ارائه شده است. با بهره‌گیری از الگوریتم بهینه‌ساز تعادل، این مسئله حل شده و نتایج با الگوریتم‌های فراابتکاری برجسته مقایسه شده است. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم پیشنهادی بهبود قابل‌توجهی در معیارهای مختلف داشته است. به‌طور خاص، در تابع زمان خاتمه آخرین وظیفه، الگوریتم پیشنهادی به ترتیب ۹٪، ۶.۲٪، ۹٪ و ۱۱٪ نسبت به الگوریتم‌های PSO، GSA، GWO و FA بهبود یافته است. همچنین، در کاهش تأخیر، این الگوریتم ۸٪، ۶.۷٪، ۹.۴٪ و ۹٪ عملکرد بهتری داشته است. از نظر هزینه پردازش، بهبود ۱.۳٪ تا ۲٪ و در مصرف انرژی کاهش ۲٪ تا ۳.۶٪ را نشان می‌دهد. در مجموع، برای توابع هدف به‌صورت همزمان، الگوریتم پیشنهادی حدود ۵.۶٪ تا ۱۰٪ عملکرد بهتری نسبت به رقبا داشته است. این نتایج برتری الگوریتم پیشنهادی در بهینه‌سازی همزمان معیارهای مختلف و افزایش کارایی سیستم‌های رایانش ابری را نشان می‌دهد.

واژه‌های کلیدی: الگوریتم‌های فراابتکاری، اینترنت اشیا، بهینه‌ساز تعادل، زمان‌بندی وظایف، زمان خاتمه آخرین وظیفه، کیفیت خدمات، محاسبات ابر

۱- مقدمه

با گسترش اینترنت اشیاء[۱] به‌عنوان یک فناوری تحول‌آفرین، حجم داده‌ها و تعداد درخواست‌های پردازشی به شکل چشمگیری افزایش یافته است. این تغییرات چالش‌های جدیدی را برای مدیریت و پردازش این حجم عظیم از داده‌ها به وجود آورده‌اند. در این میان، رایانش ابری به‌عنوان یک فناوری کلیدی در محاسبات توزیع‌شده و پردازش موازی، نقشی حیاتی در افزایش کارایی و بهبود عملکرد سیستم‌های اینترنت اشیاء ایفا می‌کند[۱].

رایانش ابری به دلیل ویژگی‌هایی مانند خدمات متنوع و انعطاف‌پذیری، به یکی از فناوری‌های محبوب برای مدیریت منابع

تبدیل شده است[۲]. این فناوری خدماتی همچون نرم‌افزار، فضای ذخیره‌سازی، خدمات شبکه و زیرساخت را به مشتریان ارائه می‌دهد. استفاده بهینه از این منابع، به‌ویژه در محیط‌های پویا مانند اینترنت اشیاء، نیازمند الگوریتم‌های زمان‌بندی وظایف کارآمد است. این الگوریتم‌ها تلاش می‌کنند تا علاوه بر کاهش هزینه‌ها، کیفیت خدمات[۲] را بهبود بخشند[۳].

رایانش ابری به کاربران اینترنت اشیاء امکان می‌دهد تا به مجموعه‌ای از منابع محاسباتی مشترک دسترسی پیدا کنند. این ویژگی‌ها، شامل انعطاف‌پذیری در تخصیص منابع و قابلیت گسترش سریع، رایانش ابری را به گزینه‌ای مناسب برای پردازش حجم بالای داده‌های اینترنت اشیاء تبدیل کرده است[۴]. علاوه بر این، هزینه‌های پایین نگهداری و مقیاس‌پذیری بالا از جمله مزایای کلیدی رایانش ابری هستند[۵].

ویژگی‌های متمایز محاسبات ابری، مانند خدمات قابل اندازه‌گیری، دسترسی گسترده به شبکه، سازگاری سریع و ادغام منابع، توجه تعداد زیادی از کاربران را به خود جلب کرده است. چهار مدل استقرار ابر متمایز وجود دارد: (۱) ابر عمومی، که خدمات زیرساختی را به عموم مردم ارائه می‌کند (مانند مراکز داده، فضای ذخیره‌سازی، شبکه‌ها و سکوهای توسعه). (۲) ابر خصوصی، که منابع ابری را برای استفاده انحصاری یک سازمان ارائه می‌دهد. (۳) ابر جامعه، که منابع ابری را بین چندین سازمان با نگرانی‌های مشابه به اشتراک می‌گذارد. (۴) ابر ترکیبی، که ترکیبی از سایر مدل‌های ابری است[۶, ۷].

با توجه به ماهیت پیچیده و پویای محیط ابری، تکنیک‌های زمان‌بندی وظایف برای به حداکثر رساندن مزایای ابر حیاتی هستند. رایانش ابری برای مدیریت منابع با اندازه‌های مختلف به روش‌های زمان‌بندی کار مؤثر نیاز دارد[۷].

از نظر واکنش‌پذیری[۳]، مقیاس‌پذیری و سازگاری، محیط ابری روشی برتر برای تکمیل فعالیت‌ها ارائه می‌دهد[۸]. زمان‌بندی به عنوان سیستمی توصیف می‌شود که تعیین می‌کند کدام وظایف در یک محیط توزیع‌شده باید به کدام رایانه اختصاص داده شود. الگوریتم‌های زمان‌بندی را می‌توان به دو دسته تقسیم کرد: (۱) الگوریتم‌های زمان‌بندی ایستا و (۲) الگوریتم‌های زمان‌بندی پویا. الگوریتم‌های زمان‌بندی ایستا قبل از اجرا به اطلاعات مرتبط با کار (مانند طول کار، تعداد کارها و مهلت‌ها) و منابع (مانند ظرفیت پردازش ماشین، قدرت پردازش و حافظه) نیاز دارند [۹].

الگوریتم‌های ایستا زمانی که حجم کار بسیار کم است و رفتار سیستم دائماً تغییر نمی‌کند گزینه‌های خوبی هستند، اما برای محاسبات ابری به‌خصوص زمانی که بار (حجم وظایف) دائماً در محیط در نوسان است، گزینه‌های خوبی نیستند. در حالی که روش‌های ایستا ساده هستند، پارامترهای کیفیت خدمات را بهبود نمی‌بخشند یا عملکرد کافی را در دنیای واقعی ارائه نمی‌دهند. به این دلیل به یک الگوریتم زمان‌بندی کار پویا برای محیط ابری نیاز است[۱۰].

در روش زمان‌بندی پویا، الگوریتم علاوه بر استفاده از اطلاعات قبلی، باید قابلیت انعطاف‌پذیری در حین اجرا را داشته باشد. این انعطاف‌پذیری با نظارت مداوم بر ماشین‌های موجود در محیط و تطبیق تصمیمات بر اساس شرایط فعلی سیستم امکان‌پذیر می‌شود. این الگوریتم‌ها در محیط ابری کارآمدتر و دقیق‌تر هستند، زیرا هنگامی که بار روی یکی از ماشین‌ها تغییر می‌کند، وظیفه را از یک ماشین پربار به ماشین کم‌بار منتقل می‌کنند. به عنوان مثال، شرایط الگوریتم اغلب با تغییر بار روی ماشین تغییر می‌کند (افزایش یا کاهش می‌یابد)[۱۰].

زمان‌بندی وظایف در محاسبات ابری یکی از جنبه‌های کلیدی در بهینه‌سازی منابع و عملکرد سیستم‌های ابری است. اهداف اصلی زمان‌بندی وظایف شامل بهبود تعادل بار، افزایش توان عملیاتی، بهبود کیفیت خدمات، کاهش زمان اجرا و مدیریت هزینه‌ها است[۱۱, ۱۲]. عوامل متعددی بر زمان‌بندی وظایف در محیط ابری تأثیر می‌گذارند، از جمله زمان پاسخ، مصرف انرژی، امنیت، هزینه، اولویت وظایف و قابلیت اطمینان. با این حال، بسیاری از روش‌های موجود تنها بر روی یک یا دو پارامتر تمرکز دارند[۱۳]. در محاسبات ابری، وظایف به‌طور پویا بین منابع مختلف (سرورها، ماشین‌ها و منابع محاسباتی) توزیع می‌شوند و مدیریت بهینه این توزیع می‌تواند به کاهش زمان پاسخ، افزایش بهره‌وری و کاهش مصرف انرژی کمک کند. علاوه بر این، زمان‌بندی مؤثر به کاهش بار روی منابع، جلوگیری از ازدحام و بهبود تجربه کاربر کمک می‌کند. این مسئله به‌ویژه در محیط‌های پردازشی با حجم داده بالا و تعداد زیاد کاربران و وظایف، اهمیت ویژه‌ای دارد. همچنین، به دلیل مصرف بالای انرژی در محیط‌های ابری، نگرانی‌های زیست‌محیطی و اقتصادی نیز باید مورد توجه قرار گیرند[۱۴].

در این مقاله، ابتدا یک مدل چندهدفه برای مسئله زمان‌بندی وظیفه‌ها در محیط‌های رایانشی ابر ارائه می‌شود که در آن زمان خاتمه آخرین وظیفه[۴]، تاخیر، تعادل بار، پایداری، کارایی، مصرف انرژی و هزینه پولی به طور توأم بهینه می‌شوند. سپس با الگوریتم‌های فراابتکاری سعی در زمانبندی وظایف کرده و نتایج مقایسه شده است.

یکی از این الگوریتم‌ها، الگوریتم بهینه‌ساز تعادل[۵] است که بنا بر ادعای نویسندگان این الگوریتم دارای قدرت کاوش و بهره‌وری زیادی است که کمک می‌کند جواب‌های مسئله در بهینه محلی گرفتار نشوند[۱۵].

برای حل این مسئله، الگوریتم فراابتکاری بهینه‌ساز تعادل تطبیق داده شده و با بهره‌گیری از آن، مدل‌سازی صورت گرفته است. این الگوریتم سپس با الگوریتم‌های برجسته در زمینه زمان‌بندی وظایف در محاسبات ابری مقایسه شده است.

در ادامه، ساماندهی مقاله به این صورت است که در بخش ۲ به مرور کارهای گذشته پرداخته می‌شود. الگوریتم بهینه‌ساز تعادل در بخش ۳ توصیف می‌شود. مدلسازی سیستم شامل معماری سیستم و فرموله‌سازی مسئله در بخش ۴ ارائه می‌گردد. بخش ۵ به ارزیابی کارایی الگوریتم پیشنهادی اختصاص داده شده است. در نهایت، در بخش ۶ مقاله با نتیجه‌گیری و بیان چند جهت‌دهی تحقیقاتی برای آینده به پایان می‌رسد.

۲- پژوهش‌های پیشین

در سال‌های اخیر مطالعات زیادی در مورد زمان‌بندی کار در رایانش ابری انجام شده است که در این بخش به چند مورد از آنها پرداخته شده است.

لیو[۱۶] در پژوهش خود از الگوریتم بهینه‌ساز کلونی مورچه‌ها[۶] برای حل زمان‌بندی کار در مسائل رایانش ابری استفاده کرد. الگوریتم پیشنهادی از به‌روزرسانی‌های فرمون برای افزایش سرعت همگرایی الگوریتم و همچنین جلوگیری از به دام افتادن در بهینه محلی استفاده می‌کند. اهداف در نظر گرفته شده در مقاله فوق شامل کاهش طول زمان‌بندی کار و تعادل بار است. نتایج شبیه‌سازی الگوریتم پیشنهادی نشان داده است که الگوریتم فوق توانسته است معیار زمان خاتمه آخرین کار را کاهش و تخصیص منابع را بهبود بخشد. قابل ذکر است که در مقاله فوق معیارهای پایداری و کارایی در نظر گرفته نشده است.

در پژوهش گئو [۱۷]، الگوریتم دفاع شخصی فازی برای زمان‌بندی وظایف چندهدفه در رایانش ابری ارائه شده که اهداف آن کاهش فاصله زمانی، تعادل بار و هزینه است. نتایج شبیه‌سازی نشان داده است که این الگوریتم زمان خاتمه وظایف را کاهش داده، استفاده از ماشین‌های مجازی را بهبود بخشیده و مهلت زمانی را ۵ درصد نسبت به سایر الگوریتم‌ها ارتقا داده است، اما معیارهای پایداری و کارایی نادیده گرفته شده است.

ایمن و همکاران [۱۸] الگوریتم ژنتیک مرتب‌سازی نامغلوب-نسخه سوم[۷] را برای زمان‌بندی کار در رایانش ابری ارائه کرده‌اند که با اهداف زمان اجرا، مصرف برق و هزینه عمل می‌کند. مقایسه با الگوریتم ژنتیک مرتب‌سازی نامغلوب نسخه دوم[۸] نشان داده است که نسخه سوم پاسخ‌ها را اندکی بهبود داده است، اما معیارهای پایداری و کارایی در این پژوهش لحاظ نشده است.

یک نسخه پیشرفته از الگوریتم بهینه‌ساز چند نظمی[۹] توسط شکری و همکاران [۱۹] برای حل مشکل زمان‌بندی کار در رایانش ابری پیشنهاد شده است. الگوریتم پیشنهادی با دو الگوریتم بهینه‌ساز چند نظمی و الگوریتم بهینه‌سازی ازدحام ذرات[۱۰] مقایسه شد و نتایج نشان داده است که الگوریتم پیشنهادی توانسته است فاصله ساخت را نسبت به دو الگوریتم دیگر کاهش دهد و همچنین درصد استفاده از منابع را افزایش دهد.

حسین و همکاران [۲۰] نویسندگان الگوریتم زمان‌بندی کار با انرژی و عملکرد کارآمد[۱۱] را برای زمان‌بندی کار در رایانش ابری ارائه کرده‌اند که با هدف کاهش مصرف انرژی و افزایش کارایی سیستم طراحی شده است. این الگوریتم در دو مرحله عمل می‌کند: مرحله اول کاهش زمان اجرا و مرحله دوم کاهش زمان خاتمه و مصرف انرژی. نتایج نشان داده است که الگوریتم مصرف انرژی را ۵ تا ۲۰ درصد کاهش داده است و زمان خاتمه وظایف را بهبود داده است، اما معیارهای پایداری و تعادل بار در این پژوهش لحاظ نشده‌اند.

وی[۲۱] الگوریتم بهبود یافته کلونی مورچه‌ها[۱۲] را برای زمان‌بندی کار در رایانش ابری پیشنهاد کرده است. این الگوریتم با اهداف زمان‌بندی، تعادل بار و کاهش هزینه تکمیل کار طراحی شده و از ضرایب پاداش و جریمه برای افزایش سرعت جستجو استفاده می‌کند. همچنین، ضریب وزن بار ماشین‌های مجازی[۱۳]، برای بهبود تعادل بار در به‌روزرسانی فرمون معرفی شده است. نتایج نشان می‌دهد که این الگوریتم مسئله را در کوتاه‌ترین زمان حل کرده و استفاده از ماشین‌های مجازی را بهبود بخشیده است.

سو و همکاران[۲۲] از الگوریتم کلونی مورچه‌ها برای تخصیص منابع و زمان‌بندی وظایف در رایانش ابری استفاده کرده‌اند. آنها ابتدا مشکلات را با محدودیت‌های ACO حل کرده و سپس نسخه بهبودیافته‌ای به نام Q-ACOA ارائه دادند. نتایج شبیه‌سازی نشان داد که الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها عملکرد بهتری داشته است.

آگراوال و سریواستاوا [۲۳] با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات و تکنیک یادگیری مبتنی بر مخالفت، به حل مسئله زمان‌بندی کار در رایانش ابری پرداخته‌اند. این الگوریتم زمان

ساخت و محاسبات را کاهش داده و عملکرد بهتری نسبت به الگوریتم‌های مقایسه‌شده داشته است، اما در برخی موارد به پاسخ قابل قبولی نرسیده و معیارهای پایداری و تعادل بار نیز لحاظ نشده‌اند.

آمر و همکاران[۲۴] با استفاده از الگوریتم یادگیری نخبگان بهینه‌سازی شاهین هریس[۱۴]، روشی برای زمان‌بندی وظایف چندهدفه در رایانش ابری ارائه کردند. این الگوریتم با در نظر گرفتن حداقل زمان تکمیل به‌جای راه‌حل‌های تصادفی اولیه، معیارهای طول برنامه، هزینه اجرا و استفاده از منابع را بهبود داده است. نتایج شبیه‌سازی نشان داد که این الگوریتم نسبت به الگوریتم بهینه‌ساز شاهین هریس[۱۵] و سایر الگوریتم‌های مقایسه‌شده عملکرد بهتری دارد.

مطالعات نشان می‌دهند که تکنیک‌های بهینه‌سازی مبتنی بر الگوریتم‌های فراابتکاری، مانند الگوریتم ژنتیک، با محدودیت‌هایی مانند زمان محاسبات طولانی، نیاز به تنظیم تعداد زیادی پارامتر و در نظر گرفتن محدودیت‌های متعدد مواجه هستند. در حالی که PSO به دلیل سرعت بالا و پارامترهای کمتر مزایایی دارد، تحقیقات نشان داده‌اند که ACO در ارائه نتایج قابل‌اعتمادتر موفق‌تر عمل می‌کند.

در این مقاله، الگوریتم بهینه‌ساز تعادل به‌عنوان یکی از الگوریتم‌های فراابتکاری برای زمان‌بندی وظایف در محیط رایانش ابری پیاده‌سازی و با استفاده از توابع ارزیابی مختلف بررسی شده است. این الگوریتم با هدف بهینه‌سازی تخصیص منابع و افزایش بهره‌وری طراحی شده و عملکرد آن با دیگر الگوریتم‌های استاندارد مقایسه شده است. نتایج نشان می‌دهند که الگوریتم بهینه‌ساز تعادل به‌طور مؤثری زمان‌بندی وظایف را بهبود داده و کارایی کلی سیستم را افزایش می‌دهد. جزئیات پیاده‌سازی، تحلیل نتایج و مقایسه عملکرد این الگوریتم در ادامه بررسی شده است.

۳- الگوریتم بهینه‌ساز تعادل

در سال ۲۰۱۹ فرامرزی و همکاران[۱۵] الگوریتم بهینه‌ساز تعادل را با الهام از قوانین تعادل در فیزیک معرفی کردند. این الگوریتم نسبت به روش‌هایی مانند PSO و الگوریتم ژنتیک، توانایی بهتری در حفظ تنوع و جلوگیری از گرفتار شدن در بهینه‌های محلی دارد. الگوریتم با استفاده از جستجوی چهار راه‌حل برتر سرعت هم‌گرایی را افزایش می‌دهد. این الگوریتم بر مبنای تعادل پویا و معادله موازنه جرم طراحی شده است که با یک معادله دیفرانسیل مرتبه اول توصیف می‌شود و برای حفظ غلظت اجزای غیرواکنشی در حجم کنترل به کار می‌رود و به صورت زیر بیان می‌شود:

$$V\frac{dc}{dt} = QC_{eq} - Qc + G \qquad (۱)$$

که V نشان‌دهنده حجم کنترل است، $V\frac{dc}{dt}$ میزان تغییر کیفیت را نشان می‌دهد، C نشان‌دهنده غلظت در حجم کنترل است، Q به عنوان نرخ جریان حجمی به داخل یا خارج از حجم کنترل در نظر گرفته می‌شود. همچنین C_{eq} غلظت داخل حجم کنترل را در غیاب تولید جرم (که حالت تعادل است) نشان می‌دهد و در نهایت G نشان دهنده نرخ تولید انبوه در داخل حجم کنترل است.

ابتدا، الگوریتم باید فرآیند مقداردهی اولیه را با جستجوی تصادفی به صورت زیر تکمیل کند:

$$C_i^{initial} = C_{min} + rand_i(C_{max} - C_{min}) \qquad (۲)$$

$$i = 1,2,\dots,n$$

که در آن C_{min} و C_{max} به ترتیب نشان‌دهنده حد پایین و بردار حد بالایی متغیر بهینه شده است.

در ابتدای فرآیند بهینه‌سازی، هیچ دانشی در مورد وضعیت تعادل وجود ندارد و تنها نامزدهای تعادلی تعیین می‌شوند تا یک الگوی جستجو برای ذرات ارائه کنند. برای بهبود قابلیت جستجوی جهانی، الگوریتم پنج ذره را به عنوان راه‌حل‌های کاندید تعریف می‌کند که چهارتای آن‌ها راه‌حل بهینه فعلی و دیگری حالت میانگین این چهار راه‌حل است. استخر به شرح زیر ساخته شده است:

$$\vec{C}_{eq.pool} = \{\ \vec{C}_{eq}(۱),\ \vec{C}_{eq}\ (2),\ \vec{C}_{eq}\ (3),\ \vec{C}_{eq}\ (4),\ \vec{C}_{eq}(ave)\} \qquad (۳)$$

با هدف کنترل نرخ همگرایی الگوریتم و ایجاد تعادل بین اکتشاف و بهره‌برداری، یک بردار نمایی $\vec{F}$ به صورت زیر تعریف می‌شود:

$$\vec{F} = e^{-\vec{\lambda}\,(t-t0)} \qquad (۴)$$

که در آن t زمان است و با تعداد تکرارها نسبت معکوس دارد. t طبق رابطه زیر تعریف می‌شود:

$$t = \left(1 - \frac{Iter}{Max_Iter}\right)^{\left(a_2\frac{Iter}{Max_Iter}\right)} \qquad (۵)$$

زمان t، به عنوان تابعی از تکرار $(Iter)$ تعریف می‌شود و بنابراین با تعداد تکرارها کاهش می‌یابد Max_Iter حداکثر تعداد تکرارها را نشان می‌دهد. همچنین t_0 توسط رابطه زیر محاسبه می‌شود:

$$\vec{t}_0 = \frac{1}{\vec{\lambda}}\ln(-a_1 sign(\vec{r}-0.5)[1-e^{-\vec{\lambda}t}]) + t \qquad (۶)$$

که در آن از a_1 برای ایجاد تعادل بین بهره‌برداری و اکتشاف استفاده می‌شود. وقتی a_1 بالاتر باشد، الگوریتم کاوش بیشتری دارد. $Sign(\vec{r}-0.5)$ برای تعیین جهت بهره‌برداری و اکتشاف استفاده می‌شود. $\vec{r}$ و $\vec{\lambda}$ مولفه‌هایی بین ۰ و ۱ هستند.

بردار نرخ تولید $\vec{G}$ به بهبود توانایی بهینه‌سازی محلی کمک می‌کند که به شرح زیر تعریف می‌شود:

$$\vec{G} = \vec{G}0\ e{-}k\ (t{-}t0) \qquad (۷)$$

جایی که $\vec{G}_0$ مقدار اولیه است و $\vec{k}$ نشان‌دهنده ثابت فروپاشی است. به منظور داشتن الگوی جستجوی کنترل‌شده‌تر و سیستماتیک‌تر و محدود کردن تعداد متغیرهای تصادفی، این تحقیق $\vec{k}=\vec{\lambda}$ را فرض می‌کند و از عبارت نمایی مشتق شده قبلی استفاده می‌کند. بنابراین، مجموعه نهایی معادلات نرخ تولید به شرح زیر است:

$$\vec{G} = \vec{G}_0^{-\vec{\lambda}(t-t_0)} = \vec{G}_0\vec{F} \qquad (۸)$$

مقدار اولیه به صورت بیان می شود:

$$\vec{G}_0 = \overrightarrow{GCP} * (\vec{C}_{eq} - \vec{\lambda}\vec{C}) \qquad (۹)$$

```
Initialize the particle's populations i=1,…,n
Assign equilibrium candidates' fitness a large number
Assign free parameters a1=2; a2=1; GP=0.5;
While Iter < Max_iter
      For i=1: number of particles (n)
      Calculate fitness of ith particle
          If fit(C"i)<fit(C"eq 1)
              Replace C"eq 1 with C"i and fit (C"eq 1) with fit(C"i)
          Elseif fit(C"i)>fit(C"eq 1) & fit(C"i)<fit(C"eq 2)
              Replace C"eq 2 with C"i and fit (C"eq 2) with fit(C"i)
          Elseif fit(C"i)>fit(C"eq 1) & fit(C"i)>fit(C"eq 2) & fit(C"i)<fit(C
"_eq)
              Replace C"eq 3 with C"i and fit (C"eq 3) with fit(C"i)
          Elseif fit(C"i)>fit(C"eq 1) & fit(C"i)>fit(C"eq 2) &
fit(C"i)>fit(C"eq 3)          & fit(C"i)<fit(C"eq 4)
              Replace C"eq 4 with C"i and fit (C"eq 4) with fit(C"i)
          End (If)
      End (For)
   C"ave=(C"eq 1+C"eq 2+C"eq 3+C"eq 4)/4
   Construct the equilibrium pool
   C"eq,pool={C"eq(1),C"eq(2),C"eq(3),C"eq(4),C"eq(ave) }
   Accomplish memory saving (if Iter > 1)
   Assign Eq(6)
      For i=1: number of particles (n)
          Randomly choose one candidate from the equilibrium pool (vector)
              Generate random vectors of λ", r"
              Construct F"=a1sign( r"-0.5)[e-λ "t-1]
              Construct Eq(10)
              Construct Eq(9)
              Construct G"= G0.F"
              Update concentrations Eq(11)
          End (For)
      Iter=Iter+1
End while
```

شبه‌کد (۱) الگوریتم بهینه‌ساز تعادل

$\overrightarrow{GCP}$ پارامتری است که نرخ تولید را کنترل می‌کند. اندازه آن عمدتاً به احتمال تولید GP بستگی دارد و تعریف آن بصورت زیر است:

$$GCP{=}0.5r1\ ,\quad r2{\geq}GP0\ ,\ r2{<}GP \qquad (۱۰)$$

که در آن r_1 و r_2 اعداد تصادفی از ۰ تا ۱ هستند. در نهایت، قوانین به روز رسانی EO را می توان به زیر صورت خلاصه کرد:

$$\vec{C} = \vec{C}_{eq} + (\vec{C}-\vec{C}_{eq}).\vec{F} + \frac{\vec{G}}{\vec{\lambda}V}(1-\vec{F}) \qquad (۱۱)$$

که در آن V به عنوان واحد در نظر گرفته می‌شود.

۴- معماری سیستم

معماری سیستم زمان‌بندی وظایف در محیط محاسبات ابری به صورت سلسله‌مراتبی طراحی شده است. این معماری شامل سه لایه اصلی است: لایه دستگاه‌های اینترنت اشیا، لایه مدیریت وظایف و لایه ماشین‌های مجازی. در لایه دستگاه‌های اینترنت اشیا، دستگاه‌ها به تولید داده و ارسال وظایف پردازشی به سرورهای ابری مشغول هستند که این وظایف شامل داده‌های حسگرها، اطلاعات محیطی و درخواست‌های کاربری است. در لایه مدیریت وظایف، یک واحد مرکزی مسئول دریافت وظایف از دستگاه‌های اینترنت اشیا، تحلیل و ارزیابی آن‌ها و تعیین اولویت‌ها و منابع مورد نیاز برای هر وظیفه است و سپس وظایف پس از تحلیل به ماشین‌های مجازی مناسب تخصیص داده می‌شوند. در لایه ماشین‌های مجازی، وظایف به ماشین‌های مجازی تخصیص داده شده پردازش می‌شوند و نتایج آن‌ها بازگشت داده می‌شوند. همچنین این لایه شامل سیستم مانیتورینگ برای نظارت بر وضعیت بارگیری و عملکرد ماشین‌های مجازی است.

۴-۱-مدل‌سازی سیستم

در مدل‌سازی سیستم زمان‌بندی وظایف در محاسبات ابری، یک محیط محاسباتی در نظر گرفته شده است که در آن وظایف متعددی از دستگاه‌های اینترنت اشیا (IoT) به ماشین‌های مجازی (VM) اختصاص داده می‌شوند. این وظایف شامل پردازش داده‌ها، تحلیل اطلاعات و ارسال نتایج می‌باشد. در این مدل، تعداد وظایف بین ۱۰۰ تا ۱۰۰۰ متغیر است و تعداد ماشین‌های مجازی بین ۱۰ تا ۴۰ تنظیم می‌شود. هر وظیفه و ماشین مجازی دارای ویژگی‌های خاصی هستند که در ادامه توضیح داده می‌شود.

برای محیط رایانشی ابر، ۴۰ ماشین مجازی ناهمگن در نظر گرفته

شده است. برای بررسی تأثیر تعداد وظایف، تعداد آن‌ها از ۱۰۰ تا ۱۰۰۰ (با گام‌های ۵۰تایی) تغییر می‌یابد. همچنین، اندازه هر وظیفه به طور تصادفی از بازه [۱۰۰۰-۴۰۰۰] انتخاب شده است. در این مقاله از مجموعه داده‌های تصادفی برای انجام شبیه‌سازی استفاده شده که کمک می‌کند تا بتوان سناریوهای متنوعی را پوشش داد. جدول (۱) مشخصات محیط رایانشی را نشان می‌دهد.

جدول ۱: مشخصات محیط محاسبات ابری

پارامتر	مقدار هر پارامتر
تعداد وظایف درخواستی	[۱۰۰-۱۰۰۰]
تعداد دستورالعمل های هر وظیفه	[۱۰۰۰-۴۰۰۰]
انتقال اطلاعات کار از یک کاربر به مراکز محاسبات ابری	[۱۰۰-۶۰۰]
تعداد ماشین های مجازی	[۱۰-۴۰]
میزان پهنای باند	[۵۰۰-۱۰۰۰]
نرخ سرعت پردازنده (بیت بر ثانیه)	[۵۰-۱۰۰]
هزینه پردازش	[۰.۷-۱]
مصرف توان	[۲۰۰-۴۰۰]

۴-۲- تابع هدف

در این بخش، زمان‌بندی وظایف با استفاده از الگوریتم بهینه‌ساز تعادل بر اساس پارامترهایی همچون زمان خاتمه آخرین وظیفه، تعادل بار، پایداری، کارایی، مصرف انرژی، تأخیر و هزینه پردازش ارزیابی می‌شود. ارزیابی توابع هدف انتخاب شده امری ضروری است، زیرا این ارزیابی‌ها اطمینان می‌دهند که سیستم بهینه‌سازی شده و عملکرد مطلوبی دارد. انجام منظم و دقیق این ارزیابی‌ها امکان شناسایی نقاط ضعف و اعمال بهبودهای لازم را فراهم می‌کند. علاوه بر این، چنین ارزیابی‌هایی به تطبیق سیستم با نیازهای متغیر کاربران و شرایط محیطی کمک کرده و تضمین می‌کنند که سیستم همواره بهینه و پایدار باقی بماند. این فرآیند نه تنها به شناسایی مشکلات و نواقص سیستم می‌انجامد، بلکه فرصتی برای تحلیل عملکرد آن تحت شرایط مختلف و بارهای کاری گوناگون فراهم می‌سازد. بدین ترتیب، سیستم می‌تواند به‌صورت مستمر به‌روزرسانی شده و نیازهای کاربران را به بهترین شکل ممکن برآورده کند.

در ادامه، روش‌های مختلف ارزیابی عملکرد سیستم و نتایج حاصل از این ارزیابی‌ها به‌تفصیل مورد بررسی قرار خواهند گرفت.

- **زمان خاتمه آخرین وظیفه**

زمان خاتمه آخرین وظیفه، که به عنوان makespan نیز شناخته می‌شود، معیاری است برابر با مدت زمانی که طول می‌کشد تا تمام کارها اجرا شده باشند. هدف این است که تا حد امکان این زمان کوتاه باشد. این معیار به ویژه در محیط‌های محاسباتی ابری که تعداد زیادی از وظایف به صورت همزمان پردازش می‌شوند، اهمیت دارد. کاهش makespan به معنی استفاده کارآمدتر از منابع و افزایش بهره‌وری کلی سیستم است. از این رو، این معیار به عنوان یکی از تابع‌های هدف اصلی انتخاب شده است.

از معادله زیر برای محاسبه زمان خاتمه آخرین وظیفه استفاده شده است[۲۵]:

$$Makespan = Max_{1<i\leq n}\{T_i\} \quad (۱۲)$$

که در آن n تعداد کل کارها و T_i زمان اتمام کار i ام است.

- **تعادل بار**

تعادل بار در محیط‌های محاسبات ابری، به توزیع یکنواخت وظایف بین منابع مختلف اشاره دارد. بدون تعادل بار مناسب، برخی از سرورها ممکن است بیش از حد بارگذاری شوند در حالی که سایر سرورها بیکار هستند. این مسئله نه تنها باعث کاهش کارایی سیستم می‌شود بلکه می‌تواند به خرابی منابع و افزایش هزینه‌ها نیز منجر شود. مقدار کمتر این معیار به این معنی است که الگوریتم توانسته است بار را بهتر توزیع کند. بنابراین، تعادل بار به عنوان یک معیار کلیدی انتخاب شده است تا اطمینان حاصل شود که منابع به طور بهینه و عادلانه مورد استفاده قرار می‌گیرند. مقدار تعادل بار با استفاده از معادله زیر به دست می‌آید [۲۵, ۲۶].

$$Load\ Balancing = \frac{Makespan}{Avg} \quad (۱۳)$$

متغیر Avg به عنوان نسبت کل پردازش انجام‌شده توسط هر پردازنده به تعداد کل پردازنده‌ها تعریف می‌شود. این مقدار میزان بار کاری پردازنده را به صورت میانگین توصیف می‌کند.

- **پایداری**

پایداری در شبیه‌سازی‌های رایانش ابری به معنای توانایی الگوریتم در ارائه عملکردی پایدار و قابل اعتماد تحت شرایط مختلف بار کاری است. برای ارزیابی پایداری، از انحراف معیار نتایج الگوریتم در چندین اجرای متوالی استفاده می‌شود. هرچقدر انحراف معیار کم باشد نشان‌دهنده پایداری بیشتر الگوریتم است.

- **کارایی**

کارایی به معنای میزان بهره‌برداری مؤثر از منابع موجود در هنگام اجرای وظایف است. این معیار نشان می‌دهد که الگوریتم چقدر از منابع پردازشی استفاده کرده و چقدر زمان‌بندی بهینه‌ای داشته

است. برای این کار ابتدا مقدار سرعت پردازش نسبی[۱۶] محاسبه می‌شود که برابر خواهد بود با:

$$RPS = \frac{(\sum_{i=1}^{n} T_i)}{Makespan} \quad (۱۴)$$

که در آن T_i برابر با زمان پردازش هر وظیفه است. سپس تابع کارایی به صورت زیر محاسبه می‌شود:

$$Efficiency = \frac{RPS}{N_p} \quad (۱۵)$$

که در آن N_p برابر با تعداد پردازنده‌های استفاده شده برای پردازش وظایف است.

- **تاخیر در پردازش**

تاخیر در زمان پردازش به مجموع مدت زمانی اشاره دارد که هر وظیفه از لحظه ورود به سیستم تا زمان شروع پردازش در صف انتظار قرار می‌گیرد. این تأخیر شامل مدت زمانی است که وظایف به دلیل محدودیت منابع یا اولویت‌بندی در صف باقی می‌مانند. به عبارت دیگر، تأخیر در زمان پردازش برابر است با جمع کل زمان‌های انتظار وظایف در صف قبل از آغاز پردازش. کاهش این تأخیر از طریق زمان‌بندی بهینه وظایف، نقش مهمی در بهبود کارایی سیستم و افزایش کیفیت خدمات ایفا می‌کند. برای محاسبه تأخیر در زمان پردازش از معادله‌ی زیر استفاده می‌شود:

$$D = \sum_{i=1}^{N} P_i \quad (۱۶)$$

که در آن D تاخیر در پردازش است و P_i زمان انتظار وظیفه i در صف، از لحظه ورود تا شروع پردازش است.

- **مصرف‌انرژی**

مصرف انرژی یکی دیگر از جنبه‌های مهم در محاسبات ابری است. با افزایش نگرانی‌ها در مورد تاثیرات زیست محیطی و هزینه‌های عملیاتی، کاهش مصرف انرژی به یک اولویت تبدیل شده است. بهینه‌سازی مصرف انرژی نه تنها باعث کاهش هزینه‌ها می‌شود بلکه به حفظ محیط زیست نیز کمک می‌کند. لذا، مصرف انرژی به عنوان یکی از تابع‌های هدف انتخاب شده است.

مصرف انرژی کل یک سیستم محاسباتی ابر، با مجموع مصرف انرژی تمامی ماشین‌های مجازی آن برابر است. برای هر ماشین مجازی مانند v_i، دو حالت فعال و غیرفعال در نظر می‌گیرند، که معمولا در حالت فعال، انرژی بیشتری مصرف می‌شود. میزان مصرف انرژی با توجه به هر وضعیت و مدت زمانی که در آن وضعیت بوده مشخص می‌شود که در این مقاله صرفا به بررسی حالت فعال می‌پردازد. به طور دقیق‌تر[۱]:

$$\varphi_i = E_i \times V_i^{active} \quad (۱۷)$$

که واحد آن وات ثانیه (یا ژول) می‌باشد.

بر اساس رابطه بالا می‌توان مصرف انرژی کل سیستم جهت اجرای همه وظیفه‌ها را به صورت زیر بیان کرد:

$$E_{total} = \sum_{i=1}^{M} \varphi_i \quad (۱۸)$$

- **هزینه پردازش**

هزینه‌های عملیاتی یک سیستم رایانشی می‌تواند شامل موارد متعددی همچون هزینه‌های پردازش، حافظه، پهنای‌باند شبکه، ذخیره‌سازی، خنک‌سازی و به‌روزرسانی نرم‌افزارها باشد. این هزینه‌ها نه تنها بر عملکرد سیستم تأثیر می‌گذارند، بلکه می‌توانند نقش مهمی در تصمیم‌گیری‌های مدیریتی و استراتژیک ایفا کنند. با توجه به اینکه تمرکز مقاله حاضر بر روی پردازش وظایف است، در مدل‌سازی مسئله تنها هزینه پردازشی در نظر گرفته شده است.

هزینه پردازشی به عنوان یکی از عوامل کلیدی در بهینه‌سازی و تخصیص منابع، می‌تواند تأثیر قابل توجهی بر کارایی و بهره‌وری سیستم داشته باشد. به همین دلیل، در این مقاله تمرکز اصلی بر تحلیل و بهینه‌سازی این نوع هزینه خواهد بود. با در نظر گرفتن هزینه پردازشی به عنوان معیار اصلی، می‌توان مدل‌های دقیق‌تری برای زمان‌بندی و تخصیص وظایف در محیط‌های رایانش ابری توسعه داد.

در ادامه، به بررسی جزئیات مدل‌سازی هزینه پردازشی، روش‌های بهینه‌سازی مرتبط و تأثیر آن بر عملکرد کلی سیستم پرداخته خواهد شد. همچنین، نتایج حاصل از این مدل‌سازی و تحلیل‌ها، به منظور ارائه راهکارهای بهینه برای مدیریت هزینه‌های پردازشی در سیستم‌های رایانش ابری مورد بحث قرار خواهد گرفت. وقتی که ماشین مجازی v_i وظیفه t_j را اجرا میکند، هزینه پردازشی آن برابر خواهد بود با:

$$\mathbb{C}_{i,j} = E_{i,j} \times v_i^{cost} \quad \forall j \in T, \forall i \in V \quad (۱۹)$$

بنابراین، هزینه پردازش کل سیستم را می‌توان با استفاده از رابطه زیر محاسبه کرد:

$$C = \sum_{i=1}^{M}\sum_{j=1}^{N} X_{i,j} \times \mathbb{C}_{i,j} \quad (۲۰)$$

که در آن N تعداد کل وظایف است و $X_{i,j}$ به این معنی است که وظیفه t_i به ماشین مجازی v_i نگاشت شده است.

مسئله بهینه‌سازی در این پژوهش به صورت یک مسئله بهینه‌سازی چندهدفه تعریف شده است که هدف آن کمینه‌سازی زمان خاتمه آخرین وظیفه، تأخیر، کارایی، پایداری، تعادل بار، مصرف انرژی و هزینه پردازشی کل سیستم است. این اهداف به‌دلیل تأثیر مستقیم بر کیفیت خدمات (QoS) و بهره‌وری منابع در محیط‌های رایانش ابری انتخاب شده‌اند. برای تبدیل این مسئله به یک مسئله بهینه‌سازی تک‌هدفه، از رویکرد مجموع وزنی استفاده شده است.

هدف مسئله بهینه‌سازی مطرح شده شامل موارد زیر است:

- کمینه‌سازی زمان خاتمه آخرین وظیفه: کاهش مدت زمان تکمیل تمامی وظایف
- کمینه‌سازی مصرف انرژی: کاهش مصرف انرژی سیستم به منظور صرفه‌جویی در هزینه‌ها و حفاظت از محیط زیست
- کمینه‌سازی هزینه پردازشی کل سیستم: کاهش هزینه‌های مرتبط با استفاده از منابع محاسباتی و ذخیره‌سازی
- کمینه‌سازی تأخیر: کاهش زمان انتظار وظایف در صف پردازش و تأخیر در ارائه خدمات.

علاوه بر این، معیارهای دیگری مانند تعادل بار، پایداری و کارایی نیز در این پژوهش مورد ارزیابی قرار گرفته‌اند. با این حال، این معیارها به‌صورت غیرمستقیم از ترکیب یا وابستگی به توابع هدف اصلی (زمان خاتمه آخرین وظیفه، مصرف انرژی، تأخیر و هزینه پردازش) مشتق می‌شوند. به همین دلیل، از قرار دادن این معیارها به‌عنوان توابع مستقل در مدل بهینه‌سازی خودداری شده است. این تصمیم علاوه بر کاهش پیچیدگی محاسباتی، تمرکز بر اهداف اصلی و افزایش دقت و شفافیت مدل بهینه‌سازی را تضمین کرده است.

با استفاده از رویکرد مجموع وزنی، اهداف مختلف با وزن‌های مشخص ترکیب می‌شوند تا یک تابع هدف کلی ایجاد شود. وزن‌های تخصیص‌یافته اهمیت نسبی هر یک از اهداف را مشخص می‌کنند و امکان ساده‌سازی مسئله چندهدفه به یک مسئله تک‌هدفه را فراهم می‌آورند.

پس می‌توان بهینه‌سازی مقید را بصورت زیر بیان کرد:

$$F = w_1 \times E + w_2 \times M + w_3 \times C + w_4 \times D \quad (۲۱)$$

که در اینجا w_1، w_2، w_3 و w_4 وزن‌های هر کدام از تابع‌های هدف ما هستند و همچنین E، M، C و D به ترتیب نمایانگر توابع مصرف انرژی، زمان خاتمه آخرین وظیفه، هزینه پردازش و تاخیر هستند.

مدل‌سازی سیستم با رعایت قیدهای زیر انجام شده است:

$$w_1 + w_2 + w_3 + w_4 = 1 \quad (۲۲)$$

$$\sum_{i=1}^{M} X_{i,j} = 1 \quad \forall j \in T \quad (۲۳)$$

$$0 \le w_k \le 1 \quad \forall k \in \{1,2,3,4\} \quad (۲۴)$$

این قیدها به ما کمک می‌کنند تا اطمینان حاصل کنیم که راه‌حل به دست آمده نه تنها بهینه است بلکه قابل پیاده‌سازی و عملی نیز می‌باشد. قید (۲۲) به ترتیب مشخص می‌کنند که ضریب هر هدف باید بین صفر و یک باشد و مجموع آنها باید برابر یک باشد. قید (۲۳) تضمین می‌کند که هر وظیفه به طور دقیق به یک پردازنده تخصیص داده شود. به عبارت دیگر، هیچ وظیفه‌ای نمی‌تواند به بیش از یک پردازنده اختصاص یابد و همچنین هیچ وظیفه‌ای بدون تخصیص باقی نمی‌ماند. این قید برای اطمینان از صحت و یکپارچگی زمان‌بندی وظایف در سیستم طراحی شده است. قید (۲۴) بیان می‌کند که وزن‌های w_k که نشان‌دهنده اهمیت نسبی هر یک از توابع هدف (مصرف انرژی، زمان خاتمه، هزینه پردازش و تأخیر) هستند، باید بین ۰ و ۱ قرار داشته باشند. این محدودیت تضمین می‌کند که مقادیر وزن‌ها معتبر بوده و در ترکیب خطی توابع هدف قابل استفاده باشند. علاوه بر این، این قید امکان تنظیم اهمیت هر تابع هدف بر اساس نیازهای خاص سیستم را فراهم می‌کند.

۵- ارزیابی کارایی

در این بخش، الگوریتم بهینه‌ساز تعادل ارزیابی شده است و با الگوریتم‌های متاهیورستیک از جمله الگوریتم جستجوی گرانشی، الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم گرگ خاکستری و الگوریتم کرم شب‌تاب مقایسه شده است.

آزمایش‌ها با استفاده از زبان برنامه‌نویسی MATLAB R2020a انجام شده است. تمامی اجراها بر روی یک کامپیوتر شخصی با مشخصات زیر انجام شده است: پردازنده از نوع Intel Core i7-

7700HQ با سرعت ۲.۸ GHz و چهار هسته، حافظه RAM با ظرفیت ۱۶.۰۰ گیگابایت و سیستم عامل Windows 10. به منظور افزایش قابلیت اطمینان نتایج، هر آزمایش ۱۰۰ بار تکرار شده و میانگین آنها گزارش شده است.

الگوریتم بهینه‌سازی ازدحام ذرات است که این الگوریتم از رفتار اجتماعی پرندگان الهام گرفته شده است. در الگوریتم بهینه‌سازی ازدحام ذرات، هر ذره به عنوان یک نقطه در فضای جستجو در نظر گرفته می‌شود که موقعیت و سرعت آن براساس تجربیات خودش و همسایگانش به‌روزرسانی می‌شود. هدف الگوریتم بهینه‌سازی ازدحام ذرات یافتن موقعیتی است که مقدار تابع هدف را بهینه کند. این الگوریتم به دلیل سادگی و کارایی بالا در حل مسائل بهینه‌سازی چندبعدی مورد توجه قرار گرفته است[۲۷].

الگوریتم جستجوی گرانشی[۱۷] از قانون جاذبه نیوتن الهام گرفته شده است. در الگوریتم جستجوی گرانشی، هر عامل به عنوان یک جرم در نظر گرفته می‌شود که تحت تأثیر نیروی جاذبه سایر عوامل قرار می‌گیرد. هرچه جرم یک عامل بیشتر باشد، نیروی جاذبه بیشتری بر روی سایر عوامل اعمال می‌کند. موقعیت هر عامل به‌روزرسانی می‌شود تا به سمت بهترین موقعیت حرکت کند. الگوریتم جستجوی گرانشی به دلیل توانایی بالای خود در جستجوی فضای حل و همگرایی به پاسخ‌های بهینه برای مسائل پیچیده مورد استفاده قرار می‌گیرد[۲۸].

الگوریتم بهینه‌سازی گرگ خاکستری[۱۸] از رفتار شکار گرگ‌های خاکستری الهام گرفته شده است. الگوریتم بهینه‌سازی گرگ خاکستری شامل چهار نوع گرگ به نام‌های آلفا، بتا، دلتا و امگا است که هر کدام نقش خاصی در فرآیند شکار دارند. الگوریتم به‌طور متناوب بین جستجوی محلی و جستجوی جهانی تغییر می‌کند تا بهترین راه‌حل ممکن را پیدا کند. الگوریتم بهینه‌سازی گرگ خاکستری به دلیل ساختار ساده و توانایی بالا در حل مسائل بهینه‌سازی پیچیده، مورد توجه بسیاری از محققان قرار گرفته است[۲۹].

الگوریتم دیگری که در این مقاله به آن اشاره شد، الگوریتم کرم شب‌تاب[۱۹] نام دارد. این الگوریتم از رفتار کرم‌های شب‌تاب و نحوه جذب آن‌ها به نور یکدیگر الهام گرفته شده است. در الگوریتم کرم شب‌تاب، هر کرم شب‌تاب به عنوان یک عامل در فضای جستجو در نظر گرفته می‌شود که به سمت کرم‌های شب‌تاب با روشنایی بیشتر حرکت می‌کند. روشنایی هر کرم شب‌تاب با مقدار تابع هدف مرتبط است. الگوریتم کرم شب‌تاب به دلیل سادگی و کارایی در حل مسائل بهینه‌سازی پیوسته و گسسته شناخته شده است[۳۰].

۵-۱- نتایج شبیه‌سازی

در این زیربخش، نتایج حاصل از پیاده‌سازی مورد بررسی و گزارش قرار خواهد گرفت. برای رسیدن به این هدف، ابتدا به مقایسه نتایج الگوریتم بهینه‌ساز تعادل با سایر الگوریتم‌های موجود پرداخته‌شده است.

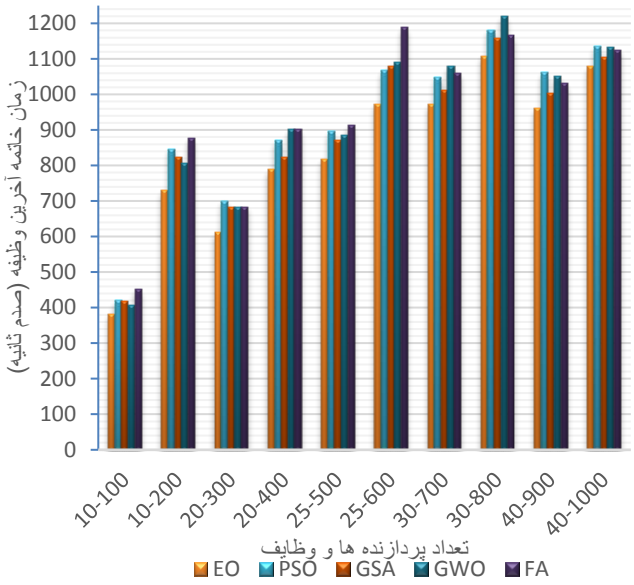
مقایسه عملکرد الگوریتم‌ها در شرایط مختلف و با توجه به معیارهای ارزیابی گوناگون، می‌تواند دیدگاه جامعی از نقاط قوت و ضعف هر الگوریتم ارائه دهد. در این بخش، تلاش شده است تا نتایج حاصل از آزمایشات به‌صورت دقیق و جامع تحلیل شوند تا بتوان تصویری واضح از کارایی و اثربخشی الگوریتم بهینه‌ساز تعادل در مقایسه با دیگر الگوریتم‌ها ارائه داد.

جدول (۲) و شکل (۱) نتایج مربوط به زمان خاتمه آخرین وظیفه برای پنج الگوریتم را (بر حسب صدم ثانیه) نشان می‌دهد. با افزایش تعداد وظیفه‌ها، بار بیشتری به سیستم تحمیل شده و زمان خاتمه آخرین وظیفه برای همه الگوریتم‌ها تقریباً به صورت خطی افزایش می‌یابد. این امر طبیعی است، زیرا افزایش وظایف نیازمند منابع بیشتر و زمان پردازش طولانی‌تر است.

الگوریتم بهینه‌ساز تعادل (EO) بهترین عملکرد را نسبت به سایر الگوریتم‌ها نشان داده است. این برتری به مدیریت بهینه منابع و توزیع مناسب وظایف نسبت داده می‌شود. در مقایسه با الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم جستجوی گرانشی، الگوریتم بهینه‌سازی گرگ خاکستری و الگوریتم کرم شب‌تاب، الگوریتم بهینه‌ساز تعادل به ترتیب ۹٪، ۶.۲٪، ۹٪ و ۱۱٪ عملکرد بهتری داشته است.

جدول ۲: مقایسه داده‌های زمان خاتمه آخرین وظیفه

Test Data		EO	PSO	GSA	GWO	FA
VMs	Tasks	Makespan	Makespan	Makespan	Makespan	Makespan
۱۰	۱۰۰	**۳۸۲**	۴۲۰	۴۱۸	۴۰۷	۴۵۲
۱۰	۲۰۰	**۷۳۰**	۸۴۷	۸۲۳	۸۰۵	۸۷۶
۲۰	۳۰۰	**۶۱۲**	۷۰۰	۶۸۴	۶۸۴	۶۸۳
۲۰	۴۰۰	**۷۸۹**	۸۷۰	۸۲۴	۹۰۲	۹۰۱
۲۵	۵۰۰	**۸۱۸**	۸۹۵	۸۷۰	۸۸۶	۹۱۲
۲۵	۶۰۰	**۹۷۴**	۱۰۶۷	۱۰۸۰	۱۰۹۰	۱۱۸۹
۳۰	۷۰۰	**۹۷۲**	۱۰۴۹	۱۰۱۱	۱۰۷۸	۱۰۵۹
۳۰	۸۰۰	**۱۱۰۹**	۱۱۸۱	۱۱۵۹	۱۲۲۰	۱۱۶۶
۴۰	۹۰۰	**۹۶۱**	۱۰۶۱	۱۰۰۳	۱۰۵۱	۱۰۳۰
۴۰	۱۰۰۰	**۱۰۷۹**	۱۱۳۵	۱۱۰۵	۱۱۳۲	۱۱۲۵



شکل ۱: مقایسه زمان خاتمه آخرین وظیفه بین الگوریتم‌ها

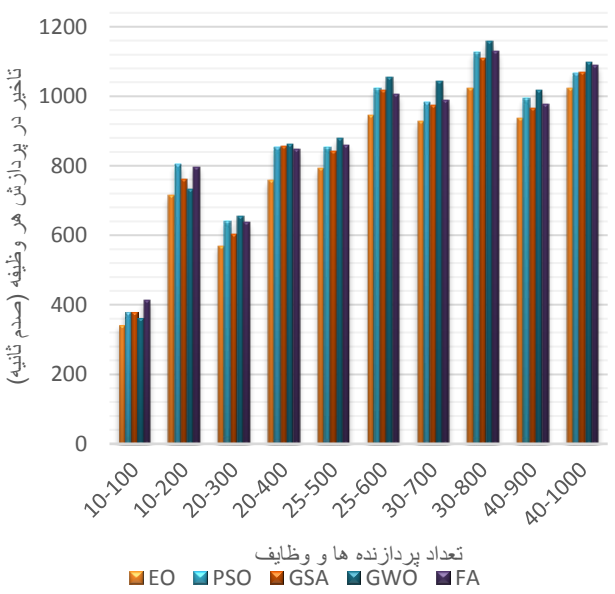
این الگوریتم با کاهش زمان بیکاری و استفاده بهینه از ظرفیت پردازشی سیستم، زمان خاتمه آخرین وظیفه را کاهش داده و در شرایط بار عادی و سنگین سازگاری بالایی از خود نشان داده است.

جدول (۳) و شکل (۲) نتایج مربوط به مجموع زمان تأخیر پنج الگوریتم را (بر حسب صدم ثانیه) نشان می‌دهد. همان‌طور که مشاهده می‌شود، با افزایش تعداد وظیفه‌ها، زمان تأخیر برای همه الگوریتم‌ها تقریباً به صورت خطی افزایش می‌یابد، زیرا افزایش بار کاری معمولاً به منابع بیشتر و زمان پردازش طولانی‌تر نیاز دارد. با این حال، تأثیر تعداد پردازنده‌های موجود نیز باید در نظر گرفته شود، چرا که می‌تواند رابطه خطی را تعدیل کند.

الگوریتم بهینه‌ساز تعادل بهترین عملکرد را در مقایسه با سایر الگوریتم‌ها نشان داده است. این الگوریتم توانسته است زمان تأخیر را نسبت به الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم جستجوی

جدول ۳: مقایسه الگوریتم‌ها از نظر مجموع تاخیر پردازش وظایف

Test Data		EO	PSO	GSA	GWO	FA
VMs	Tasks	Delay	Delay	Delay	Delay	Delay
۱۰	۱۰۰	**۳۴۲**	۳۷۹	۳۷۹	۳۶۲	۴۱۲
۱۰	۲۰۰	**۷۱۴**	۸۰۵	۷۶۰	۷۳۲	۷۹۷
۲۰	۳۰۰	**۵۶۹**	۶۴۰	۶۰۳	۶۵۴	۶۳۷
۲۰	۴۰۰	**۷۵۸**	۸۵۳	۸۵۷	۸۶۳	۸۴۸
۲۵	۵۰۰	**۷۹۳**	۸۵۴	۸۴۱	۸۷۸	۸۵۸
۲۵	۶۰۰	**۹۴۵**	۱۰۲۲	۱۰۱۷	۱۰۵۴	۱۰۰۵
۳۰	۷۰۰	**۹۲۷**	۹۸۴	۹۷۴	۱۰۴۳	۹۸۸
۳۰	۸۰۰	**۱۰۲۴**	۱۱۲۶	۱۱۱۰	۱۱۵۸	۱۱۲۸
۴۰	۹۰۰	**۹۳۶**	۹۹۳	۹۶۵	۱۰۱۸	۹۷۸
۴۰	۱۰۰۰	**۱۰۲۴**	۱۰۶۷	۱۰۶۹	۱۰۹۸	۱۰۸۹



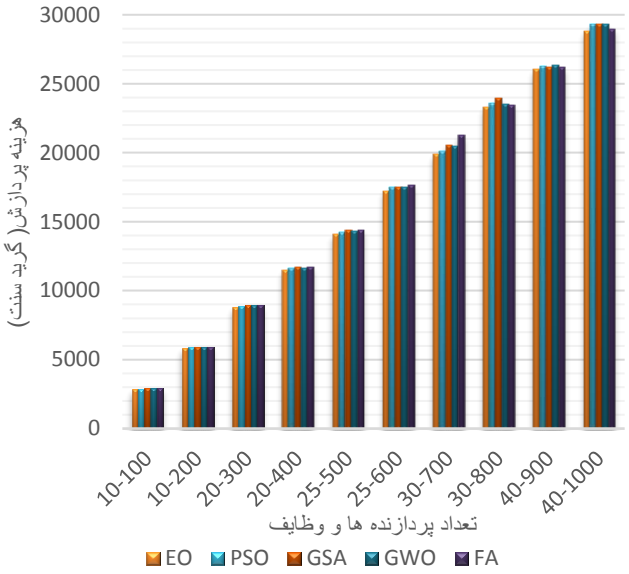
شکل ۲: نمودار مقایسه مجموع تأخیر در پردازش وظایف

گرانشی، الگوریتم بهینه‌سازی گرگ خاکستری و الگوریتم کرم شب‌تاب به ترتیب ۸٪، ۶.۷٪، ۹.۴٪ و ۹٪ کاهش دهد.

شکل (۳) و جدول (۴)، هزینه پردازش وظایف را در شرایط مختلف (بر حسب گرید سنت) نشان می‌دهند. همان‌طور که انتظار می‌رود، با افزایش تعداد وظایف، هزینه پردازش برای همه الگوریتم‌ها افزایش می‌یابد، زیرا بار کاری بیشتر نیازمند منابع بیشتری است. در این مقایسه، الگوریتم بهینه‌ساز تعادل نسبت به الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم جستجوی گرانشی، الگوریتم بهینه‌سازی گرگ خاکستری و الگوریتم کرم شب‌تاب به ترتیب ۱.۳٪، ۱.۹٪، ۱.۶٪ و ۲٪ بهبود داشته است. این برتری ناشی از بهینه‌سازی زمان‌بندی وظایف و استفاده کارآمدتر از منابع پردازشی است.

جدول ۴: مقایسه الگوریتم‌ها از نظر هزینه پردازش وظایف

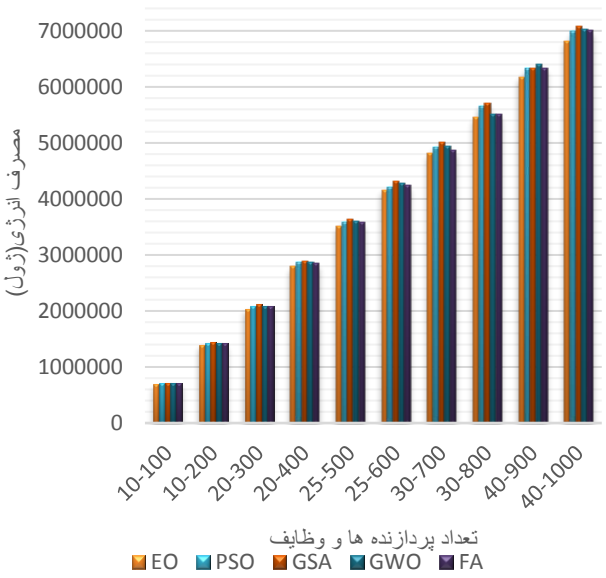
Test Data		EO	PSO	GSA	GWO	FA
VMs	Tasks	Processing Cost	Processing Cost	Processing Cost	Processing Cost	Processing Cost
۱۰	۱۰۰	۲۸۴۶	۲۸۸۰	۲۹۰۱	۲۸۹۲	۲۹۰۹
۱۰	۲۰۰	۵۷۹۵	۵۸۶۵	۵۹۰۷	۵۸۸۹	۵۹۲۲
۲۰	۳۰۰	۸۷۵۸	۸۸۶۵	۸۹۲۸	۸۹۰۱	۸۹۵۰
۲۰	۴۰۰	۱۱۴۴۶	۱۱۶۰۶	۱۱۶۸۹	۱۱۶۵۲	۱۱۷۱۸
۲۵	۵۰۰	۱۴۰۹۱	۱۴۲۶۳	۱۴۳۶۵	۱۴۳۲۰	۱۴۴۰۱
۲۵	۶۰۰	۱۷۲۴۵	۱۷۵۰۳	۱۷۴۸۰	۱۷۵۰۱	۱۷۶۲۳
۳۰	۷۰۰	۱۹۸۶۳	۲۰۰۷۶	۲۰۵۴۲	۲۰۴۵۹	۲۱۲۶۷
۳۰	۸۰۰	۲۳۲۸۹	۲۳۵۷۴	۲۳۹۱۰	۲۳۴۷۶	۲۳۴۵۹
۴۰	۹۰۰	۲۶۰۴۵	۲۶۲۳۴	۲۶۲۱۷	۲۶۳۳۷	۲۶۱۸۲
۴۰	۱۰۰۰	۲۸۸۲۱	۲۹۲۸۱	۲۹۲۸۷	۲۹۳۲۲	۲۸۹۶۴



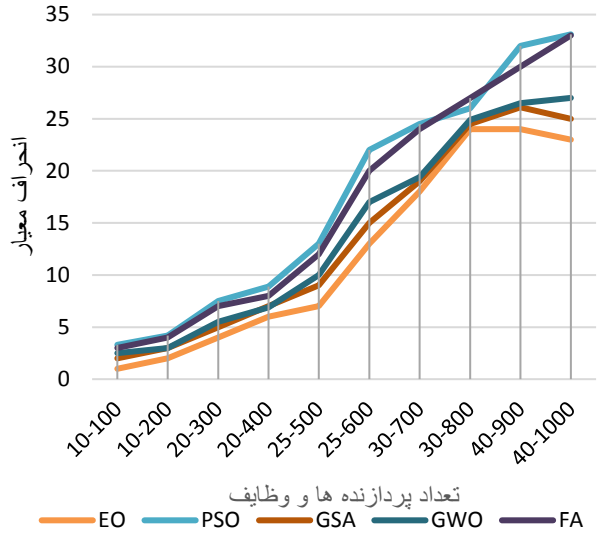
شکل ۳: مقایسه نتایج شبیه‌سازی تابع هزینه پردازش

در شکل (۴)، مصرف انرژی سیستم (بر حسب ژول) در پنج الگوریتم مقایسه شده است. الگوریتم بهینه‌ساز تعادل توانسته است مصرف انرژی را نسبت به الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم جستجوی گرانشی، الگوریتم بهینه‌سازی گرگ خاکستری و الگوریتم کرم شب‌تاب به ترتیب ۲.۴٪، ۳.۶٪، ۲.۶٪ و ۲٪ کاهش دهد. این کاهش به دلیل مدیریت بهینه منابع انرژی و کاهش تلفات پردازشی است.

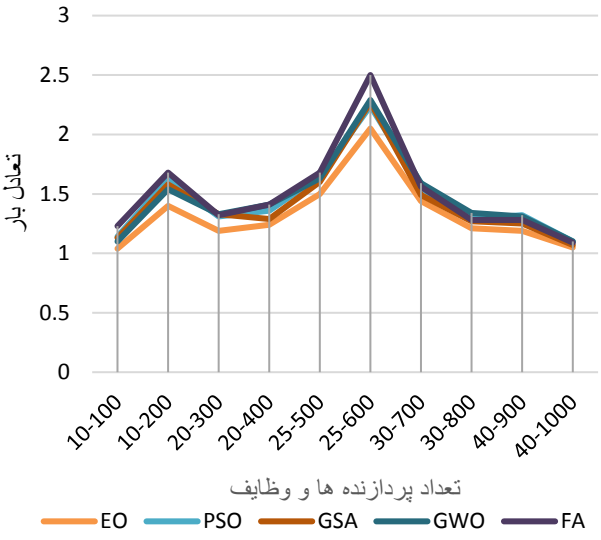
شکل‌های (۵)، (۶) و (۷) نتایج پایداری، تعادل بار و کارایی سیستم را نشان می‌دهند. با افزایش تعداد وظایف، عملکرد سیستم در هر سه معیار برای تمام الگوریتم‌ها کاهش می‌یابد.



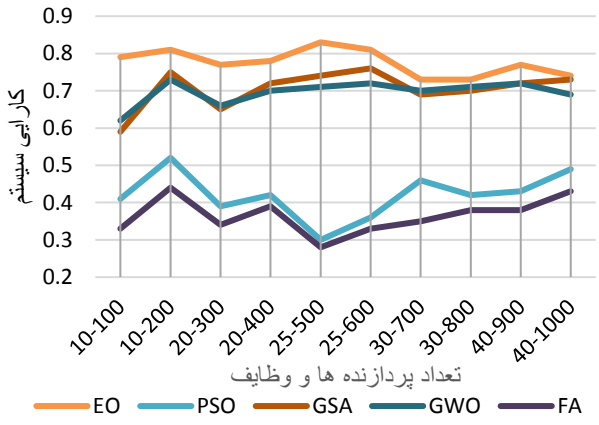
شکل ۴: مقایسه نتایج شبیه‌سازی تابع مصرف انرژی



شکل ۵: نتایج شبیه‌سازی پارامتر پایداری برای الگوریتم‌ها



شکل ۶: مقایسه نتایج شبیه‌سازی پارامتر تعادل بار الگوریتم‌ها



شکل۷: نمودار کارایی الگوریتم‌ها در سیستم مدل تعریف شده

در شکل (۵)، کارایی سیستم مقایسه شده است. نتایج حاکی از آن است که الگوریتم بهینه‌ساز تعادل با حفظ توازن مناسب بین کاوش و بهره‌وری، پایداری بیشتری را نسبت به سایر الگوریتم‌ها ارائه داده است. این پایداری به سیستم امکان می‌دهد تا در شرایط مختلف بار کاری عملکرد مطلوبی داشته باشد.

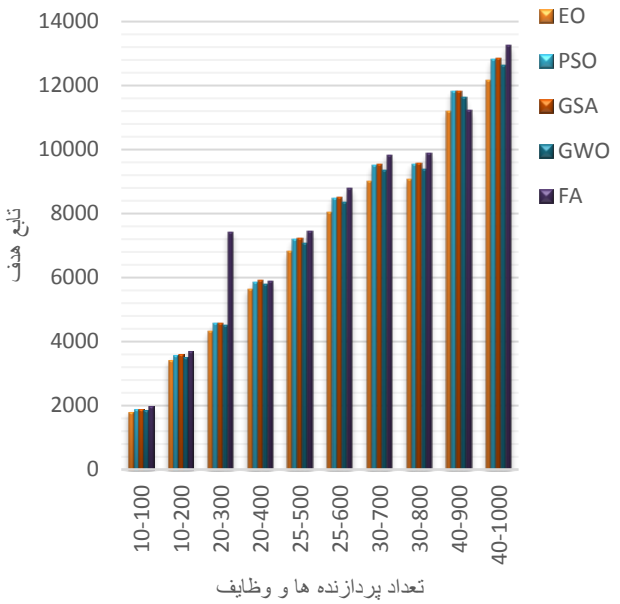
در شکل (۶)، تعادل بار سیستم بررسی شده است. الگوریتم بهینه‌ساز تعادل با توزیع متعادل‌تر وظایف میان منابع، نسبت به الگوریتم‌های دیگر تعادل بار بهتری را ارائه داده است. این ویژگی باعث جلوگیری از ایجاد نقاط گلوگاه در سیستم و بهبود عملکرد در شرایط بار کاری سنگین می‌شود.

شکل (۷) نیز پایداری سیستم را نشان می‌دهد. الگوریتم بهینه‌ساز تعادل بهترین عملکرد را نسبت به سایر الگوریتم‌ها نشان داده است. این برتری ناشی از توانایی بهینه‌ساز تعادل در زمان‌بندی بهینه وظایف و استفاده مؤثر از منابع است، که منجر به افزایش بهره‌وری کلی سیستم می‌شود. به طور کلی، الگوریتم بهینه‌ساز تعادل در هر سه معیار بررسی‌شده، شامل کارایی، تعادل بار و پایداری، عملکرد بهتری نسبت به الگوریتم‌های مقایسه‌شده داشته است. این نتایج نشان می‌دهد که استفاده از الگوریتم بهینه‌ساز تعادل می‌تواند به افزایش بهره‌وری سیستم، توزیع متعادل‌تر بار و بهبود پایداری کلی منجر شود.

شکل (۸) توازن میان زمان خاتمه آخرین وظیفه، تأخیر پردازش، تعادل بار، مصرف انرژی و هزینه پردازش الگوریتم‌ها را نشان می‌دهد. ضرایب به گونه‌ای تنظیم شده‌اند که تمامی توابع هدف با تأثیر یکسان در نظر گرفته شوند و قید شماره (۲۴) نیز رعایت شود.

همان‌طور که مشخص است، الگوریتم بهینه‌ساز تعادل عملکرد برتری نسبت به سایر الگوریتم‌ها دارد. این الگوریتم با کاهش زمان خاتمه آخرین وظیفه و تأخیر پردازش، بهبود قابل توجهی در کارایی سیستم ایجاد کرده است. همچنین، با بهینه‌سازی مصرف انرژی و کاهش هزینه پردازش، توانسته است در جنبه‌های مختلف بهینه‌سازی موفق عمل کند.

در مجموع، الگوریتم بهینه‌ساز تعادل نسبت به الگوریتم بهینه‌سازی ازدحام ذرات ۵.۷٪، نسبت به الگوریتم جستجوی گرانشی ۵.۶٪، نسبت به الگوریتم بهینه‌سازی گرگ خاکستری ۳.۶٪ و نسبت به الگوریتم کرم شب‌تاب ۱۰٪ عملکرد بهتری داشته است. این نتایج نشان‌دهنده توانایی بالای الگوریتم در مدیریت همزمان معیارهای مختلف و دستیابی به توازنی مؤثر میان آن‌ها است.

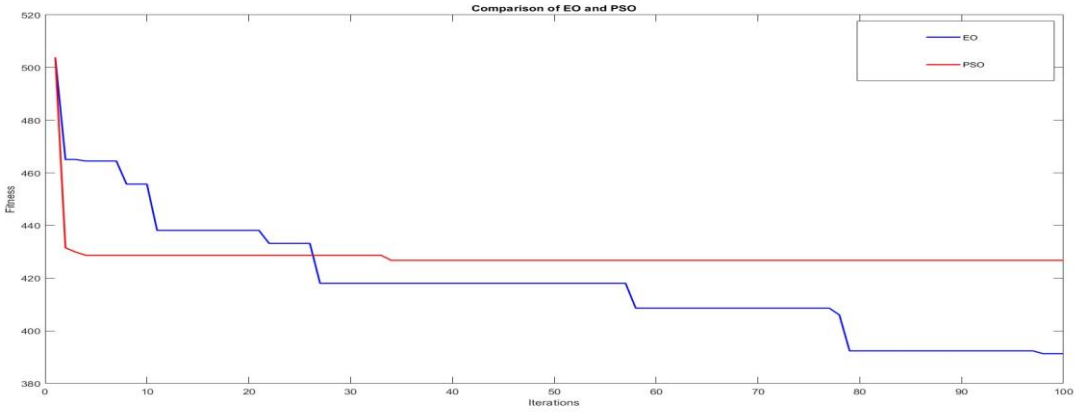


شکل ۸: تابع هدف‌ها به صورت همزمان

شکل (۹) عملکرد دو الگوریتم بهینه‌سازی ازدحام ذرات و بهینه‌ساز تعادل را طی ۱۰۰ تکرار نشان می‌دهد و برتری واضح الگوریتم بهینه‌ساز تعادل را به تصویر می‌کشد. در مراحل اولیه، هر دو الگوریتم بهبود مشابهی در مقادیر تابع هدف دارند، اما الگوریتم بهینه‌ساز تعادل، به دلیل ساختار پیشرفته و توازن مؤثر بین جستجوی محلی و جهانی، توانسته است به طور پیوسته نتایج بهتری ارائه دهد.

در مقابل، الگوریتم ازدحام ذرات در برخی تکرارها دچار ایستایی شده و به دلیل گرفتار شدن در بهینه‌های محلی، روند بهینه‌سازی آن متوقف می‌شود. الگوریتم بهینه‌ساز تعادل، با اجتناب از این مشکل و ادامه بهبود تا انتهای تکرارها، کارایی خود را در مسائل پیچیده بهینه‌سازی اثبات کرده است. این ویژگی‌ها شامل استمرار در بهبود، جلوگیری از گیر افتادن در بهینه‌های محلی و تولید نتایج بهتر در تمام مراحل، نشان‌دهنده برتری این الگوریتم نسبت به الگوریتم‌های دیگر است.

یکی از ویژگی‌های برجسته الگوریتم بهینه‌ساز تعادل نسبت به سایر الگوریتم‌های موجود، سرعت اجرای بهتر است. این الگوریتم با بهره‌گیری از مکانیسم‌های هوشمند تنظیم پارامترها و جستجوی کارآمد، توانسته است زمان اجرای خود را به طور قابل توجهی کاهش دهد. بر خلاف الگوریتم‌هایی مانند الگوریتم بهینه‌سازی ازدحام ذرات و الگوریتم جستجوی گرانش که ممکن است در جستجوی نقاط بهینه با چالش‌هایی مواجه شوند، الگوریتم بهینه‌ساز تعادل با به‌کارگیری روش‌های نوین، از گرفتار شدن در تله بهینه محلی جلوگیری می‌کند. این ویژگی، به الگوریتم اجازه



شکل ۹: مقایسه بهترین جواب دو الگوریتم EO و PSO

می‌دهد تا با دقت بیشتری به سمت نقاط بهینه جهانی حرکت کند و نتایج بهتری در حل مسائل پیچیده بهینه‌سازی ارائه دهد. به علاوه، ساختار انعطاف‌پذیر و توانایی بالای این الگوریتم در تطبیق با شرایط مختلف، موجب شده است که در مقایسه با الگوریتم‌های دیگر مانند الگوریتم بهینه‌سازی گرگ خاکستری و الگوریتم کرم شب‌تاب، عملکرد سریع‌تری داشته باشد و بهینه‌سازی را در زمان کوتاه‌تری به انجام برساند. این مزایا، الگوریتم بهینه‌ساز تعادل را به عنوان انتخابی مناسب برای مسائل بهینه‌سازی پیچیده و زمان‌بر مطرح کرده است.

در نتیجه، استفاده از الگوریتم بهینه‌ساز تعادل در سیستم‌های پیچیده و با بار کاری متغیر، به دلیل توازن بهینه بین معیارهای مختلف، توصیه می‌شود. این الگوریتم می‌تواند بهبود قابل توجهی در بهره‌وری و کارایی سیستم ایجاد کرده و هزینه‌های عملیاتی را کاهش دهد. نتایج به‌دست‌آمده تأیید می‌کنند که الگوریتم پیشنهادی می‌تواند به عنوان یک راه‌حل جامع و کارآمد برای بهینه‌سازی سیستم‌های رایانشی مورد استفاده قرار گیرد.

۵-۲- نتیجه‌گیری و کارهای آینده

در این مقاله، مسئله زمان‌بندی وظایف در محیط یکپارچه ابر مورد بررسی قرار گرفت. هدف اصلی، تخصیص بهینه وظایف به منابع به‌گونه‌ای است که معیارهای عملکردی مختلف مانند زمان خاتمه آخرین وظیفه، تأخیر پردازش، تعادل بار، کارایی، پایداری، هزینه پردازش و مصرف انرژی بهینه شوند. با توجه به اینکه زمان‌بندی وظایف یک مسئله NP-hard است، از الگوریتم‌های فراابتکاری برای حل آن استفاده شد.

در این پژوهش، یک مدل چندهدفه برای بهینه‌سازی هم‌زمان معیارهای ذکرشده ارائه و الگوریتم فراابتکاری بهینه‌ساز تعادل (EO) برای حل این مسئله تطبیق داده شد. این الگوریتم با سایر الگوریتم‌های برجسته مانند الگوریتم بهینه‌سازی ازدحام ذرات (PSO)، الگوریتم جستجوی گرانشی (GSA)، الگوریتم بهینه‌سازی گرگ خاکستری (GWO) و الگوریتم کرم شب‌تاب (FA) مقایسه شد. نتایج نشان داد که الگوریتم بهینه‌ساز تعادل در تمامی معیارها عملکرد برتری داشته است.

در مقایسه با الگوریتم‌های بهینه‌سازی ازدحام ذرات، جستجوی گرانشی، گرگ خاکستری و کرم شب‌تاب، الگوریتم بهینه‌ساز تعادل در معیارهای مختلف عملکرد بهتری داشته است. این بهبودها شامل کاهش زمان خاتمه وظایف تا ۱۱٪، کاهش تأخیر پردازش تا ۹.۴٪، کاهش هزینه پردازش تا ۲٪، کاهش مصرف انرژی تا ۳.۶٪ و بهبود بهینه‌سازی توابع هدف به صورت هم‌زمان تا ۱۰٪ بوده است. علاوه بر این، الگوریتم بهینه‌ساز تعادل توانسته است در شرایط افزایش تعداد وظایف، تعادل بار، پایداری و کارایی سیستم را حفظ کند.

با توجه به نتایج به‌دست‌آمده، الگوریتم بهینه‌ساز تعادل نه تنها در بهینه‌سازی معیارهای مختلف عملکردی موفق بوده، بلکه به دلیل توانایی در مدیریت مؤثر منابع و کاهش هزینه‌ها، به‌عنوان یک راه‌حل جامع و کارآمد برای بهینه‌سازی سیستم‌های رایانشی معرفی می‌شود. این الگوریتم همچنین انعطاف‌پذیری بالایی در پاسخ به نیازهای مختلف تأمین‌کنندگان سرویس‌های ابری و کاربران اینترنت اشیاء دارد.برای کارهای آینده، قصد داریم که الگوریتم بهینه‌ساز تعادل را بهبود دهیم و محیط مه را به صورت یک شبکه مدل کنیم تا پارامترهایی مانند تأخیر انتقال و محدودیت پهنای باند را نیز در مدل‌سازی مسئله مدنظر قرار دهیم. به منظور ارزیابی بیشتر کارآمدی الگوریتم بهینه‌ساز تعادل، ما علاقمند هستیم که کارایی الگوریتم ارائه شده را تحت مجموعه

داده‌های واقعی و با استفاده از چارچوب‌های شناخته شده محیط مه مانند iFogSim ارزیابی کنیم.

مراجع

[1] S. Azizi, "A Multi-objective Model for Task Scheduling Optimization in Fog-Cloud Computing Environments," *Journal of Soft Computing and Information Technology*, vol. 10, no. 4, pp. 43-52, 2022.

[2] N. Mansouri and M. M. Javidi, "A review of data replication based on meta-heuristics approach in cloud computing and data grid," *Soft computing*, vol. 24, no. 19, pp. 14503-14530, 2020.

[3] M. Bansal and S. K. Malik, "A multi-faceted optimization scheduling framework based on the particle swarm optimization algorithm in cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100429, 2020.

[4] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural computing and applications*, vol. 33, pp. 13075-13088, 2021.

[5] B. Saemi, M. Sadeghilalimi, A. A. R. Hosseinabadi, M. Mouhoub, and S. Sadaoui, "A new optimization approach for task scheduling problem using water cycle algorithm in mobile cloud computing," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021: IEEE, pp. 530-539.

[6] F. Jauro, H. Chiroma, A. Y. Gital, M. Almutairi, M. A. Shafi'i, and J. H. Abawajy, "Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend," *Applied Soft Computing*, vol. 96, p. 106582, 2020.

[7] N. Mansouri, R. Ghafari, and B. M. H. Zade, "Cloud computing simulators: A comprehensive review," *Simulation Modelling Practice and Theory*, vol. 104, p. 102144, 2020.

[8] A. A. Zubair, S. B. A. Razak, M. A. B. Ngadi, A. Ahmed, and S. H. H. Madni, "Convergence-based task scheduling techniques in cloud computing: A review," *Emerging Trends in Intelligent Computing and Informatics: Data Science, Intelligent Information Systems and Smart Computing 4*, pp. 227-234, 2020.

[9] M. Shojafar, M. Kardgar, A. A. R. Hosseinabadi, S. Shamshirband, and A. Abraham, "TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan," in *Hybrid Intelligent Systems: 15th International Conference HIS 2015 on Hybrid Intelligent Systems, Seoul, South Korea, November 16-18, 2015 15*, 2016: Springer, pp. 103-115.

[10] F. Xin and L. Zhang, "The review of task scheduling in cloud computing," in *Geo-informatics in Sustainable Ecosystem and Society: 6th International Conference, GSES 2018, Handan, China, September 25–26, 2018, Revised Selected Papers 6*, 2019: Springer, pp. 119-126.

[11] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35-44, 2022.

[12] N. Manikandan, P. Divya, and S. Janani, "BWFSO: hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing," *Materials Today: Proceedings*, vol. 62, pp. 4903-4908, 2022.

[13] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 361-371, 2020.

[14] S. Hammouti, B. Yagoubi, and S. A. Makhlouf, "Workflow security scheduling strategy in cloud computing," in *International Symposium on Modelling and Implementation of Complex Systems*, 2020: Springer, pp. 48-61.

[15] A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, "Equilibrium optimizer: A novel optimization algorithm," *Knowledge-based systems*, vol. 191, p. 105190, 2020.

[16] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," *Optik*, vol. 258, p. 168677, 2022.

[17] X. Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5603-5609, 2021.

[18] L. Imene, S. Sihem, K. Okba, and B. Mohamed, "A third generation genetic algorithm NSGAIII for task scheduling in cloud computing," *Journal of King Saud university-computer and information sciences*, vol. 34, no. 9, pp. 7515-7529, 2022.

[19] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Systems with Applications*, vol. 168, p. 114230, 2021.

[20] M. Hussain, L.-F. Wei, A. Lakhan, S. Wali, S. Ali, and A. Hussain, "Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100517, 2021.

[21] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-12, 2020.

[22] Y. Su, Z. Bai, and D. Xie, "The optimizing resource allocation and task scheduling based on cloud computing and Ant Colony Optimization Algorithm," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-9, 2021.

[23] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9855-9875, 2021.

[24] D. A. Amer, G. Attiya, I. Zeidan, and A. A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 2793-2818, 2022.

[25] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G.-B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *The Journal of Supercomputing*, vol. 76, no. 8, pp. 6302-6329, 2020.

[26] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 4, pp. 4313-4327, 2023/04/01 2023, doi: 10.1007/s12652-023-04541-9.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4: ieee, pp. 1942-1948.

[28] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232-2248, 2009.

[29] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46-61, 2014.

[30] X.-S. Yang and A. Slowik, "Firefly algorithm," in *Swarm intelligence algorithms*: CRC Press, 2020, pp. 163-174.

پاورقی‌ها:

[1] Internet of Things (IoT)

[2] Quality of Service (QoS)

[3] Reactivity

[4] Makespan
[5] Equilibrium Optimizer (EO)
[6] Ant Colony Optimization (ACO)
[7] NSGA-III
[8] NSGA-II
[9] Multi-Verse Optimizer (EMVO)
[10] Particle swarm optimization (PSO)
[11] Energy and Performance-Efficient Task Scheduling
[12] IACO
[13] Virtual Machine (VM)
[14] Elite Learning Harris Hawks Optimizer (ELHHO)
[15] Harris Hawks Optimizer (HHO)
[16] Relative Processing Speed (RPS)
[17] Gravitational Search Algorithm (GSA)
[18] Grey Wolf Optimizer (GWO)
[19] Firefly Algorithm (FA)