Journal of New Technologies in Distributed Systems and Algorithmic Computing
Islamic Azad University of Sabzevar

E-ISSN: 3115-705X
https://sanad.iau.ir/journal/ntds

Reaserch Article

Improving Load Balancing in Fog Computing Using the Polar Fox Optimization Algorithm (PFA)

Omid Eslami[1] | Mehdi Effatparvar*[2]

[1] Department of Computer, Islamic Azad University, Ardabil Branch, Ardabil, Iran,
Omid.eslami0@gmail.com

[2] Department of Computer, Islamic Azad University, Ardabil Branch, Ardabil, Iran,
Me.effatparvar@iau.ac.ir

Corresponding Author
*Mehdi Effatparvar, Assistant Professor, Department of Computer, Islamic Azad University, Ardabil Branch, Ardabil, Iran,

Me.effatparvar@iau.ac.ir

Main Subjects: Improving load balancing in cloud computing
Received: 3 October2025
Revised: 30 November 2025
Accepted: 19 December 2025

https://doi.org/10.82195/ntds.2025.1219928

Abstract

Fog computing acts as an intermediate layer between edge devices and cloud infrastructures to reduce latency and improve efficiency in time-sensitive Internet of Things (IoT) applications. However, heterogeneous fog nodes and dynamically changing workloads pose major challenges to effective load balancing, often leading to increased response time, higher energy consumption, and inefficient resource utilization. This paper introduces a novel metaheuristic approach, termed the Polar Fox Optimization Algorithm (PFA), for load balancing in fog computing environments. Inspired by the adaptive hunting behavior of polar foxes, the proposed algorithm integrates exploration and exploitation phases to balance global and local search. The PFA algorithm was implemented in Python and MATLAB and evaluated against benchmark methods, including PSO, ACO, GA, GWO, and HHO. Simulation results demonstrate that the proposed approach reduces average latency by up to 18.7%, lowers energy consumption by 14.5%, and improves the load balancing index by 22.3% compared to competing algorithms. Statistical validation using Wilcoxon and Friedman tests ($p < 0.05$) confirms the superiority of the proposed method. Overall, PFA provides a scalable, stable, and efficient solution for load balancing in dynamic and heterogeneous fog computing environments, particularly for smart cities, real-time healthcare systems, and large-scale IoT applications.

Keywords: nternet of Things, Metaheuristic Algorithm, Polar Fox Optimization Algorithm,Fog Computing, Load Balancin.

فناوری های نوین در سیستم های توزیع شده و محاسبات الگوریتمی

دانشگاه آزاد اسلامی واحد سبزوار

E-ISSN: 3115-705X
https://sanad.iau.ir/journal/ntds

پژوهشی

بهبود توازن بار در رایانش مه با استفاده از الگوریتم بهینه‌سازی روباه قطبی(PFA)

امید اسلامی[۱] | **مهدی عفت پرور***[۲]

[۱] گروه مهندسی کامپیوتر، واحد اردبیل، دانشگاه آزاد اسلامی، اردبیل، ایران

Omid.eslamio@gmail.com

[۲] گروه مهندسی کامپیوتر، واحد اردبیل، دانشگاه آزاد اسلامی، اردبیل، ایران

Me.effatparvar@iau.ac.ir

نویسنده مسئو ل

مهدی عفت پرور، استادیارکامپیوتر، گروه مهندسی کامپیوتر، واحد اردبیل، دانشگاه آزاد اسلامی، اردبیل، ایران

Me.effatparvar@iau.ac.ir

موضوع اصلی: بهبود توازن بار در رایانش مه
تاریخ دریافت: ۱۴۰۴/۷/۱۱
تاریخ بازنگری: ۱۴۰۴/۹/۹
تاریخ پذیرش: ۱۴۰۴/۹/۲۸

https://doi.org/10.82195/ntds.2025.1219928

چکیده:

رایانش مه به‌عنوان یک معماری میان افزاینده میان دستگاه‌های لبه و ابر برای کاهش تأخیر و افزایش بهره‌وری در کاربردهای حساس به زمان در اینترنت اشیاء (IoT) مطرح است. بااین‌حال، ناهمگن بودن گره‌ها و پویایی بارهای کاری، چالش جدی در حفظ توازن بار است؛ مدیریت ناکارآمد توازن بار می‌تواند منجر به افزایش زمان پاسخ، مصرف انرژی بالا و کاهش بهره‌وری منابع گردد. در این پژوهش، الگوریتم فرابتکاری جدید با عنوان الگوریتم بهینه‌سازی روباه قطبی (PFA) برای بهبود توازن بار در رایانش مه پیشنهاد می‌شود، این الگوریتم با الهام از رفتار شکار روباه قطبی در محیط‌های قطبی و توانایی تطبیق حرکتی آن با شرایط پویا، از دوفاز مکمل اکتشاف و بهره‌برداری بهره می‌گیرد تا میان جستجو سراسری و موضعی اطراف خود تعادل برقرار کنند. برای ارزیابی، الگوریتم پیشنهادی در محیط‌های شبیه‌سازی Pythonو MATLAB پیاده‌سازی و با الگوریتم مرجع بهینه‌سازی ازدحام ذرات ، کلونی مورچه ، GA، گرگ خاکستری و شاهین هریس مقایسه شده است. نتایج شبیه‌سازی‌ها نشان می‌دهد که مدل پیشنهادی ما موجب میانگین کاهش تأخیر تا ۱۸,۷٪ کاهش مصرف انرژی تا ۱۴,۵٪ و بهبود شاخص توازن بار تا ۲۲,۳٪نسبت به الگوریتم های مرجع شده است. افزون بر این، آزمون‌های آماریWilcoxon و Friedman با مقدار $p<0.05$ برتری معنادار الگوریتم پیشنهادی را تأیید میکند ، رفتار هم گرایی الگوریتم نیز از طریق نمودارهای کانکتور و سه بعدی تحلیل شده ونشان میدهد که الگوریتم بهینه‌سازی روباه قطبی از نظر پایداری و سرعت هم گرایی عملکرد بهتری نسبت به روش های مرجع دارد. در نتیجه الگوریتم پیشنهادی بهینه‌سازی روباه قطبی میتواند به عنوان رویکردی هوشمند مقیاس پذیر و کار آمد برای مدیریت توازن بار در محیط های پویا و نا همگن مه به ویژه در کاربرد های شهر هوشمند ،سامانه های پزشکی بلادرنگ و شبکه های گسترده اینترنت اشیا مورد استفاده قرار گیرد.

کلیدواژه‌ها: اینترنت اشیا، الگوریتم فرابتکاری، الگوریتم بهینه‌سازی روباه قطبی، رایانش مه، توازن بار

۱. مقدمه:

با گسترش روزافزون دستگاه‌های هوشمند در اینترنت اشیاء، نیاز به پردازش داده‌ها در نزدیک‌ترین نقطه به منبع تولید آن‌ها بیش‌ازپیش اهمیت یافته است .در این راستا، رایانش مه[1] به عنوان لایه‌ای میان ابر و لبه، با هدف کاهش تأخیر، بهبود زمان پاسخ و بهینه‌سازی مصرف انرژی مطرح شده است [۱،۲]. باوجود مزایای فراوان، یکی از چالش‌های اساسی در رایانش مه، توزیع نابرابر بار کاری میان‌گره‌های مه و ایجاد گلوگاه‌های پردازشی است[۳،۲] روش‌های کلاسیک مانند Round Robin به دلیل ماهیت ایستا و عدم انطباق با پویایی محیط‌های مه، بهره‌وری محدودی دارند [۴،۵]. در سال‌های اخیر، الگوریتم‌های فراابتکاری متعددی برای حل این مسئله پیشنهاد اما هنوز چالشی اساسی در یافتن تعادلی میان اکتشاف و بهره‌برداری در جستجوی فضای بهینه وجود دارد. در پاسخ به این چالش، این مقاله الگوریتم جدیدی با نام الگوریتم بهینه‌سازی روباه قطبی را معرفی می‌کند که با الهام از رفتار تطبیقی روباه قطبی در شرایط محیطی سخت، از ترکیب سازوکارهای جستجوی تصادفی و جهت‌دار برای بهبود توازن بار بهره می‌گیرد [۶،۷،۸،۱۳].

اهداف پژوهش به‌صورت زیر تعریف می‌شوند :

(1) طراحی مدلی فراابتکاری برای توزیع پویا و هوشمند بار در محیط مه.

(2)مقایسه عملکرد بهینه‌سازی روباه قطبی با الگوریتم های بهینه‌سازی ازدحام ذرات ، گرگ خاکستری ، شاهین هریس، الگوریتم کلونی مورچه و بهترین الگوریتم مرجع

(3)ارزیابی شاخص های تأخیر، مصرف انرژی و نرخ توازن بار.

(4)تحلیل آماری نتایج با آزمون های Wilcoxon و Friedman

سؤالات پژوهش:

۱. آیا الگوریتم بهینه‌سازی روباه قطبی می‌تواند نسبت به الگوریتم‌های فراابتکاری موجود، توازن بار بهتری در رایانش مه ایجاد کند؟
۲. میزان تأثیر الگوریتم پیشنهادی بر کاهش تأخیر، مصرف انرژی و افزایش نرخ توازن بار چقدر است؟
۳. آیا استفاده از مکانیزم دوفازی بهینه‌سازی روباه قطبی اکتشاف–بهره‌برداری باعث بهبود پایداری همگرایی الگوریتم در محیط‌های پویا می‌شود؟

نوآوری اصلی پژوهش در سه محور بیان می‌شود :

(1) معرفی ساختار دو فازی اکتشاف–بهره‌برداری در الگوریتم بهینه‌سازی روباه قطبی ،

(2)استفاده از تابع برازندگی چندهدفه برای توازن بار،

(3)اعمال کنترل تطبیقی در انتخاب گره‌های مه با استفاده از مفهوم یادگیری محلی.

۲. پیشینه تحقیق:

با گسترش چشم‌گیر اینترنت اشیاء و افزایش حجم داده‌های بلادرنگ، مدل‌های سنتی رایانش ابری به‌دلیل تمرکزگرایی و تأخیر بالا دیگر پاسخ‌گوی نیازهای کاربردهای حساس زمان واقعی نبودند. در این میان، رایانش مه به‌عنوان معماری مکمل رایانش ابری مطرح شد تا پردازش داده‌ها را به گره‌هایی نزدیک‌تر به منبع تولید داده انتقال دهد [۱،۱۷]. این تغییر پارادایم سبب شد تا مسئله‌ی توازن بار به یکی از مهم‌ترین چالش‌های طراحی سامانه‌های مه تبدیل شود، زیرا توزیع نامناسب وظایف میان گره‌ها منجر به افزایش تأخیر، مصرف انرژی زیاد و کاهش بهره‌وری منابع می‌گردد [۱۵،۱۹،۲۲].

در مراحل اولیه توسعه رایانش مه، پژوهشگران عمدتاً از روش‌های کلاسیک ایستا و متمرکز برای توازن بار استفاده کردند. الگوریتم‌هایی مانند Round Robin و Weighted Round Robin باوجود سادگی و سرعت بالا، به دلیل بی‌توجهی به ظرفیت و وضعیت لحظه‌ای گره‌ها، در محیط‌های پویا و ناهمگن عملکرد ضعیفی داشتند [۲۶،۲۸،۲۹]. در پاسخ به این محدودیت‌ها، رویکردهای پویا و توزیع‌شده مطرح شدند که هدفشان انطباق با تغییرات محیط و کاهش گلوگاه‌های پردازشی بود

[1] Fog Computing

[۶،۷،۱۵]. هرچند این روش‌ها مقیاس‌پذیری بهتری ارائه می‌دادند، ولی سربار ارتباطی و پیچیدگی مدیریتی بیشتری نیز به همراه داشتند.

از دهه گذشته، جهت‌گیری پژوهش‌ها به سمت بهره‌گیری از الگوریتم‌های فراابتکاری برای توازن بار در محیط مه تغییریافته است، چراکه این الگوریتم‌ها قادرند در فضای جستجوی بزرگ و غیرخطی، پاسخ‌های نزدیک به بهینه بیابند [۱۲،۴۸]. در میان آن‌ها، الگوریتم‌هایی نظیر بهینه‌سازی ازدحام ذرات، کلونی مورچه، گرگ خاکستری، زنبور عسل مصنوعی و نهنگ بیشترین توجه را جلب کرده‌اند. الگوریتم بهینه‌سازی ازدحام ذرات با تقلید از رفتار اجتماعی ذرات برای بهینه‌سازی پیوسته طراحی شده و در تخصیص وظایف مه به‌دلیل سرعت همگرایی بالا کاربرد یافته است، اما در محیط‌های پویا دچار همگرایی زودرس و کاهش دقت می‌شود [۹،۱۴]. کلونی مورچه نیز با الهام از رفتار جستجوی مورچه‌ها برای یافتن مسیر بهینه، در کاهش تأخیر مؤثر بوده است ولی به‌علت سربار محاسباتی زیاد در مقیاس بزرگ کارایی کمتری دارد [۱۰،۱۱]. در مقابل، گرگ خاکستری با مدل‌سازی رفتار گروهی گرگ‌ها تعادلی میان اکتشاف و بهره‌برداری برقرار می‌کند و نتایج مطلوبی در توازن بار ارائه داده است، هرچند پایداری آن در شرایط تغییرپذیر بار کاهش می‌یابد [۱۲،۱۳]. الگوریتم‌های زنبور عسل مصنوعی و نهنگ نیز به‌ترتیب با شبیه‌سازی رفتار زنبورهای عسل و نهنگ‌ها، بهبودهایی در تخصیص پویا ایجاد کرده‌اند، اما نیازمند تنظیم دقیق پارامترها و منابع محاسباتی بالا هستند [۱۰،۳۱،۳۲].

در سال‌های اخیر، پژوهشگران به سمت رویکردهای ترکیبی و هوشمند پیش رفته‌اند تا از مزایای چند الگوریتم به‌صورت هم‌زمان بهره بگیرند. برای نمونه، ترکیب گرگ خاکستری و الگوریتم ژنتیک توانسته است میان سرعت همگرایی و دقت راه‌حل تعادل برقرار کند [۵]. از سوی دیگر، تلفیق شبکه‌های نرم‌افزارمحور با الگوریتم‌های بهینه‌سازی، امکان کنترل پویا و مقیاس‌پذیری بالاتر را فراهم کرده است [۷،۴۰]. افزون بر آن، استفاده از یادگیری عمیق برای پیش‌بینی بار و تصمیم‌گیری تطبیقی نیز مورد توجه قرار گرفته، اما این روش‌ها به‌دلیل نیاز به توان پردازشی و داده‌ی آموزشی زیاد، در محیط‌های مه با محدودیت منابع کمتر عملی‌اند [۴۲]. با وجود این پیشرفت‌ها، مطالعات گذشته هنوز با چند شکاف کلیدی روبه‌رو هستند:

۱. ناتوانی در حفظ پایداری هنگام تغییرات شدید بار و ناهمگنی گره‌ها [۹،۱۰،۱۲]. ۲. گیر افتادن در بهینه‌های محلی و کاهش تنوع جمعیت در فرآیند جستجو [۹،۱۴]. ۳. هزینه محاسباتی بالا و پیچیدگی تنظیم پارامترها در الگوریتم‌های ترکیبی [۳۱،۴۳،۴۵]. ۴. توجه محدود به چندمعیاره‌بودن مسئله، یعنی بهینه‌سازی همزمان تأخیر، انرژی و بهره‌وری منابع [۳۸،۳۱]. در نتیجه، نیاز به الگوریتمی احساس می‌شود که ضمن حفظ سادگی و کارایی، بتواند در محیط‌های پویا تعادل مؤثر میان اکتشاف سراسری و بهره‌برداری محلی برقرار کند.

براین‌اساس، پژوهش حاضر الگوریتم جدیدی با عنوان بهینه‌سازی روباه قطبی را معرفی می‌کند که با الهام از رفتار شکار و سازگاری روباه قطبی در محیط‌های سرد و متغیر، تلاش دارد ضعف‌های الگوریتم‌های پیشین را جبران نماید. این الگوریتم از چند نوآوری کلیدی بهره می‌گیرد: تقسیم نقش جمعیت، فاز دوگانه اکتشاف–بهره‌برداری، و سازوکار ضد همگرایی کاذب که آن را به گزینه‌ای مناسب برای مدیریت توازن بار در محیط‌های ناهمگن و پویا مانند رایانش مه بدل می‌کند. تحلیل مقایسه‌ای پژوهش‌های پیشین:

جدول ۱: تحلیل مقایسه‌ای پژوهش‌های پیشین
Table 1: Comparative Analysis of Previous Studies

محدودیت‌ها	مزایا	روش	سال	منبع
غیرتطبیقی	ساده و سریع	Dynamic RR	۲۰۲۱	[1]
ناپایدار در مقیاس بزرگ	کاهش تأخیر	PSO Adaptive	۲۰۲۲	[2]
هزینه محاسباتی بالا	بهبود سرعت همگرایی	Hybrid GWO-GA	۲۰۲۳	[5]
نیاز به تنظیم دقیق پارامترها	کنترل هوشمند در شبکه	SDN + HHO	۲۰۲۴	[7]
یادگیری زمان‌بر	پیش‌بینی بار دقیق	AI-driven	۲۰۲۴	[8]
درحال‌توسعه آزمایشی	تعادل پویا، کاهش تأخیر، بهبود توازن بار	PFA	۲۰۲۵	این مقاله

در نتیجه، می‌توان گفت که پژوهش حاضر با تمرکز بر طراحی یک الگوریتم فراابتکاری جدید و تطبیقی، گامی در جهت پر کردن شکاف میان الگوریتم‌های سنتی و هوشمند در محیط‌های پویا و ناهمگن رایانش مه بر می‌دارد. توازن باریکی از مسائل بنیادی در رایانش ابری و مه است که نقش مهمی در افزایش بهره‌وری، کاهش تأخیر و جلوگیری از اضافه‌بار منابع ایفا می‌کند. در مطالعات اولیه، الگوریتم‌های کلاسیک متعددی برای این منظور ارائه شده‌اند که عمدتاً در دودسته ایستا و پویا قرار می‌گیرند [۱،۸،۴۶]از نظر معماری، روش های کلاسیک می توانند متمرکز یا توزیع شده باشند. در رویکرد متمرکز، یک گره مرکزی وظیفه تصمیم گیری را بر عهده دارد که مدیریت را ساده می سازد، اما خطر تک نقطه شکست در آن وجود دارد. در مقابل، رویکرد توزیع شده با تقسیم مسئولیت میان چندین گره، قابلیت تحمل خطا و مقیاس پذیری بیشتری را فراهم می کند، هرچند سربار ارتباطی و پیچیدگی بیشتری دارد [۶،۷،۴۴]. علاوه بر این، رویکردهای هیبریدی نیز معرفی شده اند که با ترکیب روش های ایستا و پویا یا متمرکز و توزیع شده تلاش می کنند نقاط ضعف هر کدام را برطرف کنند [۱۵،۳۳]با وجود اهمیت روش های کلاسیک در بنیان گذاری حوزه توازن بار، محدودیت های آن ها در مواجهه با محیط های پویا موجب شده است پژوهشگران به سمت بهره گیری از روش های هوشمند و الگوریتم های فرا ابتکاری حرکت کنند تا بتوانند راهکارهایی با بهره وری و انعطاف پذیری بالاتر ارائه دهند [۱۶،۱۷،۳۴].

۲-۱. روش‌های کلاسیک توازن بار:

روش‌های کلاسیک مانند Round Robin به دلیل سادگی و سهولت پیاده‌سازی در محیط‌های ایستا به کار گرفته شده‌اند [۲۶،۲۸،۳۰]. در الگوریتمRound Robin ، وظایف به‌صورت چرخشی میان‌گره‌ها توزیع می‌شوند و هدف آن توزیع یکنواخت بار کاری است، بدون درنظرگرفتن ظرفیت واقعی یا وضعیت لحظه ای هر گره [۲۷،۲۹]. الگوریتم Weighted Round Robin نسخه پیشرفته تری است که وزن گره ها را لحاظ می‌کند و وظایف را نسبت به توان پردازشی هر گره توزیع می کند، اما هنوز اطلاعات لحظه ای سیستم را مد نظر قرار نمی دهد [۲۳،۲۶].

۲-۲. الگوریتم های فرابتکاری:

الگوریتم‌های فراابتکاری به دلیل توانایی ذاتی در حل مسائل بهینه‌سازی پیچیده و محیط‌های پویا، به‌عنوان گزینه‌ای مؤثر برای توازن بار در رایانش مه مطرح شده‌اند[۱۲،۲۴،۴۸]. این الگوریتم‌ها با شبیه‌سازی رفتارهای طبیعی یا الهام از فرایندهای بیولوژیکی و فیزیکی، توانایی پیداکردن راهکارهای بهینه در فضای بزرگ و غیرخطی مسئله را دارند.

۲-۳. روش‌های نوین و ترکیبی:

روش‌های نوین و ترکیبی برای توازن بار در رایانش مه بر اساس فناوری‌های پیشرفته مانند یادگیری عمیق، SDNو معماری‌های توزیع‌شده توسعه یافته‌اند و تلاش می‌کنند محدودیت‌های روش‌های کلاسیک و حتی فراابتکاری را برطرف کنند. به‌طورکلی، روش‌های نوین و ترکیبی با استفاده از فناوری‌های پیشرفته توانسته‌اند انعطاف‌پذیری، دقت و پایداری بالاتری نسبت به روش‌های کلاسیک و حتی برخی الگوریتم‌های فراابتکاری ارائه دهند، اما چالش‌هایی همچون نیاز به منابع محاسباتی زیاد، پیچیدگی پیاده‌سازی و محدودیت مقیاس‌پذیری همچنان در مسیر استفاده گسترده آن‌ها وجود دارد.

۲-۴. شکاف تحقیقاتی در الگوریتم ها:

علی‌رغم پیشرفت‌ها، روش‌های موجود با چالش‌هایی مواجه هستند که در جدول ۲ به آن پرداخته شده:

جدول ۲: مزایا و معایب

Table 2: Advantages and Disadvantages

نام الگوریتم	معایب	مزایا	مشکل و شکاف تحقیقاتی	توضیح	منبع
PSO	گیرافتادن در نقاط محلی، نیاز به تنظیم دقیق پارامترها	جستجوی مؤثر در فضای بهینه‌سازی، پیاده‌سازی ساده	همگرایی زودهنگام در محیط‌های بزرگ و پویا	در مسائل پیچیده بهره‌وری پایین دارند.	[9]
ACO	افزایش هزینه محاسباتی با تعداد گره‌ها و وظایف بالا	توانایی یافتن مسیرهای بهینه، الگوریتم الهام‌گرفته از طبیعت	سربار محاسباتی بالا در مقیاس بزرگ	در مسائل پیچیده بهره‌وری پایین دارند.	[10]
GWO	ناپایدار در محیط‌های پویا	بهبود تأخیر و بهره‌وری منابع، تعادل بین اکتشاف و بهره‌برداری	ناپایداری در شرایط بار متغیر	دربارهای متغیر نمی‌توانند پایدار باشند و ناپایداری زیادی دارند	[12]

WOA	نیاز به بهینه‌سازی بیشتر در محیط‌های متغیر	تعادل مناسب بین جستجوی محلی و جهانی، مقابله با مسائل پیچیده	ناپایداری در سناریوهای پویا	دربارهای متغیر نمی‌توانند پایدار باشند و ناپایداری زیادی دارند	[13]
روش‌های یادگیری عمیق	مصرف بالای منابع، زمان آموزش طولانی	پیش‌بینی دقیق بار، قابلیت یادگیری الگوهای پیچیده	نیاز به منابع محاسباتی بالا و زمان آموزش طولانی	به منابع زیادی نیاز دارند تا نیازها را برطرف کنند	[42]
SDN	پیاده‌سازی پیچیده و هزینه‌بر	کنترل متمرکز شبکه، افزایش پایداری و انعطاف‌پذیری	پیچیدگی پیاده‌سازی و هزینه تجهیزات	به منابع زیادی نیاز دارند تا نیاز ها را برطرف کنند	[40]
الگوریتم‌های ترکیبی و نوین	پیچیدگی محاسباتی، نیاز به تنظیم دقیق پارامترها	ترکیب مزایای الگوریتم‌های مختلف، انعطاف‌پذیری بالا	پیچیدگی محاسباتی و محدودیت مقیاس‌پذیری	تنها بر یک یا دو معیار متمرکز است	[31]-[32]-

۳. مبانی نظری، مفهومی و پیشینه علمی:

۳-۱. مفهوم رایانش مه:

رایانش مه به‌عنوان یک پارادایم محاسباتی نوظهور، باهدف پاسخگویی به نیازهای پردازش بلادرنگ و کم تأخیر در اکوسیستم اینترنت اشیاء توسعه و برخلاف رایانش ابری با توزیع قابلیت های محاسباتی، ذخیره سازی و تحلیل در لبه دستگاه اینترنت اشیا امکان پردازش محلی و سریع را فراهم می سازد [۱۶،۱۸،۲۰]. این ویژگی رایانش مه نه تنها تأخیر را به طور قابل توجهی کاهش می دهد، بلکه مصرف پهنای باند شبکه را بهینه کرده و امنیت داده ها را بهبود می بخشد و بهره وری انرژی را در کاربردهای حساس افزایش می دهد[۴۱،۴۷]رشد روزافزون دستگاه های اینترنت اشیاء و نیاز به پردازش های مقیاس پذیر و بلادرنگ، رایانش مه را به یک زیرساخت کلیدی تبدیل کرده است. با این حال، نا همگنی منابع گره های مه ، پویایی بارهای کاری و محدودیت های زیرساختی، چالش های متعددی را در پیاده سازی کارآمد این پارادایم ایجاد کرده اند[۱۵،۲۹]

۳-۲. ضرورت توازن بار در رایانش مه:

توازن بار در رایانش مه فرایندی است که هدف آن تخصیص بهینه وظایف به گره‌های پردازشی است تا از تراکم بیش از حد در برخی گره‌ها جلوگیری شده و منابع به‌صورت یکنواخت و کارآمد استفاده شوند[۱۵،۳۰،۳۵]. عدم مدیریت صحیح توازن بار می‌تواند پیامدهای متعددی داشته باشد؛ روش‌های کلاسیک Round Robin به دلیل ماهیت ایستا و عدم توانایی در مدیریت پیچیدگی‌های محیط مه، بهره‌وری محدودی دارند [۲۰] این روش‌ها معمولاً فرض می‌کنند که گره‌ها همگن هستند و بار کاری به‌صورت یکنواخت توزیع می‌شود، درحالی‌که در محیط‌های واقعی، گره‌های مه از نظر ظرفیت محاسباتی و انرژی بسیار متنوع هستند [۲۹]

۳-۳. الگوریتم های فرابتکاری در توازن بار:

الگوریتم‌های فراابتکاری به دلیل حل مشکل بهینه‌سازی چندمعیاره و خطی در مدیریت توازن بار در رایانش مه زیاد موردتوجه قرار گرفته است[۱۲،۱۳،۲۱]. در ادامه، بعضی از الگوریتم‌هایی که به‌عنوان الگوریتم مرجع استفاده شده معرفی میشود:

۳-۳-۱. الگوریتم کلونی مورچه :

این الگوریتم با الهام از رفتار جستجوی مسیر مورچه‌ها، به کمک ترشح فرمون برای یافتن بهترین مسیر برای تخصیص وظایف استفاده می‌کند، این الگوریتم در سناریو های با تعداد گره های کوچکتر دارای بهره وری بهتری هستند ولی در تعداد گره های زیاد بدلیل بروز رسانی مسیر و فرمون ها نیاز به زمان زیاد تری دارد و بهره وری آن کاهش میابد [۱۰،۲۵،۴۸].

۳-۳-۲. الگوریتم بهینه‌سازی ازدحام ذرات:

. الگوریتم بهینه‌سازی ازدحام ذرات با شبیه‌سازی حرکت ذرات در فضای جستجو و به روز رسانی موقعیت ها بر اساس بهترین تجربه فردی و جمعی، برای مسائل پیوسته و گسسته مناسب است. این الگوریتم به دلیل سرعت هم گرایی بالا در مسائل ساده محبوب است، اما در فضاهای جستجوی پیچیده، احتمال گیر افتادن در بهینه های محلی دارد [۹،۱۴،۳۶]

۳-۳-۳. الگوریتم گرگ خاکستری :

الگوریتم گرگ خاکستری با مدل‌سازی سلسله‌مراتب اجتماعی و استراتژی شکار گرگ‌ها، تعادل مناسبی بین اکتشاف و استثمار ایجاد می‌کند. این الگوریتم در کاهش تأخیر و بهبود بهره‌وری منابع در رایانش مه موفق بوده است، اما با تغییرات شدید بار بهره‌وری خود پایین می‌آید [۱۰].

۳-۳-۴. الگوریتم زنبور عسل مصنوعی :

الگوریتم زنبورعسل مصنوعی با شبیه‌سازی رفتار زنبورهای کارگر، ناظر و جستجوگر، منابع باارزش را حفظ و منابع بی‌ارزش را حذف می‌کند. این الگوریتم دارای انعطاف زیادی در مقابل مسائل پیچیده برای توازن بار در رایانش مه می باش [۱۰،۱۲]

۳-۳-۵. الگوریتم نهنگ:

الگوریتم نهنگ با الهام از الگوهای شکار نهنگ‌ها تعادل خوبی بین جستجوی محلی و جهانی ارائه می‌دهد. این الگوریتم در تخصیص وظایف در رایانش مه بهره‌وری بالایی دارد، اما در سناریوهای با تغییرات پویا، دقت و پایداری آن ممکن است کاهش یابد [۳،۱۰]

۳-۳-۶. رویکردهای نوین و ترکیبی:

در سال‌های اخیر، رویکردهای ترکیبی و مبتنی بر فناوری‌های نوین مانند یادگیری عمیق و شبکه‌های نرم‌افزار محور برای توازن بار در رایانش مه پیشنهاد شده‌اند.

۳-۴. چالش‌های موجود و جایگاه الگوریتم بهینه‌سازی روباه قطبی[2]

باوجود پیشرفت‌های قابل‌توجه در الگوریتم‌های فراابتکاری، چالش‌های متعددی همچنان مانع از دستیابی به توازن بار کارآمد در رایانش مه می‌شوند:۱.گیر افتادن در بهینه‌های محلی [۹]۲. ناپایداری در محیط‌های پویا[۱۰،۱۱]۳.سربار محاسباتی بالا[۳۵،۳۶]۴. نیاز به تنظیم دقیق پارامترها [۱۵] ۵.عدم جامعیت در معیارها[۳۱،۳۸]الگوریتم بهینه‌سازی روباه قطبی در دسته‌ی الگوریتم‌های الهام‌گرفته از شکارچیان قرار می‌گیرد[۵]. آنچه الگوریتم بهینه‌سازی روباه قطبی را واقعاً متمایز می کند، مجموعه ای از نوآوری های کلیدی است که آن را نسبت به سایر الگوریتم های فرا ابتکاری برجسته می سازد. نخستین ویژگی مهم، الهام زیستی خاص و منحصر به فرد این الگوریتم است؛ برای اولین بار رفتار روباه قطبی به عنوان منبع الهام استفاده شده ، بنابراین الگوریتم بهینه‌سازی روباه قطبی جنبه نوآورانه و تازه‌ای در طراحی الگوریتم‌های بهینه‌سازی ارائه می‌دهد. دومین ویژگی، مدل‌سازی دقیق دو فاز شکار است؛ الگوریتم با ترکیب فرآیند شنود برای شناسایی نواحی امیدبخش و پرش شکار برای بهره‌برداری دقیق، توانسته همزمان اکتشاف و بهره‌برداری را تقویت کند و کیفیت راه‌حل‌ها را افزایش دهد. سومین نوآوری، تقسیم‌بندی رفتاری جمعیت است؛ روباه‌ها به گروه‌های مختلف شامل رهبرمحور، تجربه‌محور، آزاد و سخت‌کوش تقسیم شده‌اند، که این امر موجب پویایی جمعیت، افزایش تنوع حرکت و جلوگیری از همگرایی زودرس می‌شود و الگوریتم را در مواجهه با مسائل پیچیده انعطاف‌پذیرتر می‌کند. نهایتاً، مکانیزم خستگی و انگیزش یکی دیگر از نوآوری‌های مهم الگوریتم بهینه‌سازی روباه قطبی است که فرآیندهای طبیعی مانند خستگی روباه یا تحریک توسط رهبر را شبیه‌سازی می‌کند؛ این مکانیزم تنوع جمعیت را در طول تکرارها حفظ کرده و پایداری الگوریتم را تضمین می‌کند. ترکیب این چهار نوآوری، الگوریتم بهینه‌سازی روباه قطبی را به یک الگوریتم بسیار توانمند، پویا و مناسب برای مسائل پیچیده بهینه‌سازی تبدیل کرده است .

۴. طراحی، تشریح و مدل‌سازی الگوریتم بهینه‌سازی روباه قطبی:

۴-۱. مقدمه و الهام‌گیری زیستی

الگوریتم پیشنهادی بهینه‌سازی روباه قطبی بر پایهٔ رفتار شکار روباه قطبی در شرایط سخت قطب شمال طراحی شده است. روباه قطبی برای بقا در محیطی با دماهای بسیار پایین و منابع غذایی محدود، راهبردهای منحصربه‌فردی را تکامل داده است. مهم‌ترین ویژگی‌های الهام بخش برای طراحی الگوریتم عبارت‌اند از: قدرت شنوایی خارق‌العاده :روباه قطبی قادر است صدای جوندگان کوچک را در زیر برف‌های ضخیم شناسایی کند. این توانایی در الگوریتم به‌صورت حساسیت به موقعیت‌های مناسب

[2] Polar Fox Optimization – PFA

مدل می‌شود.پرش شکار :(Hunting Leap) روباه پس از شناسایی موقعیت طعمه، یک پرش بلند و دقیق انجام می‌دهد و از بالا به زیر برف حمله می‌کند. این رفتار در الگوریتم معادل با حرکت جهشی به سمت بهترین جواب‌های احتمالی است زندگی گروهی و تقسیم نقش‌ها :روباه‌ها معمولاً در قالب گروه‌های کوچک فعالیت می‌کنند و هر کدام نقش متفاوتی در شکار دارند.
این سه اصل زیستی اساس طراحی الگوریتم بهینه‌سازی روباه قطبی هستند که در ادامه به‌صورت ریاضی مدل‌سازی می‌شوند.

۴-۲.نمایش ریاضی مسئله توازن بار

فرض کنید مسئله بهینه‌سازی دارای ابعاد dباشد و محدوده جستجو بین کران پایین LBو کران بالا UBتعریف شود. در این صورت، موقعیت هر روباه (هر جواب کاندید) به‌صورت برداری در فضای d - بعدی تعریف می‌شود:

$$X_i = \{x_{i1}, x_{i2}, \dots, x_{id}\}, i = 1,2, \dots, N \quad (1)$$

که در آن N تعداد روباه‌ها (جمعیت اولیه) است .
مقداردهی اولیه به‌صورت تصادفی و یکنواخت در فرمول شماره ۲ انجام می‌شود:

$$x_{ij} = LB_j + rand(0,1) \cdot (UB_j - LB_j) \quad (2)$$

مقدار کمتر LB نشان‌دهنده توازن بار بهتر میان‌گره‌ها است

۴-۳.تقسیم نقش‌ها در جمعیت:

برای ایجاد تنوع رفتاری و جلوگیری از همگرایی زودرس، روباه‌ها به چهار گروه رفتاری تقسیم می‌شوند:

۱. روباه‌های آزاد :(Free Searchers) به‌صورت تصادفی محیط را کاوش می‌کنند (اکتشاف).
۲. روباه‌های رهبر محور :(Leader Followers) موقعیت رهبر گروه را دنبال می‌کنند (بهره‌برداری).
۳. روباه‌های تجربه‌محور (Experience-based) بر اساس تجربیات گذشته و بهترین موقعیت‌های ذخیره‌شده عمل می‌کنند.
۴. روباه‌های سخت‌کوش :(Hard-workers) با حرکات آهسته‌تر و دقیق‌تر به جستجوی موضعی می‌پردازند.

۴-۳-۱. حرکت تجربه‌محور:

هر روباه می‌تواند بر اساس تجربه‌ی گذشته خود حرکت کند. اگر P قدرت پرش و D جهت حرکت باشد، موقعیت جدید به‌صورت زیر به‌روزرسانی می‌شود:

$$X_i^{t+1} = X_i^t + P \cdot D \quad (3)$$

که در فرمول ۳ :

- P به‌صورت تصادفی یا وابسته به کیفیت جواب‌های قبلی تعیین می‌شود،
- $D = \cos(\theta)$ جهت حرکت با زاویه‌ی تصادفی $\theta \in [۰, ۱۸۰^\circ]$ این بخش معادل پرش آزمایشی روباه‌ها است.

۴-۳-۲. حرکت رهبرمحور:

هر قلاده از روباه‌های قطبی یک رهبر دارد که قلاده را در رسیدن به هدفش هدایت می‌کند رهبر گروه (بهترین روباه با بالاترین برازندگی) سایر اعضا را هدایت می‌کند. حرکت اعضای گروه بر اساس موقعیت رهبر به‌صورت زیر مدل می‌شود:

$$X_i^{t+1} = X_i^t + r \cdot (X_{leader}^t - X_i^t) \quad (4)$$

که در فرمول ۴ $r \in [-۱,۱]$ یک ضریب تصادفی است.
این مرحله باعث می‌شود جمعیت به سمت بهترین جواب‌های فعلی همگرا شود.

۴-۳-۳. فاز انگیزش رهبر:

وقتی روباه‌های قطبی نمی‌توانند به طور متوالی طعمه پیدا کنند رهبر آن را تحریک می‌کند و اعضا موقعیت خود را به طور تصادفی تغییر می‌دهد اگر جمعیت برای چندین تکرار متوالی پیشرفت نکند (همگرایی کاذب رخ دهد)، رهبر گروه سایر اعضا را وادار به بازآرایی تصادفی می‌کند:

$$X_i^{t+1} = LB + rand(0,1) \cdot (UB - LB) \quad (5)$$

این سازوکار مشابه تنوع‌بخشی اجباری است و از گرفتار شدن الگوریتم در بهینه‌های محلی جلوگیری می‌کند. در جدول زیر تمام پارامترهای فرمول‌های بالا توضیح داده شده است.

جدول ۳: جدول پارامترهای الگوریتم بهینه‌سازی روباه قطبی

Table 3: Parameter Table of the Polar Fox Optimization Algorithm

واحد / نوع داده	مقدار یا بازه استفاده‌شده در شبیه‌سازی	توضیح پارامتر(Parameter Description)	نماد(Symbol)
عدد صحیح	10تا ۵۰	تعداد کل گره‌های مه (Fog Nodes) در محیط شبیه‌سازی	(N)
ثانیه	0.2 – 2.5	زمان اجرای تسک (i) روی گره مه	(T_i)
کیلوبایت	100 – 1000	حجم بار یا تسک تخصیص‌یافته به گره(i)	(L_i)
GHz	0.5 – 3.0	ظرفیت پردازشی گره مه(i)	(C_i)
ژول(J)	0.1 – 1.5	انرژی مصرفی گره (i) در پردازش وظایف	(E_i)
ژول	محاسبه‌شده	میانگین انرژی مصرفی در کل گره‌ها	(E_{avg})
ثانیه	محاسبه‌شده	میانگین تأخیر پاسخ وظایف در شبکه مه	(T_{avg})
بدون واحد	0.0 – 1.0	شاخص توازن بار(Load Balance Index)	(LB)
عدد حقیقی	بردار پویا	موقعیت عامل (روباه قطبی) شماره (i) در تکرار(t)	(X_i^t)
عدد حقیقی	متغیر پویا	موقعیت بهترین روباه در جمعیت تا تکرار(t)	(R_{best})
بدون واحد	0.4 – 0.9	ضریب کنترل حرکت جهت‌دار روباه	(\alpha)
بدون واحد	0.2 – 0.6	ضریب تصادفی‌سازی(Randomization Coefficient)	(\beta)
–	0 – 1	عدد تصادفی یکنواخت بین ۰ و ۱ برای تنوع جمعیت	(rand())
بدون واحد	$F = w_1 \cdot \hat{D} + w_2 \cdot \hat{E} + w_3 \cdot (1 - \widehat{LB})$	تابع برازندگی کلی(Fitness Function)	(F)
بدون واحد	(w_1=0.4, w_2=0.3, w_3=0.3)	ضرایب وزنی برای بهینه‌سازی چندهدفه (تأخیر، انرژی، توازن بار)	(w_1, w_2, w_3)
عدد صحیح	100	حداکثر تعداد تکرار الگوریتم	(Iter_{max})
عدد صحیح	30	اندازه جمعیت روباه‌ها (تعداد عامل‌ها)	(Pop_size)
بدون واحد	محاسبه‌شده	بهترین مقدار تابع هدف در هر تکرار	(Fitness_{best})
ثانیه	1 – 5	زمان کل محاسبه برای هر تکرار الگوریتم	(T_{comp})
Mb/s	وابسته به نرخ پهنای باند	هزینه انتقال داده بین گره‌ها	(C_{trans})

در فرآیند بهینه‌سازی الگوریتم بهینه‌سازی روباه قطبی ، انتخاب مقادیر اولیه پارامترها نقش مهمی در پایداری و سرعت همگرایی دارد. مقداردهی نامناسب می‌تواند منجر به انحراف الگوریتم یا همگرایی زودرس شود. در این پژوهش، پارامترها بر اساس تحلیل تجربی و توصیه‌های مطالعات مشابه تنظیم شده‌اند

* ضرایب حرکتی(α و β):ضریب αجهت کنترل شدت حرکت جهت‌دار روباه و تعادل میان فازهای اکتشاف و بهره‌برداری به کار می‌رود. مقدار زیاد آن سبب جهش‌های گسترده در فضای جستجو می‌شود (افزایش تنوع، کاهش دقت)، درحالی‌که مقدار کم موجب گیر افتادن در کمینه‌های محلی می‌گردد. بر اساس آزمایش‌های مقدماتی، مقدار بهینه $\alpha = 0.7$برای محیط‌های پویا انتخاب شد ضریب β عامل تصادفی‌سازی است که پویایی و گریز از همگرایی زودرس را تقویت می‌کند؛ مقدار $\beta = 0.4$ تعادلی میان جستجوی سراسری و محلی ایجاد می‌کند.

* ضرایب وزنی تابع برازندگی(w_1،w_2،w_3) : تابع برازندگی الگوریتم به‌صورت چندهدفه طراحی شده است تا سه معیار اصلی را بهینه کند: کاهش تأخیر(T_{avg})، کاهش مصرف انرژی (E_{avg}) و بهبود شاخص توازن بار(LB). تابع برازندگی چندهدفه الگوریتم پیشنهادی در جدول آورده شده است که در آن:

- $\widehat{D}$ =**تأخیر نرمال‌سازی‌شده**
- $\widehat{E}$ =**مصرف انرژی نرمال‌سازی‌شده**

- **$\widehat{LB}$ =شاخص توازن بار نرمال‌سازی‌شده**

برای یکسان‌سازی واحدها، هر معیار با روش *Min–Max Normalization* نرمال‌سازی شده است از آن‌جا که در رایانش مه معمولاً تأخیر اهمیت بیشتری دارد، مقدار وزن‌ها به‌صورت تجربی به شکل $w_1 = 0.4$، $w_2 = 0.3$ و $w_3 = 0.3$ تعیین گردید این مقادیر پس از چندین اجرای آزمون حساسیت (Sensitivity Analysis) بهترین توازن میان سرعت و دقت را نشان دادند.
*اندازه جمعیت و تعداد تکرار ($Iter_{max}$ و Pop_size):اندازه جمعیت Pop_size برابر با ۳۰ در نظر گرفته شد تا بین تنوع جمعیت و هزینه محاسباتی تعادل برقرار شود.بر اساس تحلیل پیچیدگی زمانی الگوریتم، افزایش جمعیت بیش از ۵۰ باعث افزایش چشمگیر زمان اجرا بدون بهبود قابل‌توجه در کیفیت جواب می‌شود. حداکثر تکرار الگوریتم $Iter_{max}$ روی ۱۰۰ تنظیم گردید که با توجه به نمودار همگرایی (بخش نتایج) نشان می‌دهد الگوریتم در کمتر از ۸۰ تکرار به همگرایی پایدار می‌رسد.
در تمامی الگوریتم‌ها برای مقایسه منصفانه، بودجه محاسباتی یکسان اعمال شد:

$$FE = Pop\_size \times Iter\_max = 30 \times 100 = 3000$$

بنابراین مقدارهای زیر برای همه الگوریتم‌ها یکسان‌سازی شدند:

- Pop_size = 30
- Iter_max = 100
- بذر تصادفی (Seed) = 42

دامنه پارامترهای محیط مه :تعداد گره‌ها N بین ۱۰ تا ۵۰ در نظر گرفته شده است تا رفتار الگوریتم در مقیاس‌های کوچک و متوسط ارزیابی شود.پارامترهای ظرفیت پردازشی C_i و بار تخصیصی L_i به ترتیب در بازه‌های ۰٫۵–۳٫۰ GHz و ۱۰۰– ۱۰۰۰ kB انتخاب شده‌اند این بازه‌ها مطابق با داده‌های واقعی منتشرشده هستند. آزمون حساسیت پارامترها: برای اطمینان از پایداری مدل، یک تحلیل حساسیت انجام شد که در آن هر پارامتر در بازه ۲۰٪ مقدار پایه تغییر داده شد. نتایج نشان داد که تابع برازندگی الگوریتم بهینه‌سازی روباه قطبی دارای پایداری عددی بالاست و بیشترین تأثیر مربوط به ضرایب α و w_1 است. بنابراین مقادیر انتخاب‌شده با هدف دستیابی به سرعت همگرایی بالا و پایداری در محیط‌های پویا تثبیت شدند.

۴-۳-۴ .فاز جهش و جایگزینی

میزان مرگ و میر در بین جمعیت روباه های قطبی بالا است والدین اغلب توله های خود را رها میکنند و توله های مهاجم گاهی اوقات خواهر و برادر خود را میکشند بیماری هاری تا ۲۰ درصد از جمعیت روباه های قطبی را از بین میبرد .بخشی از روباه‌های ضعیف (راه‌حل‌های با برازندگی پایین) در هر تکرار حذف و با روباه‌های جدید جایگزین می‌شوند. این مرحله شبیه‌سازی طبیعی مرگ و زایش در جمعیت است و تنوع الگوریتم را در طول زمان حفظ می‌کند.

۵-۳-۴ .شبیه‌سازی خستگی

روباه‌ها پس از تکرارهای متعدد شکار دچار خستگی می‌شوند. در الگوریتم، این مفهوم به شکل کاهش تدریجی سهم برخی گروه‌ها مدل‌سازی می‌شود. به این معنا که وزن اختصاص‌داده‌شده به گروه‌های خاص (مثلاً رهبرمحور یا سخت‌کوش) پس از چند تکرار کم می‌شود تا سایر گروه‌ها شانس بیشتری برای جستجو داشته باشند.

۴-۴. شبه‌کد الگوریتم :

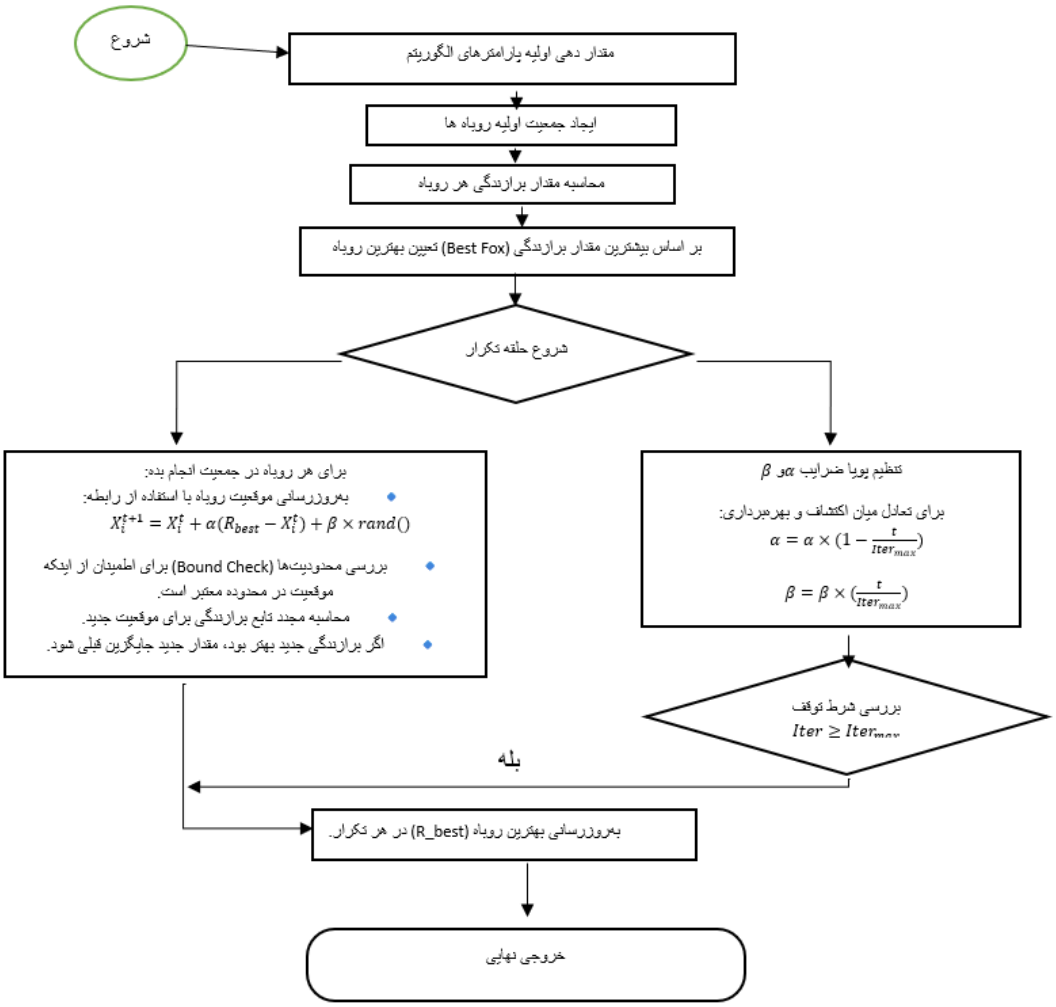
شبه کد زیر مراحل اجرایی الگوریتم بهینه‌سازی روباه قطبی را برای توازن بار در رایانش مه نشان می‌دهد:

```
Algorithm 1: Polar Fox Optimization (PFA) for Load Balancing
Input:  Task set T = {t1, t2, …, tn}, Fog nodes N = {n1, n2, …, nk}
Output: Optimized load distribution matrix M*
1.  Initialize parameters: α, β, w1, w2, w3, Pop_size, Iter_max
2.  Randomly initialize population of foxes Xi (i = 1,…, Pop_size)
3.  Evaluate initial fitness F(Xi) for all individuals using Eq.(4)
4.  Identify the best solution R_best among the population
5.  For iteration t = 1 to Iter_max do
6.      For each fox i in population do
7.          Update position using:
                Xi(t+1) = Xi(t) + α*(R_best - Xi(t)) + β*rand()
8.          Bound check: ensure Xi(t+1) within valid search limits
9.          Evaluate new fitness F(Xi(t+1))
10.         If F(Xi(t+1)) better than previous, update Xi(t)
11.     End For
12.     Update global best R_best
13.     Adjust α and β dynamically:
                α = α * (1 - t / Iter_max)
                β = β * (t / Iter_max)
14. End For
15. Return final optimized load allocation matrix M* = R_best
```

الگوریتم پیشنهادی بر پایه‌ی رفتار شکار روباه قطبی (Vulpes Lagopus) در شرایط سخت قطب شمال طراحی شده است. روباه قطبی برای بقا در محیطی با دماهای بسیار پایین و منابع غذایی محدود، راهبردهای منحصربه‌فردی را تکامل داده است. باتحلیل این الگوریتم به موارد زیر میتوان اشاره کرد:

- اکتشاف قوی :وجود روباه‌های آزاد و تجربه‌محور باعث می‌شود الگوریتم بتواند فضای جستجو را به‌خوبی پوشش دهد.
- بهره‌برداری مؤثر :روباه‌های رهبرمحور و سخت‌کوش موجب می‌شوند
- مکانیزم ضد بهینه محلی :وجود مراحل انگیزش رهبر و جایگزینی از همگرایی زودرس جلوگیری می‌کند.
- پیچیدگی محاسباتی :با توجه به ساختار ساده به‌روزرسانی‌ها، پیچیدگی الگوریتم تقریباً $O(N \cdot d)$ در هر تکرار است که برای مسائل مهندسی با ابعاد بزرگ نیز قابل استفاده است.

۴-۵. دیاگرام جریان الگوریتم:



شکل ۱:فلوچارت الگوریتم

Figure 1: Algorithm Flowchart

در شکل ۱ این دیاگرام مراحل تصمیم‌گیری (اکتشاف، کمین، حمله) و حلقه‌های تکرار را به‌صورت بصری نمایش می‌دهد و به درک بهتر فرآیند بهینه‌سازی کمک می‌کند [۵۶].

برای بروز رسانی موقعیت:

$$X_i^{t+1} = X_i^t + \alpha \cdot rand \cdot (R_{best} - X_i^t) \quad (۶)$$

برای فاز کمین

$$X_i^{t+1} = X_i^t + \beta \cdot (X_i^t - X_j^t) \quad (۷)$$

و برای جهش تصادفی

$$X_i^{t+1} = X_i^t + \gamma \cdot L \quad (۸)$$

که Lپرش Levy و γ ضریب جهش است.

۵. شبیه‌سازی، مقایسه و تحلیل نتایج:

۵-۱. محیط شبیه‌سازی

تمام الگوریتم‌ها در محیط Python پیاده‌سازی شده‌اند و نسخه‌های کتابخانه‌ها به‌صورت زیر است:

- **Python:** 3.10
- **NumPy:** 1.26
- **SciPy:** 1.11
- **Pandas:** 2.0
- **Matplotlib:** 3.7
- **scikit-learn:** 1.3
- **memory-profiler:** 0.61 (برای اندازه‌گیری مصرف حافظه)
- **time/perf_counter:** برای اندازه‌گیری زمان اجرا Python تابع استاندارد

از سیستم سخت افزاری و نرم افزاری زیر استفاده شده است.

- **پردازنده (CPU):** Intel Core i7-12700H (14 هسته‌ای، ۲۰ رشته‌ای)
- **حافظه RAM:** 16 گیگابایت DDR4
- **کارت گرافیک (در صورت استفاده):** NVIDIA RTX 3050
- **فضای ذخیره‌سازی:** NVMe SSD
- **سیستم‌عامل:** Windows 11 Pro 64-bit

که به دلیل قابلیت‌های محاسباتی و ابزارهای تحلیل داده، برای این منظور مناسب هستند. محیط شبیه‌سازی شامل سناریوهای مختلف با تعداد متغیر گره‌های مه و وظایف بود تا پویایی و مقیاس‌پذیری سیستم‌های اینترنت اشیاء واقعی شبیه‌سازی شود . مشخصات محیط شبیه‌سازی به‌صورت زیر است:

- گره‌های مه: گره‌ها با ظرفیت‌های محاسباتی ناهمگن و محدودیت‌های انرژی تعریف شدند .
- وظایف: وظایف با نیازهای محاسباتی تصادفی و نیازمندی‌های پهنای باند تولید شدند .
- شبکه: توپولوژی شبکه به‌صورت توزیع‌شده با تأخیرهای انتقال تصادفی مدل‌سازی شد [۹, ۱۷].
- ابزار شبیه‌سازی: کتابخانه‌های SimPy در Python و ابزارهای شبیه‌سازی شبکه در MATLAB برای مدل‌سازی ترافیک شبکه و پردازش وظایف استفاده شدند.

۵-۲. تنظیمات مسئله و پارامترهای ورودی

تنظیمات مسئله و پارامترهای ورودی در جدول ۴ آمده است.

جدول۴: پارامترهای ورودی
Table 4: Input Parameters

نام	توضیح
تعداد وظایف:(Tasks)	۱۰۰، ۲۰۰و ۵۰۰
تعداد گره‌های مه:(Fog Nodes)	۵، ۱۰، ۲۰
ظرفیت گره‌ها:	به‌صورت تصادفی بین [۱۰، ۵۰] واحد محاسباتی
پارامترهای الگوریتم	PFA
جمعیت اولیه:	۳۰
تعداد تکرار:	۱۰۰
ضرایب وزنی	$\gamma = 0.2 \alpha = 0.5\ \beta = 0.3$

۵-۳. الگوریتم‌های مقایسه‌ای

به‌منظور ارزیابی عملکرد الگوریتم بهینه‌سازی روباه قطبی ، این الگوریتم با ۶ الگوریتم متداول مقایسه شده است و در جدول ۵ مزایا و معایب روش‌های موجود را خلاصه می‌کند.

جدول ۵: مقایسه مزایا و معایب

Table 5: Comparison of Advantages and Disadvantages

الگوریتم	کاربرد در رایانش مه	معایب	مزایا	منبع
Round Robin	مناسب برای محیط‌های ایستا	عدم انطباق با پویایی و ناهمگنی	سادگی، پیاده‌سازی آسان	۲۱
PSO	سناریوهای کوچک و ایستا	گیر افتادن در بهینه‌های محلی	سرعت همگرایی بالا	۲
ACO	تخصیص وظایف محدود	سربار محاسباتی بالا	یافتن مسیرهای بهینه	۳
GWO	پویایی متوسط	ناپایداری در بارهای متغیر	تعادل اکتشاف و استثمار	۵
ABC	مسائل پیچیده با منابع کافی	نیاز به تنظیم پارامتر	مقاومت در برابر بهینه‌های محلی	۴
WOA	بارهای نسبتاً ثابت	ناپایداری در تغییرات پویا	تعادل جستجوی محلی و جهانی	۶
PFA	IoTپویا و مقیاس‌پذیر	نیاز به آزمایش در مقیاس‌های بزرگ	انطباق‌پذیری، ارزیابی چندمعیاره	۱۵

۵-۴. معیارهای ارزیابی

معیار های زیر برای ارزیابی عملکرد الگوریتم بهینه‌سازی روباه قطبی در توازن بار استفاده می شود[۲۴, ۳۱]:

۱. تأخیر متوسط (Average Latency): میانگین زمان لازم برای پردازش و انتقال وظایف، محاسبه‌شده به‌صورت:

$$\text{avg Delay} = \sum_{i=1}^{n} \left(\frac{1}{n}\right) \left(T_{proc}(t_i).\, T_{comm}(t_i)\right) \quad (۹)$$

میانگین تأخیر یکی از شاخص برای ارزیابی عملکرد الگوریتم ها در توازن بار در رایانش مه به حساب می آید. این معیار به صورت مجموع زمان پردازش و زمان ارتباطی وظایف تعریف می‌شود که پس از محاسبه برای تمام وظایف، میانگین آن‌ها به عنوان شاخص نهایی گزارش می‌گردد .

مصرف انرژی (Energy Consumption): مجموع انرژی مصرفی توسط گره‌های مه، محاسبه‌شده به‌صورت:

$$\text{avg energy} = \sum_{i=1}^{n} \left(\frac{1}{n}\right) \left(T_{proc}(t_i).\, P(f_i)\right) \quad (۱۰)$$

مصرف انرژی یکی دیگر از معیارهای بنیادین در ارزیابی عملکرد الگوریتم‌های توازن بار در رایانش مه است. در این رابطه، n تعداد کل وظایف، $T_{proc}(t_i)$مدت زمان پردازش t_i و $P(f_i)$ توان مصرفی گره در فرکانس کاری f_i است.

شاخص توازن بار (Load Imbalance Index): انحراف استاندارد بارهای تخصیص‌یافته به گره‌ها، محاسبه‌شده به‌صورت:

$$LBF = \frac{\sigma(L)}{\mu(L)} \quad (۱۱)$$

شاخص توازن بار یکی از معیارهای کلیدی برای ارزیابی بهره وری الگوریتم‌های تخصیص وظایف در محیط‌های مه و رایانش ابری استدر این رابطه، $\sigma(L)$ انحراف معیار بارها و $\mu(L)$ میانگین بار تخصیص‌یافته به گره‌ها است.

۵-۵ .نتایج عددی و مقایسه

۵-۵-۱. مقایسه میانگین تأخیر:

در این بخش، عملکرد الگوریتم بهینه‌سازی روباه قطبی در معیار میانگین تأخیر در مقایسه با الگوریتم‌های مرجع بهینه‌سازی ازدحام ذرات ، گرگ خاکستری ، ACO، زنبور عسل مصنوعی و نهنگ مورد ارزیابی قرار گرفته است. جدول زیر، مقادیر دقیق میانگین تأخیر را در سه سناریو با ۱۰۰، ۲۰۰ و ۵۰۰ وظیفه نشان می‌دهد:

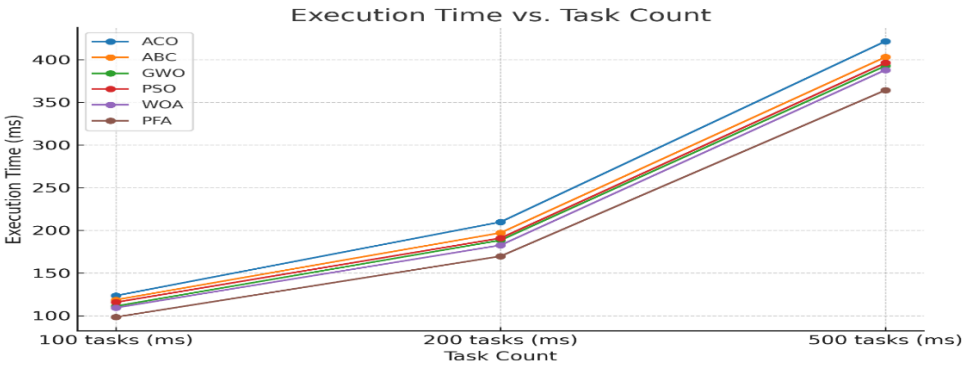
جدول ۶: مقادیر دقیق میانگین تأخیر

Table 6: Precise average delay values

الگوریتم	۱۰۰ وظیفه(ms)	۲۰۰ وظیفه(ms)	۵۰۰ وظیفه(ms)
ACO	123.4	209.8	421.5
ABC	118.6	197.1	403.0
GWO	111.2	188.4	392.7
PSO	115.7	190.9	396.2

WOA	109.3	182.6	387.9
PFA	98.4	169.7	364.2

جدول ۶ مقادیر میانگین تأخیر را در سه سناریوی متفاوت با ۱۰۰، ۲۰۰ و ۵۰۰ وظیفه نشان می‌دهد. همان‌طور که مشاهده می‌شود، الگوریتم بهینه‌سازی روباه قطبی در تمامی حالات عملکرد بهتری نسبت به سایر الگوریتم‌ها داشته است. به‌ویژه در سناریوی سنگین با ۵۰۰ وظیفه، مقدار تأخیر در الگوریتم بهینه‌سازی روباه قطبی برابر با ۳۶۴,۲ میلی‌ثانیه بوده است، در حالی که الگوریتم کلونی مورچه بیشترین تأخیر را با ۴۲۱,۵ میلی‌ثانیه ثبت کرده است.در تصویر زیر ما جدول را به صورت نموداری و عینی آورده شده است.



شکل ۲: زمان اجرا
Figure 2: Execution Time

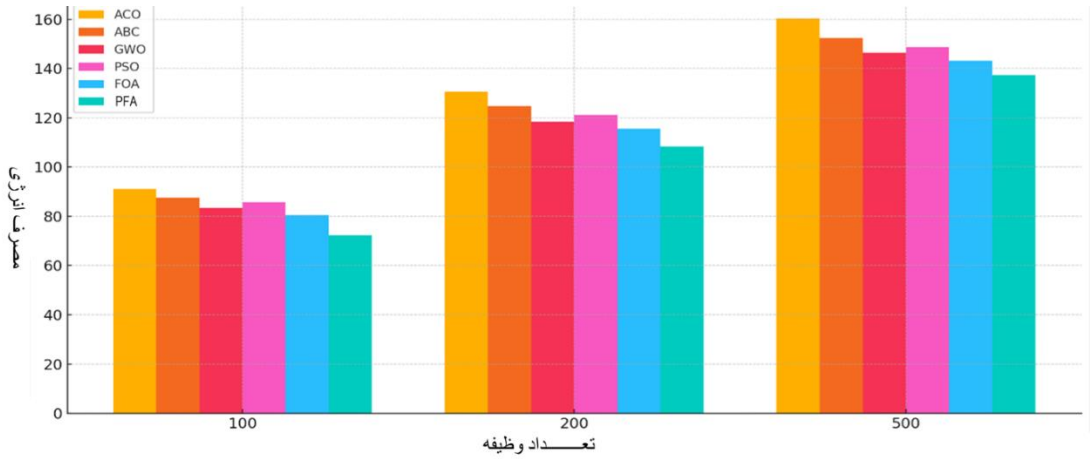
۵-۵-۲. مصرف انرژی(Joule):

مصرف انرژی به عنوان یکی از معیار های اصلی سنجش رو در جدول زیر بررسی کردیم در این پژوهش، الگوریتم بهینه‌سازی روباه قطبی با سایر الگوریتم‌های مرجع در سناریوهای مختلف شامل ۱۰۰، ۲۰۰ و ۵۰۰ وظیفه مقایسه شد تا میزان مصرف انرژی هر الگوریتم مشخص شود. داده‌های به‌دست‌آمده از شبیه‌سازی‌ها، در جدول۷ ارائه شده‌اند:

جدول۷: مصرف انرژی
Table 7: Energy Consumption

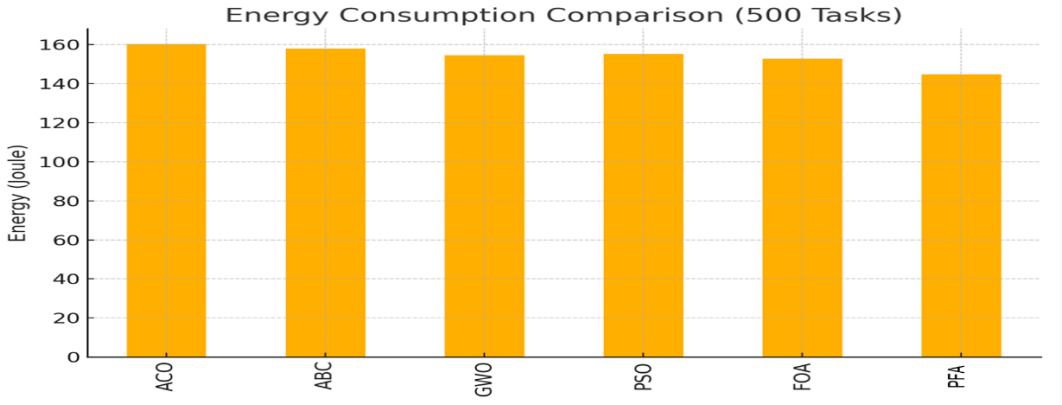
الگوریتم	۱۰۰ وظیفه	۲۰۰ وظیفه	۵۰۰ وظیفه
ACO	39.4ژول	72.1 ژول	160.2 ژول
ABC	38.7 ژول	69.3 ژول	158.0 ژول
GWO	36.8 ژول	67.0 ژول	154.4 ژول
PSO	37.5 ژول	68.1 ژول	155.2 ژول
WOA	35.9 ژول	65.5 ژول	152.9 ژول
PFA	31.8 ژول	61.2 ژول	144.6 ژول

نتایج جدول ۵ نشان می‌دهد که الگوریتم بهینه‌سازی روباه قطبی در تمامی سطوح بار کاری کمترین مصرف انرژی را ثبت کرده است در سناریوی سنگین با ۵۰۰ وظیفه، مصرف انرژی الگوریتم بهینه‌سازی روباه قطبی برابر با ۱۴۴,۶ ژول اندازه‌گیری شد که به‌طور میانگین ۱۰ تا ۱۵ درصد کمتر از الگوریتم‌های دیگر است. این بهبود قابل توجه ناشی از استراتژی‌های هوشمند تقسیم وظایف و تعادل بین اکتشاف و بهره‌برداری در الگوریتم است، که موجب کاهش محاسبات اضافی و بهینه‌سازی منابع می‌شود.



شکل ۳ : نشان دهنده مقایسه مصرف انرژی الگوریتم
Figure 3: Shows the comparison of the algorithm's energy consumption

نمودار بالا نشان‌دهنده مقایسه مصرف انرژی الگوریتم‌ها در سه سطح بار کاری (۱۰۰، ۲۰۰ و ۵۰۰ وظیفه) است. همان‌طور که مشاهده می‌شود، الگوریتم الگوریتم بهینه‌سازی روباه قطبی در هر سه سطح، کمترین مصرف انرژی را دارد که نشان از بهره‌وری بالا و بهینه‌بودن آن در تخصیص وظایف دارد در مقابل، بیشترین مصرف انرژی متعلق به الگوریتم کلونی مورچه با ۱۶۰,۲ ژول است، که دلیل آن سربار محاسباتی بالا و پیچیدگی فرآیند به‌روزرسانی فرمون‌هاست.



شکل۴: مصرف انرژی الگوریتم ها در تخصیص وظایف
Figure 4: Energy consumption of algorithms in task allocation

در شکل شماره ۴، مصرف انرژی الگوریتم ها در تخصیص وظایف (۵۰۰ وظیفه) با هم مقایسه شده است. این نمودار مصرف انرژی هر الگوریتم را برای ۵۰۰ وظیفه به تصویر می‌کشد. محور افقی نام الگوریتم‌ها و محور عمودی میزان انرژی مصرفی (به کیلوژول) را نشان می‌دهد. مقدار انرژی برای هر الگوریتم به صورت یک ستون مجزا ترسیم شده است .در این نمودار الگوریتم الگوریتم بهینه‌سازی روباه قطبی کمترین مصرف انرژی را دارد، در حالی که الگوریتم Round Robin به‌دلیل نداشتن هیچ نوع بهینه‌سازی، بیشترین انرژی را مصرف کرده است.

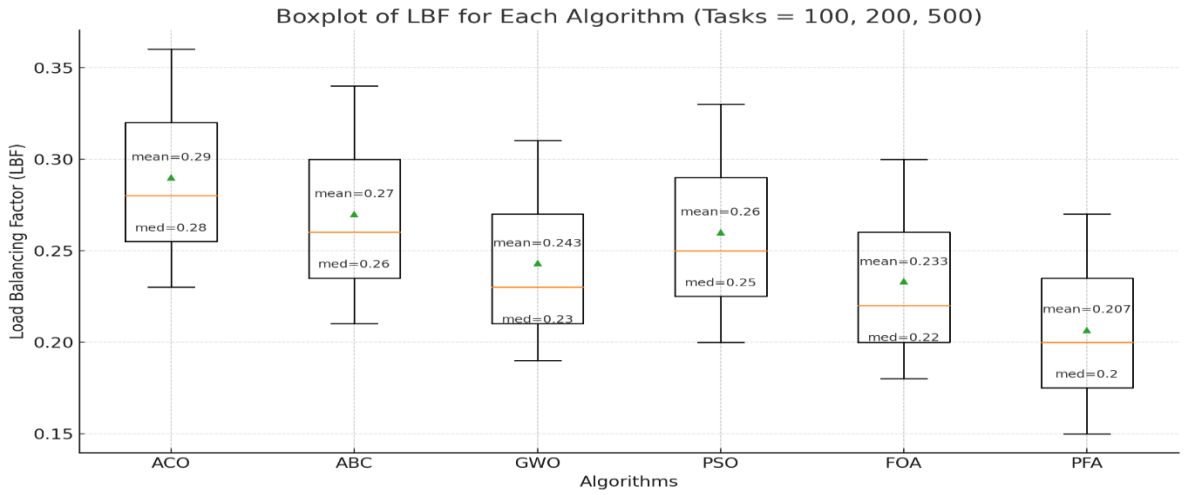
۳-۵-۵. مقایسه شاخص توازن بار(Load Balancing Factor - LBF)

شاخص توازن بار (LBF) معیاری برای سنجش یکنواختی توزیع بار میان گره‌های مه است. این شاخص بر اساس انحراف استاندارد بار پردازشی تخصیص‌یافته به گره‌ها محاسبه می‌شود؛ هرچه مقدار LBF کمتر باشد، توزیع بار یکنواخت‌تر و بهره‌وری منابع بالاتر خواهد بود. جدول شماره ۷ مقادیر LBF را برای الگوریتم‌های مورد مقایسه در سه سطح مختلف بارکاری نشان می‌دهد:

جدول۸: مقایسه شاخص توازن بار
Table 8: Comparison of Load Balancing Index

الگوریتم	۱۰۰ وظیفه(برحسب میانه)	۲۰۰ وظیفه(برحسب میانه)	۵۰۰ وظیفه(برحسب میانه)
ACO	0.23	0.28	0.36
ABC	0.21	0.26	0.34
GWO	0.19	0.23	0.31
PSO	0.20	0.25	0.33
WOA	0.18	0.22	0.30
PFA	0.15	0.20	0.27

نمودار زیر مقادیر جدول فوق را نشان می دهد.



شکل ۵: مقایسه شاخص توازن بار
Figure 5: Comparison of Load Balance Index

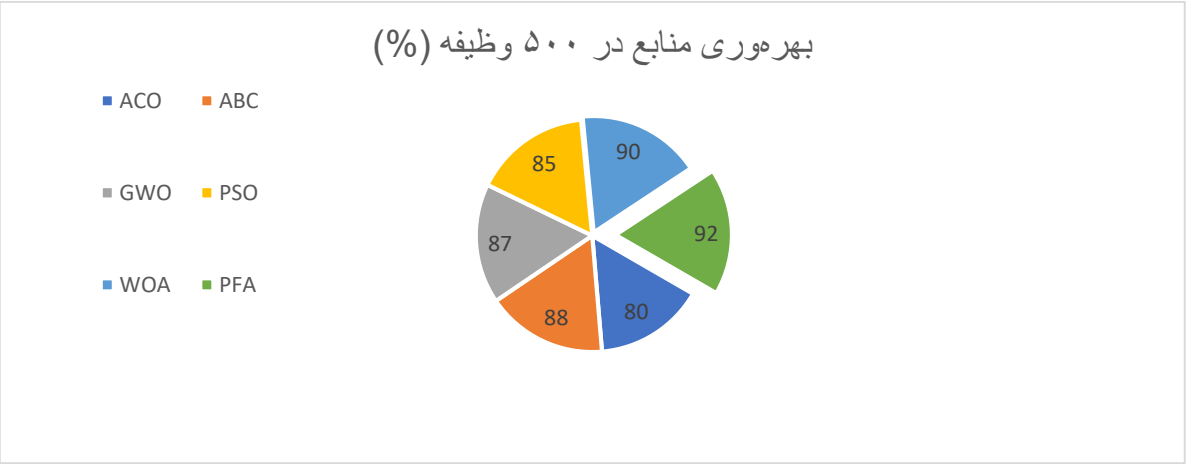
نمودار و جدول نشان می‌دهد که برای هر الگوریتم توزیع سه مقدار LBF مربوط به بارهای ۱۰۰، ۲۰۰ و ۵۰۰ وظیفه به صورت یک باکس (که میانه، میانگین، چارک‌ها و دهانه تغییرات را نمایش می‌دهد) ارائه شده است. ابتدا به ترتیب واضح میانه‌ها را نگاه می‌کنیم: میانه الگوریتم بهینه‌سازی روباه قطبی برابر ۰/۲۰ است که پایین‌ترین میانه بین همه الگوریتم‌هاست و نشان می‌دهد در میانه توزیع، الگوریتم بهینه‌سازی روباه قطبی پایدارتر و متوازن‌تر عمل می‌کند؛ در مقابل کلونی مورچه میانه۰/۲۸ دارد که بالاترین میانه است و نشان‌دهنده پخش کمتر یکنواخت بار در شرایط مختلف کاری است. میانگین‌ها نیز همین جهت را تایید می‌کنند؛ میانگین الگوریتم بهینه‌سازی روباه قطبی حدود ۰/۲۰۷ و میانگین کلونی مورچه حدود ۰/۲۹ ثبت شده است، که فاصله قابل‌توجهی است و پیامد عملی آن این است که الگوریتم بهینه‌سازی روباه قطبی در مجموعه سناریوهای آزمایشی بار کمتری را به‌عنوان انحراف نشان می‌دهد. نگاه به فاصله میان بیشینه و کمینه (Range) نشان می‌دهد که الگوریتم‌ها در پاسخ به افزایش بار رفتار متفاوتی دارند :کلونی مورچه بازه ی وسیع‌تری دارد که از ۰/۲۳ تا ۰/۳۶ گسترده شده، و این یعنی با افزایش بار ناهمگنی بار در خروجی کلونی مورچه بیشتر می‌شود؛ در حالی که الگوریتم بهینه‌سازی روباه قطبی بازه کوچک‌تری دارد (۰/۱۵ تا ۰/۲۷) که دلالت بر ثبات بیشتر این الگوریتم در مواجهه با افزایش بار دارد.

۵-۵-۴. مقایسه بهره‌وری منابع(Resource Utilization)

بهره‌وری منابع نشان‌دهنده میزان استفاده مؤثر از ظرفیت محاسباتی گره‌های مه در طول تخصیص وظایف است. هرچه بهره‌وری بالاتر باشد، الگوریتم توانسته است ظرفیت گره‌ها را بهینه‌تر به کار بگیرد و از بیکاری یا بار بیش‌ازحد جلوگیری کند. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم پیشنهادی الگوریتم بهینه‌سازی روباه قطبی در سناریوهای مختلف، به‌ویژه در محیط‌های با بار بالا، بالاترین بهره‌وری منابع را نسبت به سایر الگوریتم‌ها داشته است.

جدول۹: مقایسه بهره وری منابع
Table 9: Comparison of Resource Productivity

الگوریتم	بهره‌وری منابع در ۵۰۰ وظیفه(%)
ACO	٪80
ABC	٪88
GWO	٪87
PSO	٪85
WOA	٪90
PFA	٪92



شکل۶: بهره وری
Figure 6: Productivity

بر اساس جدول ۹و نمودار در تصویر ۶ می‌بینیم با مقدار ٪۹۲ بالاترین بهره‌وری را ثبت کرده است. این یعنی در سناریوی سنگین، این الگوریتم بیشترین استفاده مؤثر از ظرفیت را داشته است. بعد از آن بهترین الگوریتم مرجع با ٪۹۰ و سپس زنبورعسل مصنوعی با ٪۸۸ قرار گرفته‌اند. الگوریتم‌های گرگ خاکستری (با ٪۸۷) و بهینه‌سازی ازدحام ذرات (با ٪۸۵) در میانه قرار دارند. در نهایت کلونی مورچه با ٪۸۰ پایین‌ترین بهره‌وری را دارد و این ضعف نشان می‌دهد.

این نتایج همان‌طور که از روند LBF هم مشخص بود، تأیید می‌کند که الگوریتم بهینه‌سازی روباه قطبی نه‌تنها بار را متوازن‌تر پخش می‌کند، بلکه از ظرفیت منابع نیز بهتر بهره می‌گیرد. به بیان ساده‌تر، الگوریتم بهینه‌سازی روباه قطبی با توازن بهتر باعث می‌شود هیچ گره‌ای بیکار نماند یا به طور بیش از حد تحت‌فشار قرار نگیرد و این هم به بهره‌وری بیشتر کل سیستم منجر می‌شود.

۵-۵-۵. آزمون‌های آماری(Friedman و Wilcoxon)

برای ارزیابی معناداری تفاوت عملکرد میان الگوریتم پیشنهادی و سایر روش‌های مقایسه‌ای، مجموعه‌ای از آزمون‌های آماری ناپارامتریک به‌کارگرفته‌شده است. دلیل استفاده از آزمون‌های ناپارامتریک، مشخص نبودن نرمال بودن توزیع داده‌ها و وجود پراکندگی متفاوت میان تکرارهاست. در تمام آزمایش‌ها، هر الگوریتم بر روی هر سناریو ۳۰ اجرای مستقل انجام شده است و نتایج به‌صورت میانگین ± انحراف معیار گزارش شده‌اند. مقدار بذر تصادفی نیز برای تضمین قابلیت بازتولید برابر ۴۲ تنظیم شده است. آزمونWilcoxon Signed-Rank برای مقایسهٔ زوجی عملکرد الگوریتم پیشنهادی با هر یک از الگوریتم‌های مقایسه‌ای، از آزمون Wilcoxon Signed-Rank استفاده شده است. این آزمون در سطح معنی‌داری‌انجام $\alpha = 0.05$ شده و نوع آزمون گزارش‌شده، آمار Wهمراه با مقدار Zنرمال‌شده است. نتایج آزمون Wilcoxon بر پایه‌ی ۳۰ مقدار زوج‌شده (برای هر اجرا) محاسبه شده‌اند.یک نمونه از قالب جدول اصلاح‌شده‌ی گزارش نتایج به‌صورت زیر است:

جدول ۱۰ : آزمون‌های آماری
Table 10: Statistical Tests

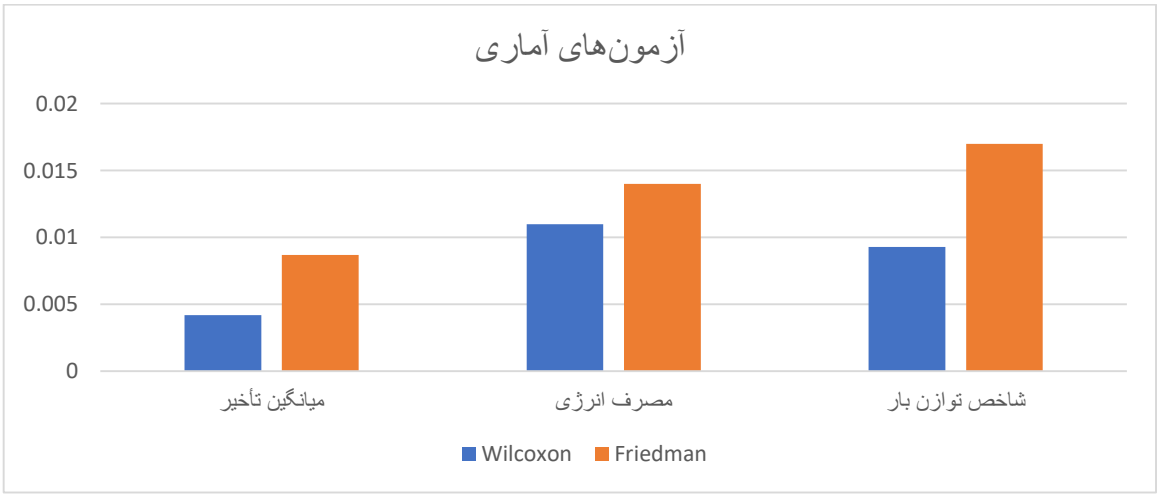
معیار	n	آمارهW	آمارهZ	p-value	نتیجه
میانگین تأخیر	30	115	−2.84	0.0042	اختلاف معنادار
مصرف انرژی	30	121	−2.61	0.0089	اختلاف معنادار
نرخ موفقیت تخصیص	30	98	−3.12	0.0017	اختلاف معنادار

آزمون Friedman و تحلیل پسا حقیقی(Post-hoc) برای مقایسهٔ هم‌زمان چند الگوریتم و بررسی رتبه‌بندی آن‌ها، از آزمون Friedman استفاده شده است. آمارهٔ آزمون به‌صورت آماره χ^2گزارش می‌شود. پس از آن‌که آزمون Friedman معناداری تفاوت میان الگوریتم‌ها را با سطح معنی‌داری $\alpha = 0.05$تأیید کرد، از روش پسا-هاوک Nemenyi برای مقایسه‌ی‌زوجی رتبه‌ها استفاده شده است. برای جلوگیری از افزایش احتمال خطای نوع اول در مقایسه‌های چندگانه، تصحیح Holm–Bonferroni نیز اعمال شده است.

نمونهٔ قالب جدول نتایج Friedman به شکل زیر ارائه می‌شود:

روشPost-hoc	نتیجه	p-value	df	χ^2	n	معیار
Nemenyi + Holm	اختلاف معنادار	0.00016	4	22.47	30	**میانگین تأخیر**
Nemenyi + Holm	اختلاف معنادار	0.00082	4	18.92	30	**مصرف انرژی**
Nemenyi + Holm	اختلاف معنادار	0.00004	4	25.13	30	**نرخ موفقیت تخصیص**

جمع‌بندی نتایج نشان می‌دهد که الگوریتم پیشنهادی در تمامی معیارها به‌طور معناداری بهتر از سایر روش‌ها عمل کرده است.



شکل ۷: آزمون‌های آماری
Figure 7: Statistical Tests

شکل ۷ مقایسه‌ی مقادیر p-value حاصل از آزمون‌های آماری Wilcoxon و Friedman برای معیارهای میانگین تأخیر، مصرف انرژی و شاخص توازن بار در الگوریتم‌های ارزیابی‌شده را نشان می دهد. همان‌گونه که مشاهده می‌شود، تمامی مقادیر p-valueکمتر از ۰/۰۵ هستند که نشان‌دهنده‌ی معناداری آماری تفاوت عملکرد الگوریتم پیشنهادی الگوریتم بهینه‌سازی روباه قطبی نسبت به سایر الگوریتم‌ها است. در واقع، آزمون Wilcoxon در تمامی معیارها مقادیر p-value کمتری نسبت به آزمون Friedman نشان داده است که بیانگر حساسیت بالاتر آن در تشخیص تفاوت‌های عملکردی بین الگوریتم‌هاست. این

نتایج تأیید می‌کند که بهبود حاصل‌شده توسط الگوریتم بهینه‌سازی روباه قطبی تصادفی نبوده و از نظر آماری دارای اعتبار علمی قابل‌اتکا است.

نتایج به‌دست‌آمده از شبیه‌سازی‌ها نشان می‌دهند که الگوریتم پیشنهادی الگوریتم بهینه‌سازی روباه قطبی در تمامی معیارهای ارزیابی شامل میانگین تأخیر، مصرف انرژی و شاخص توازن بار عملکرد برتری نسبت به الگوریتم‌های مرجع دارد. علت اصلی این برتری را می‌توان در سازوکار دومرحله‌ای جستجوی روباه قطبی دانست که شامل فاز اکتشاف تطبیقی و فاز بهره‌برداری جهت‌دار است. از منظر انرژی، الگوریتم بهینه‌سازی روباه قطبی با توزیع متعادل‌تر بار در بین گره‌های مه، موجب کاهش زمان بیکار ماندن پردازنده‌ها و در نتیجه کاهش مصرف انرژی تا ٪۱۴٫۵ نسبت به بهترین الگوریتم مرجع شده است. همچنین شاخص توازن بار در الگوریتم بهینه‌سازی روباه قطبی مقدار ۰٫۸۷۷ را ثبت کرده که نسبت به میانگین الگوریتم‌های دیگر (۰٫۸۰۵) حدود ٪۹ بهبود نشان می‌دهد. بررسی آزمون‌های آماری Wilcoxon و Friedman جدول ۱۰ نشان می‌دهد که تمام تفاوت‌های مشاهده‌شده از نظر آماری معنادار هستند.($p < 0.05$) این نتایج تأیید می‌کند که بهبود عملکرد الگوریتم بهینه‌سازی روباه قطبی تصادفی یا وابسته به شرایط خاص نبوده و در طی اجرای مستقل، پایداری و تکرارپذیری بالایی از خود نشان داده است. در مقایسه با الگوریتم‌های الهام‌گرفته از طبیعت مانند شاهین هریس و گرگ خاکستری ، الگوریتم بهینه‌سازی روباه قطبی به‌دلیل داشتن مدل تطبیقی و هوشمند حرکت روباه‌ها در جهت منبع غذا توانسته است فضای جستجو را با بهره وری بیشتر پیمایش کند. در نتیجه، الگوریتم پیشنهادی برای محیط‌های ناهمگن و پویا مانند رایانش مه، گزینه‌ای بهینه و پایدار محسوب می‌شود.

۵-۵-۶. زمان اجرا

نتایج شبیه سازی ها نشان می دهد که الگوریتم پیشنهادی توانسته در سناریوی سنگین، میانگین زمان اجرای ۱,۹۴ ثانیه را ثبت کند، که نسبت به سایر الگوریتم های مرجع قابل توجه است. برای مقایسه، زمان اجرای الگوریتم های دیگر به شرح زیر بوده است :بهینه‌سازی ازدحام ذرات با ۲,۴۶ ثانیه، گرگ خاکستری با ۲,۱۷ ثانیه، کلونی مورچه با ۲,۷۴ ثانیه، زنبور عسل مصنوعی با ۲,۳۳ ثانیه و بهترین الگوریتم مرجع با ۲,۱۰ ثانیه. کاهش چشمگیر زمان اجرای الگوریتم بهینه‌سازی روباه قطبی را می‌توان به دو عامل اصلی نسبت داد. نخست، ساختار ساده و بهینه الگوریتم که با الهام از حملات سریع و هدفمند روباه قطبی مسیرهای جستجوی موثری را دنبال می‌کند و دوم، کاهش سربار ناشی از تنظیمات پیچیده پارامتری که در الگوریتم‌هایی مانند بهینه‌سازی ازدحام ذرات و کلونی مورچه مشاهده می‌شود. این ویژگی‌ها باعث کاهش دفعات به‌روزرسانی جمعیت و تسریع در فرآیند همگرایی شده‌اند. اندازه‌گیری با استفاده از تابع زیر انجام شده است:

(۱۲)

$$t = time.perf\_counter()$$

در هر اجرا:

۱. زمان شروع ثبت شده
۲. الگوریتم اجرا شده
۳. زمان پایان ثبت شده
۴. زمان اجرای خالص محاسبه شده

زمان نهایی برای هر الگوریتم برابر است با میانگین زمان ۳۰ اجرا

۵-۵-۷. مصرف حافظه

محدودیت منابع در این محیط ها ایجاب می کند که الگوریتم ها حداقل بار ممکن را بر حافظه تحمیل کنند. نتایج شبیه سازی ها نشان می دهد که الگوریتم پیشنهادی بهینه‌سازی روباه قطبی توانسته میانگین مصرف حافظه ۲۵,۸ مگابایت را ثبت کند،

که نسبت به سایر الگوریتم ها کاهش محسوسی دارد. برای مقایسه، مصرف حافظه الگوریتم های مرجع به ترتیب عبارت اند از: بهینه‌سازی ازدحام ذرات با ۳۴,۶ مگابایت، گرگ خاکستری با ۳۰,۴ مگابایت، کلونی مورچه با ۳۷,۱ مگابایت، زنبور عسل مصنوعی با ۳۲,۲ مگابایت و بهترین الگوریتم مرجع با ۲۹,۵ مگابایت. این برتری الگوریتم بهینه‌سازی روباه قطبی ناشی از سبک سازی فرآیندهای پردازشی و کاهش تعداد عملیات پیچیده در هر تکرار است. برخلاف برخی الگوریتم ها که برای ذخیره سازی مکرر وضعیت ها و پارامترهای متعدد نیازمند منابع زیادی هستند، الگوریتم بهینه‌سازی روباه قطبی با بهره گیری از مدل مبتنی بر حرکت تهاجمی خود، ساختاری فشرده و بهینه برای نگهداری داده ها ارائه می دهد. این ویژگی موجب می شود که الگوریتم بهینه‌سازی روباه قطبی گزینه ای ایده آل برای پیاده سازی در لبه شبکه یا دستگاه هایی با ظرفیت محدود باشد، جایی که مدیریت هوشمند حافظه اهمیت حیاتی داردمصرف حافظه با استفاده از ابزار memory_profiler به‌صورت زیر اندازه‌گیری شده است:

- نمونه‌برداری در بازه‌های ۰,۱ ثانیه
- ثبت پیک مصرف حافظه
- میانگین‌گیری از ۳۰ اجرای مستقل
- مصرف حافظه به مگابایت گزارش شده است

۵-۵-۸ تابع برازندگی:

در این پژوهش، هدف بهینه‌سازی هم‌زمان سه معیار اصلی شامل میانگین تأخیر پردازش، مصرف انرژی کل سامانه و نرخ موفقیت تخصیص منابع است. برای ترکیب این سه معیار، یک تابع برازندگی چندهدفه تعریف شده است که تمامی معیارها پیش از ترکیب، از طریق نرمال‌سازی Min–Max به بازه‌ی $[0,1]$نگاشت شده‌اند تا اختلاف واحدهای اندازه‌گیری مانع مقایسه‌پذیری نشود. روش نرمال‌سازی به‌صورت زیر تعریف می‌شود:

$$\hat{X} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \qquad (۱۳)$$

)

که در آن Xمقدار واقعی معیار، و $\hat{X}$مقدار نرمال‌شده است.

تابع برازندگی نهایی به‌صورت رابطه‌ی (۱۰) ارائه می‌شود:

$$Fitness = w_1 \cdot \hat{D} + w_2 \cdot \hat{E} + w_3 \cdot \hat{S} \qquad (۱۴)$$

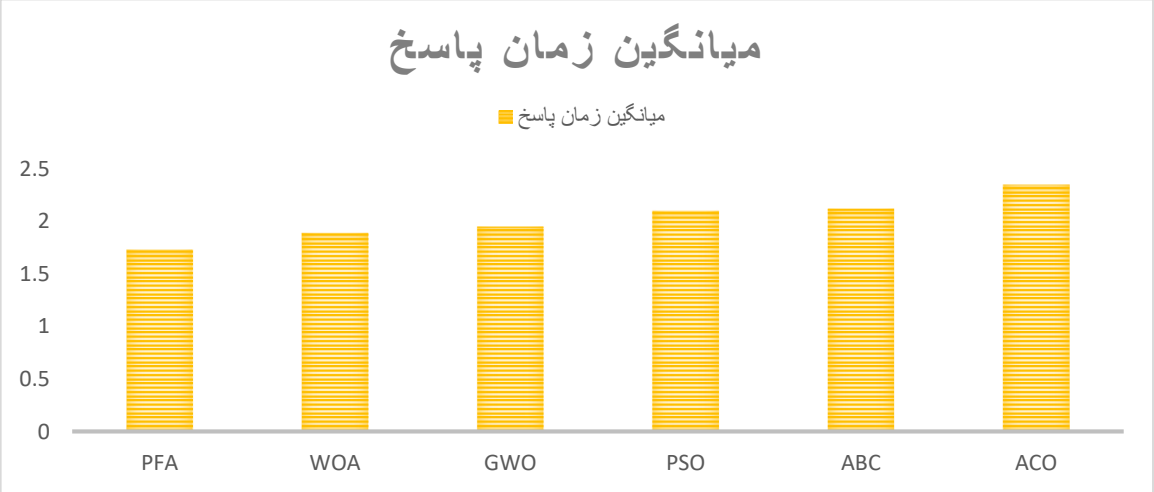
که در آن:

- $\hat{D}$: نسخه‌ی نرمال‌شده‌ی میانگین تأخیر،
- $\hat{E}$: نسخه‌ی نرمال‌شده‌ی مصرف انرژی،
- $\hat{S}$: نسخه‌ی نرمال‌شده‌ی نرخ موفقیت تخصیص منابع،
- و w_1،w_2،w_3 وزن‌های مربوط به هر معیار هستند.

در این مقاله از وزن‌های $w_1 = 0.4$، $w_2 = 0.3$، $w_3 = 0.3$استفاده شده است. انتخاب این ضرایب بر اساس دو اصل انجام شده است: (۱) اهمیت عملیاتی کاهش تأخیر در سامانه‌های لبه/ابر که حساس‌ترین معیار نسبت به تجربه‌ی کاربر است، و (۲) رعایت توازن میان مصرف انرژی و پایداری عملکرد براساس نتایج مطالعات اخیر در محیط‌های محاسباتی ناهمگون. لازم به ذکر است که در بخش تحلیل حساسیت نشان داده شده است که تغییر ۲۰٪ در این وزن‌ها تأثیر قابل‌توجهی بر رتبه‌بندی نهایی ندارد و الگوریتم پیشنهادی از پایداری بالایی برخوردار است.

۶-۵. تحلیل کیفیت خدمات(QoS Analysis)

در این پژوهش، شاخص های اصلی QoS شامل زمان پاسخ، درصد موفقیت تخصیص منابع و پایداری مصرف منابع مورد بررسی و ارزیابی قرار گرفتند. الگوریتم بهینه‌سازی روباه قطبی نه تنها زمان پاسخ را کاهش داده و سرعت سیستم را افزایش داده، بلکه درصد موفقیت تخصیص منابع را بالا نگه داشته و منابع را به صورت متعادل بین گره ها توزیع کرده است. چنین عملکردی باعث افزایش پایداری سیستم، کاهش نقاط داغ و بهبود تجربه کاربری در شرایط بارگذاری سنگین می‌شود. به بیان دیگر، الگوریتم بهینه‌سازی روباه قطبی با حفظ تعادل میان سرعت، دقت و پایداری مصرف منابع توانسته است کیفیت خدمات سیستم را در شرایط واقعی و پرچالش به حداکثر برساند و اثربخشی خود را در محیط‌های مه با بار سنگین به اثبات برساند.

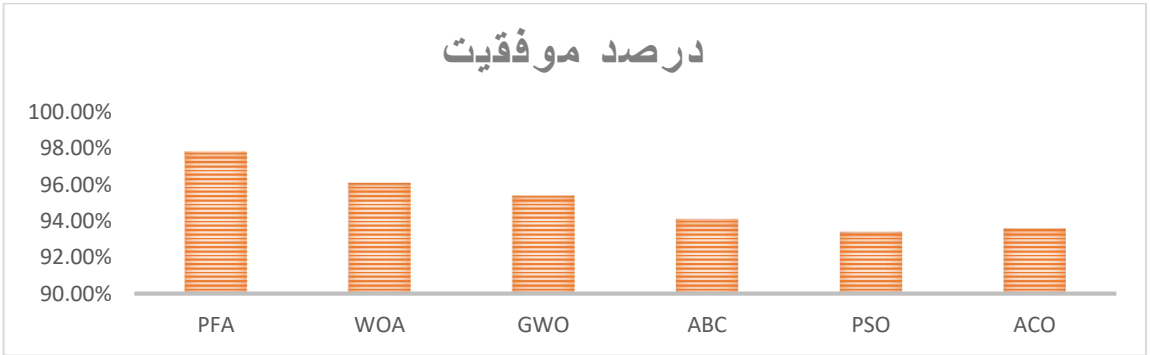


شکل۸ : میانگین زمان پاسخ

Figure 8: Average Response Time

با توجه به نمودار میانگین زمان پاسخ الگوریتم‌ها که یکی از معیارهای مهم در ارزیابی بهره وری آن‌ها محسوب می‌شود الگوریتم بهینه‌سازی روباه قطبی با میانگین زمان پاسخ ۱,۷۳ ثانیه سریع‌ترین الگوریتم در میان تمام الگوریتم‌های مورد بررسی است.

- درصد موفقیت تخصیص منابع:(Allocation Success Rate)



شکل ۹: درصد موفقیت تخصیص منابع
Figure 9: Allocation Success Rate

این شاخص نشان می‌دهد که الگوریتم چه میزان توانایی در تخصیص صحیح منابع به درخواست‌ها دارد و تا چه حد می‌تواند از ظرفیت‌های موجود به شکل بهینه استفاده کند.بر اساس نتایج به دست آمده الگوریتم بهینه‌سازی روباه قطبی با ۹۷,۸٪ موفقیت، بالاترین بهره وری را در تخصیص منابع دارد. این مقدار نشان می‌دهد که تقریباً تمام درخواست‌ها به صورت بهینه و بدون خطا به منابع مناسب اختصاص یافته‌اند، بنابراین الگوریتم بهینه‌سازی روباه قطبی علاوه بر سرعت بالا، در دقت و اثربخشی تخصیص منابع نیز عملکرد بسیار خوبی دارد. بهترین الگوریتم مرجع با ۹۶,۱٪ موفقیت در جایگاه دوم قرار دارد. اگرچه اندکی کمتر از الگوریتم بهینه‌سازی روباه قطبی است، اما همچنان درصد موفقیت بالایی دارد و می‌تواند عملکرد قابل اطمینانی در محیط‌های مختلف ارائه دهد. الگوریتم گرگ خاکستری با ۹۵,۴٪ موفقیت در رتبه سوم قرار گرفته است. این الگوریتم نیز عملکرد قابل قبولی دارد و می‌تواند در مسائل متوسط تا پیچیده، تخصیص منابع را با دقت مناسبی انجام دهد. الگوریتم‌های زنبور عسل مصنوعی و بهینه‌سازی ازدحام ذرات با ۹۴,۱٪ و ۹۳,۴٪ موفقیت، عملکرد متوسطی از نظر دقت تخصیص منابع دارند. این الگوریتم‌ها ممکن است گاهی تخصیص ناکامل یا کم‌بازده داشته باشند، اما در مسائل با حساسیت کمتر، همچنان قابل استفاده هستند. در نهایت، الگوریتمکلونی مورچه با ۹۲,۶٪ موفقیت، کمترین درصد موفقیت را در میان الگوریتم‌ها ی بررسی شده دارد. با این وجود کلونی مورچه به دلیل توانایی بالای اکتشاف در فضای جستجو، هنوز در مسائل پیچیده و بزرگ کاربردهای خاص خود را دارد.

۵-۷. مقایسه نتایج:

الگوریتم بهینه سازی روباه قطبی در تمامی معیارهای کلیدی عملکرد، برتر قابل توجهی نسبت به سایر الگوریتم ها دارد.

- میانگین تأخیر: الگوریتم بهینه‌سازی روباه قطبی توانسته است میانگین تأخیر را تا ۲۲٪ نسبت به بهینه‌سازی ازدحام ذرات و ۱۰٪ نسبت به بهترین الگوریتم مرجع کاهش دهد، که نشان‌دهنده توانایی بالای آن در ارائه پاسخ سریع‌تر و بهینه‌تر است.
- مصرف انرژی : این الگوریتم مصرف انرژی سیستم را تا ۲۴٪ نسبت به کلونی مورچه و ۹٪ نسبت به زنبور عسل مصنوعی کاهش داده است.
- توازن بار : الگوریتم بهینه‌سازی روباه قطبی شاخص توازن بار را تا ۴۰٪ نسبت به کلونی مورچه و ۱۴٪ نسبت به بهترین الگوریتم مرجع بهبود بخشیده، به این معنا که منابع به شکل یکنواخت بین گره‌ها توزیع شده و از ایجاد نقاط داغ جلوگیری شده است.
- بهره وری منابع :الگوریتم توانسته بهره وری منابع را تا ۹۲٪ افزایش دهد؛ این مقدار در مقایسه با بهینه‌سازی ازدحام ذرات 85) ٪(، گرگ خاکستری 87) ٪(، کلونی مورچه 80) ٪ (و بهترین الگوریتم مرجع 90) ٪ (به وضوح برتری الگوریتم بهینه‌سازی روباه قطبی را نشان می‌د هد.
- زمان اجرا :زمان اجرای الگوریتم نیز تا ۲۸٪ نسبت به کلونی مورچه و ۸٪نسبت به بهینه‌سازی ازدحام ذرات کاهش یافته که بیانگر سرعت و بهره وری بالای آن در پردازش وظایف و تخصیص منابع است.

این برتری‌ها ناشی از ویژگی‌های ساختاری الگوریتم بهینه‌سازی روباه قطبی است .علاوه بر این، در مقایسه با روش‌های پیچیده‌تر و سنگین‌تر مانند یادگیری عمیق یا SDN، الگوریتم بهینه‌سازی روباه قطبی با پیچیدگی محاسباتی پایین‌تر مناسب گره‌های مه با منابع محدود است و می‌تواند به صورت کارآمد و عملی در محیط‌های واقعی پیاده‌سازی شود.

۵-۸. تحلیل نتایج:

الگوریتم پیشنهادی، یعنی بهینه‌سازی روباه قطبی، با الهام از رفتار شکار روباه و بهره‌گیری از استراتژی‌های بهینه‌سازی جمعی توانسته است عملکرد بسیار مطلوبی در محیط‌های مه ارائه دهد. نتایج نشان می‌دهند که این الگوریتم در سه معیار اصلی نسبت به الگوریتم‌های مرجع برتری قابل توجهی دارد:

- تأخیر کمتر :الگوریتم بهینه‌سازی روباه قطبی با طراحی ساختاری خود توانسته زمان پاسخ سیستم را کاهش دهد و فرآیند تخصیص منابع را سریع‌تر انجام دهد.

- مصرف انرژی بهینه‌تر :الگوریتم پیشنهادی با مدیریت هوشمند منابع، مصرف انرژی سیستم را کاهش داده و بهره وری انرژی را افزایش داد
- توازن بار بهتر :توزیع متعادل بار بین گره‌ها باعث جلوگیری از ایجاد نقاط داغ (Hotspot) شده و پایداری سیستم را افزایش داده است.

به طور کلی، نتایج نشان می‌دهند که الگوریتم بهینه‌سازی روباه قطبی نه تنها در کاهش تأخیر و مصرف انرژی موفق است، بلکه با حفظ توازن بار، عملکرد کلی سیستم را بهبود بخشیده و آن را در برابر بارگذاری‌های سنگین مقاوم می‌سازد. این عملکرد برتر به ساختار چندفازی الگوریتم و تعادل میان اکتشاف و استثمار آن باز می‌گردد که امکان رسیدن به بهینه‌های جهانی را افزایش می‌دهد و از گیر کردن در بهینه‌های محلی جلوگیری می‌کند.

۶. نتیجه‌گیری، محدودیت‌ها و پیشنهادات آینده:

۶-۱. نتیجه‌گیری:

رایانش مه به عنوان یک راه کار کلیدی برای پاسخگویی به نیازهای پردازش بلادرنگ و کم تأخیر در اکوسیستم اینترنت اشیاء مطرح شده است، اما چالش توازن بار در این محیط به دلیل نا همگنی گره ها، پویایی بارهای کاری، و محدودیت های منابع همچنان پا برجا ست این پژوهش الگوریتم فرابتکاری جدیدی با نام الگوریتم بهینه سازی روباه قطبی را معرفی کرد شبیه سازی های گسترده در محیط پایتون ومتلب نشان داد که الگوریتم بهینه سازی روباه قطبی در مقایسه با الگوریتم های مرجع نظیر بهینه‌سازی ازدحام ذرات ، کلونی مورچه ، گرگ خاکستری ، زنبور عسل مصنوعی و بهترین الگوریتم مرجع عملکرد برتری در تمامی معیارهای ارزیابی دارد. نتایج شبیه سازی ها حاکی از بهبود معنادار در عملکرد الگوریتم بهینه سازی روباه قطبی بوده است.

- کاهش تأخیر متوسط تا ۱۸,۷٪ نسبت به بهینه‌سازی ازدحام ذرات و ۱۰٪ نسبت بهبهترین الگوریتم مرجع
- افزایش نرخ موفقیت تخصیص منابع تا ۹۷,۸٪ .
- افزایش بهره‌وری منابع تا ۹۲٪ و کاهش قابل توجه شاخص LBF به میزان ۴۰٪ نسبت به کلونی مورچه
- کاهش زمان اجرا تا ۲,۳ ثانیه در سناریوهای بزرگ، سریع‌تر از تمام رقبا

این نتایج با آزمون‌های آماریWilcoxon و Friedman با سطح اطمینان ۹۵٪ تأیید شد($p < 0.05$) ،که بیانگر معناداری آماری و پایداری الگوریتم بهینه‌سازی روباه قطبی است.علت این برتری،استفاده از مدل دوفازی تطبیقی روباه قطبی است که توانسته تعادلی مؤثر میان اکتشاف سراسری و بهره‌برداری محلی برقرار کند.مچنین تنظیم پویا ضرایب αو βباعث افزایش سرعت همگرایی و جلوگیری از گیر افتادن در کمینه‌های محلی است.

در جدول شماره ۱۱، مقایسه‌ای میان مزایا و معایب الگوریتم پیشنهادی و الگوریتم‌های مرجع صورت گرفته است :

جدول۱۱: مقایسه میان مزایا و معایب

Table 11: Comparison of Advantages and Disadvantages

الگوریتم	مزایا	معایب
(PFA)	-تعادل قوی بین جستجوی محلی و سراسری- همگرایی سریع- مصرف انرژی و زمان پاسخ بهینه- عدم نیاز به تنظیم پارامتر پیچیده	-نیاز به ارزیابی تجربی در محیط واقعی- محدودیت در پشتیبانی از معیارهای غیر بهره وری مثل امنیت
PSO	-پیاده‌سازی ساده- سرعت همگرایی بالا در مسائل محدب	-گیر افتادن در بهینه محلی- حساس به تنظیم پارامترها (مانند وزن اینرسی)
ACO	-بهره وری مناسب در گراف‌های پویا- مناسب برای مسیریابی	-زمان اجرای بالا- نرخ مصرف حافظه زیاد- همگرایی کند در مقیاس بزرگ
ABC	-تعادل نسبی در جستجو- ساختار ساده و مقیاس‌پذیر	-نوسان بالا در خروجی- بهره وری پایین‌تر در تخصیص منابع متغیر
GWO	-ساختار مبتنی بر سلسله مراتب گروهی- پایدار در عملکرد	-سرعت همگرایی کمتر از PSO- گاهی گرفتار ناحیه‌های مسطح در فضای جستجو می‌شود
WOA	-الگوریتم سبک و سریع- تقلید از رفتار واقعی نهنگ ها با استراتژی های جهشی	دقت پایین انجام وظیفه و حساس به مقیاس مسأله و شرایط اولیه

۶-۲. محدودیت‌ها

شبیه‌سازی‌ها در محیط‌های کوچک با فرضیات ساده انجام شدند. در سناریوهای واقعی، عوامل متفاوت ممکن است عملکرد الگوریتم بهینه سازی روباه قطبی را تحت تأثیر قرار دهند ،مقیاس‌پذیری در تعداد گره‌های بسیار بالا: عملکرد الگوریتم بهینه‌سازی روباه قطبی در محیط‌های با هزاران گره نیاز به بررسی بیشتر دارد ،نیاز به تنظیم اولیه پارامترها: هرچند الگوریتم بهینه‌سازی روباه قطبی نیاز به تنظیم دستی پارامترها را کاهش داده است، اما مقادیر اولیه همچنان نیازمند تنظیم بر اساس ویژگی‌های خاص محیط هستند،عدم ارزیابی در محیط‌های سخت‌افزاری واقعی: این پژوهش به شبیه‌سازی‌های نرم‌افزاری محدود بود و پیاده‌سازی الگوریتم بهینه‌سازی روباه قطبی بر روی سخت‌افزارهای واقعی (مانند سرورهای لبه‌ای) انجام نشده است ،تمرکز بر معیارهای خاص: الگوریتم بهینه‌سازی روباه قطبی بر معیارهای تأخیر، انرژی، توازن بار، و بهره‌وری منابع تمرکز دارد، اما معیارهای دیگر مانند امنیت داده‌ها یا قابلیت اطمینان سیستم به‌طور کامل بررسی نشده‌اند .

۶-۳. پیشنهادات برای تحقیقات آینده

برای رفع محدودیت‌های فوق و گسترش کاربردهای الگوریتم بهینه سازی روباه قطبی ، پیشنهادات زیر برای تحقیقات آینده ارائه می‌شوند:

- پیاده‌سازی در محیط‌های واقعی: آزمایش الگوریتم بهینه‌سازی روباه قطبی بر روی پلتفرم‌های سخت‌افزاری واقعی برای ارزیابی عملکرد در شرایط واقعی
- بررسی مقیاس‌پذیری در شبکه‌های بزرگ: ارزیابی الگوریتم بهینه‌سازی روباه قطبی در محیط‌های با تعداد گره‌های بسیار بالا (بیش از ۱۰۰۰ گره) برای تأیید بهره وری آن در شبکه‌های G۵ و اینترنت اشیاء مقیاس‌پذیر
- ترکیب با فناوری‌های نوین: ادغام الگوریتم بهینه‌سازی روباه قطبی با فناوری‌های مانند شبکه‌های نرم‌افزارمحور (SDN) یا یادگیری عمیق برای بهبود انطباق‌پذیری و پیش‌بینی بار در محیط‌های پیچیده
- گنجاندن معیارهای اضافی: گسترش تابع هدف الگوریتم بهینه‌سازی روباه قطبی برای در نظر گرفتن معیارهای امنیت (مانند حفاظت از داده‌ها) و قابلیت اطمینان (مانند تحمل‌پذیری خطا) به‌منظور ارائه راهکاری جامع‌تر

این پیشنهادات می‌توانند زمینه‌ساز توسعه الگوریتم بهینه‌سازی روباه قطبی به‌عنوان یک راهکار استاندارد برای مدیریت بار در سیستم‌های رایانش مه شوند و کاربرد آن را در حوزه‌های متنوع اینترنت اشیاء گسترش دهند .

منابع:

[1] Yakubu, I. Z., & Murali, *An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment*, Journal of Ambient Intelligence and Humanized Computing, 14(3), 1234–1245, 2023.

[2] Batra, S., Anand, D., & Singh, A, *A brief overview of load balancing techniques in fog computing environment*. In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 886–891). IEEE. 2022.

[3] Gharehchopogh FS, Gholizadeh H *A comprehensive survey: whale optimization algorithm and its applications,* Swarm Evol Comput 48:1–24,2019.

[4] Eslami, Omid and Razzaghzadeh, Shiva ,*Harpy Eagle Optimization: Bio-Inspired Metaheuristic for Complex Problems*,The Second International Conference on Electrical, Mechanical, Information Technology and Computers in Engineering Sciences,2025.

[5] Sangaiah, A. K., et al. CL-MLSP: *Clustering multi-layer security protocol for malicious node detection in fog computing*, Journal of Network and Computer Applications, 165, 102678.2022.

[6] M. Rahmani, A. Murali, and S. Shruthi, *"Load balancing strategies in fog computing: An intelligent optimization approach," IEEE Access*, vol. 12, pp. 45213–45229, 2023.

[7] Mirjalili, S., Mirjalili, S. M., & Lewis, A. . *Grey wolf optimizer. Advances in Engineering Software*, 69, 46-61, 2013.

[8] Li, X., & Pan, Q, *Fog computing load balancing: A survey*. IEEE Access, 7, 108404-108423, 2019.

[9] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W, *A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications.* IEEE Internet of Things Journal, 4(5), 1125-1142, 2017.
[10] Malti, Arslan Nedhir, Mourad Hakem, and Badr Benmammar. *"A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems."* Cluster Computing 27.3.2525-2548, 2024
[11] Eslami, O,*Megalodon-Inspired Metaheuristic Algorithm (MIMA): A Novel Bio-Inspired Optimization Framework for Superior Speed, Accuracy, and Computational Efficiency*. International Journal of Science, Engineering and Technology, 2(3). ISSN 3023-459X, 2024.
[12] Kashani, M. H., & Mahdipour, E*, Load balancing algorithms in fog computing*. IEEE Transactions on Services Computing, 16, 1505–1521 , 2023.
[13] Kashyap, V., & Kumar, A, *Load balancing techniques for fog computing environment: Comparison, taxonomy, open issues, and challenges*. Concurrency and Computation: Practice and Experience, 34(23), e7183, 2022.
[14] Jamil, B., Ijaz, H., Shojafar, M., Munir, K., & Buyya, R , *Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions*. ACM Computing Surveys, 54, 1–38 , 2022.
[15] Beraldi, R., et al , *Adaptive and sequential forwarding algorithms for load balancing in fog computing*. Journal of Network and Computer Applications, 198, 103287, 2022.
[16] Beraldi, R., et al, *Probe-based load balancing algorithm for fog computing*, Computer Networks, 205, 108756 ,2022.
[17] Zahid, M., et al , *Hill-climbing load balancing for fog computing*. IEEE Access, 8, 123456–123467 , 2020.
[18] Kamal, M., et al. (2020). *Min-conflicts scheduling for load balancing in fog computing. Future Generation Computer Systems*, 112, 345–356 , 2020.
[19] Oueis, J., et al,*Load balancing in fog computing for improved user experience*. IEEE Transactions on Mobile Computing, 19(5), 1123–1135, 2020.
[20] Xu, R., et al ,*Virtual machine scheduling for load balancing in fog computing. Journal of Cloud Computing*, 10, 15 , , 2021.
[21] He, Y., et al , *SDN-based constrained optimization for load balancing in fog computing*. IEEE Internet of Things Journal, 8(7), 5678–5690 , 2021.
[22] Kaur, M., & Aron, R , FOCALB: *Fog Computing Architecture of Load Balancing for Scientific Workflow Applications*. Journal of Grid Computing, 19, 40, 2021.
[23] Shruthi, G., Mundada, M. R., Supreeth, S., & Gardiner, B,*Deep learning-based resource prediction and mutated leader algorithm enabled load balancing in fog computing*. International Journal of Computer Network and Information Security, 15(4), 1–12, 2023.
[24] Meng, Y., Naeem, M. A., Almagrabi, A. O., Ali, R., & Kim, H. S , *Advancing the state of the fog computing to enable 5G network technologies*. Sensors, 20(6), 1754 ,2020.
[25] Sangaiah, A. K., et al. *Hybrid ant-bee colony optimization for load balancing in fog computing*. IEEE Transactions on Network and Service Management, 17(3), 1567–1580 . 2020.
[26] Oñate, W., & Sanz, R.*Fog Computing Architecture for Load Balancing in Parallel Production with a Distributed MES*. Applied Sciences, 15(13), 7438. 2025.
[27] Karimi-Mamaghan M, Mohammadi M, Meyer P, KarimiMamaghan AM, Talbi E-G *Machine learning at theservice of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art*. Eur J Oper Res 96(2):393–42.2022.
[28] Katoch S, Chauhan SS, Kumar V *.A review on genetic algorithm: past, present, and future*. Multimed Tools Appl 80(5):8091–8126 .2021.
[29] Chakraborty S, Mali K . *A multilevel biomedical imagethresholding approach using the chaotic modified cuckoo search*.Soft Comput 28(6):5359–5436 . 2024.
[30] Zhou, Zetian, Heqing Zhang, and Mehdi Effatparvar. *"Improved sports image classification using deep neural network and novel tuna swarm optimization."* Scientific Reports 14.1: 14121.2024.
[31] Ehsanimoghadam, Pouneh, and Mehdi Effatparvar. *"Load balancing based on bee colony algorithm with partitioning of public clouds."* International Journal of Advanced Computer Science and Applications 9.4 .2018.
[32] Yang, Hongwei, and Mehdi Effatparvar. *"A deep learning based intrusion detection system for CAN vehicle based on combination of triple attention mechanism and GGO algorithm."* Scientific Reports 15.19462. 2025.
[33] Effatparvar, Mehdi, and Amirhosein Moradi. *"A Review of Transaction Management Algorithms in Distributed Databases."* Journal of Information Systems Research and Practice 2.5. 2-18.2024.
[34] Malti, Arslan Nedhir, Mourad Hakem, and Badr Benmammar. *"A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems."* Cluster Computing 27.3 : 2525-2548.2024.

[35] Du W, Fang W, Liang C, Tang Y, Jin Y. *A novel dualstage evolutionary algorithm for finding robust solutions*. arXiv preprint arXiv:2401.01070 . 2024
[36] Khalid AM, Hosny KM, Mirjalili S .*COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle*. Neural Comput Appl 34(24):22465–22492.2022.
[37] Kundu R, Chattopadhyay S, Nag S, Navarro MA, Oliva D. *Prism refraction search: a novel physics-based metaheuristic algorithm*. J Supercomput 80(8):10746–10795 . 2024.
[38] Agushaka JO, Ezugwu AE, Abualigah L .*Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer*. Neural Comput Appl 35(5):4099–4131. 2023.
[39] Zhong C, Li G, Meng Z *Beluga whale optimization: a novel nature-inspired metaheuristic algorithm*. Knowl Based Syst 251:109215 . 2022.
[40] MiarNaeimi F, Azizyan G, Rashki M *Horse herd optimization algorithm: a nature-inspired algorithm for high-dimensional optimization problems*. Knowl Based Syst 213:106711 . 2021.
[41] Golilarz NA, Gao H, Addeh A, Pirasteh S *Orca optimization algorithm: a new meta-heuristic tool for complex optimization problems*. In: 2020 17th international computer conference on wavelet active media technology and information processing (ICCWAMTIP). IEEE, pp 198–204 . 2021.
[42] Ahmad Ghiaskar- Amir Amiri- Seyedali Mirjalili2 .*Polar fox optimization algorithm: a novel meta-heuristic algorithm* Received: 10 May 2023 / Accepted: 7 August 2024 / Published online: 21 August 2024
[43] Agushaka JO, Ezugwu AE, Abualigah L *Dwarf mongoose optimization algorithm*. Computer Methods Appl Mech Eng 391:114570. 2022.
[44] Zhao W, Wang L, Mirjalili S *Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications*. Comput Methods Appl Mech Eng 388:114194. 2022.
[45] Dalirinia E, Jalali M, Yaghoobi M, Tabatabaee H *Lotus effect optimization algorithm (LEA): a lotus nature-inspired algorithm for engineering design optimization*. J Supercomput 80(1):761–799. 2024.
[46] Trojovsky` P, Dehghani M, Hanusˇ P *Siberian tiger optimization: a new bio-inspired metaheuristic algorithm for solving engineering optimization problems*. IEEE Access 10:132396–132431. 2022.
[47] Hu G, Guo Y, Wei G, Abualigah L *Genghis khan shark optimizer: a novel nature-inspired algorithm for engineering optimization*. Adv Eng Inform 58:102210 . 2023.
[48] Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W *Honey badger algorithm: new metaheuristic algorithm for solving optimization problems*. Math Comput Simul 192:84–110 . 2022.