Journal of Electronical & Cyber Defence
Vol. 9, No. 2, 2021, Serial No. 34

Security-aware Task Scheduling Algorithm based on Multi Adaptive Learning and PSO Techniques

N. Mansouri*, B. M. Hasani Zade, R. Ghafari

*Shahid Bahonar University of Kerman

(Received: 03/10/2020, Accepted: 11/01/2021)

ABSTRACT

Today, many scientific problems require high computational power and storage space. Cloud computing is a model for easy access to different resources such as storage space with minimal need for service provider interaction. Hence the cloud environment has become a welcoming solution because of its numerous advantages, but due to outsourcing, security and privacy issues are critical challenges. On the other hand, task scheduling is another fundamental issue in distributed systems that use cloud computing, because there are several tasks to be performed that require different resources while resources are limited. Therefore, cloud tasks must be intelligently scheduled so that system performance and provider profitability are maximized. To solve this challenge, various techniques such as gradient-based algorithms for continuous and single-model problems are common. In cloud computing, due to the large search space and the complex nature, these algorithms may not provide a suitable solution. Efficient meta-heuristic techniques can deal with these problems and find near-optimal solutions in a reasonable time. In this paper, a security-based scheduling algorithm using an improved Particle Swarm Optimization algorithm is presented. The improved algorithm uses multi adaptive learning to provide diversity in a population and thus provides a good balance between exploration and exploitation. The proposed task scheduling algorithm simultaneously considers five parameters (i.e., round trip time, load, energy consumption, cost, and security) to provide load balancing and reduce energy consumption. The proposed algorithm is implemented using the CloudSim simulator and compared with the relevant strategies (i.e., CJS, OTSS, GTSA, and JSSS). The simulation results show that considering the characteristics of tasks and resources, the proposed algorithm has significant efficiency and effectiveness in the cloud environment, especially at high workloads.

Keywords: Cloud computing, Meta-heuristic algorithms, Task scheduling, Security

* Corresponding Author Email: najme.mansouri@gmail.com

نشریه علمی "پدافند الکترونیکی و سایبری"
سال نهم، شماره ۲، تابستان ۱۴۰۰، ص ۱۷۸-۱۵۹

علمی – پژوهشی

الگوریتم زمان‌بندی کار مبتنی بر بهینه‌سازی ازدحام ذرات و یادگیری انطباقی چندگانه برای بهبود امنیت در محیط رایانش ابری

نجمه منصوری[1*]، بهنام محمدحسنی‌زاده[2]، ریحانه غفاری[3]

۱- استادیار، ۲- دکتری تخصصی، ۳- کارشناسی ارشد، دانشگاه شهید باهنر کرمان، کرمان، ایران
(دریافت: ۱۳۹۹/۰۷/۱۲، پذیرش: ۱۳۹۹/۱۰/۲۲)

چکیده

امروزه بسیاری از مسائل علمی پیچیده نیاز به قدرت محاسباتی و فضای ذخیره‌سازی بالایی دارند. رایانش ابری مدلی است برای دسترسی آسان و بنا به سفارشِ منابع رایانشی مانند فضای ذخیره‌سازی با کمترین نیاز به دخالت فراهم‌کننده خدمات. ابرها به‌دلیل مزایای بسیار مورد استقبال قرار گرفتند ولی با توجه به برون‌سپاری، مسائل مربوط به امنیت و حفظ حریم خصوصی به‌عنوان مهم‌ترین مشکلات این حوزه مطرح می‌شوند. از طرف دیگر، زمان‌بندی کارها یک مسئله اساسی در سامانه‌های توزیع‌شده‌ای چون رایانش ابری است. زیرا در یک زمان واحد، کارهای متعددی برای اجرا شدن وجود دارد که به منابع متفاوتی احتیاج دارند درحالی‌که منابع محدود هستند. از این‌رو باید به‌طور هوشمندانه کارها زمان‌بندی شوند تا عملکرد سیستم و سوددهی فراهم‌کننده حداکثر گردد. برای حل این مشکل، روش‌های مختلف مانند الگوریتم‌های مبتنی بر گرادیان برای مسائل مستمر و تک مدلی معمول هستند. اما اگر برای زمان‌بندی در رایانش ابری استفاده شوند، به‌دلیل فضای جستجوی بزرگ و طبیعت پیچیده مسائل، این الگوریتم‌ها ممکن است راه‌حل رضایت‌بخشی ارائه ندهند. روش‌های فرااکتشافی کارآمد می‌توانند با این مشکل مقابله کرده و راه‌حل نزدیک به بهینه در کوتاه‌ترین دوره زمانی را پیدا کنند. در نتیجه در این مقاله، الگوریتم زمان‌بندی برای بهبود امنیت با استفاده الگوریتم بهینه‌سازی ازدحام ذرات بهبودیافته ارائه شده است. الگوریتم بهبودیافته با استفاده از یادگیری انطباقی منجر به تنوع در جمعیت می‌شود و لذا تعادلی بین عملیات اکتشاف و بهره‌برداری به‌دست می‌آید. الگوریتم زمان‌بندی پیشنهادی هم‌زمان پنج پارامتر (زمان بازگشت، بار، مصرف انرژی، هزینه و امنیت) را در حین توزیع کارها در نظر می‌گیرد تا در نهایت منجر به توزیع بار و کاهش مصرف انرژی می‌گردد. الگوریتم پیشنهادی با استفاده از شبیه‌ساز کلودسیم پیاده‌سازی و با روش‌های مربوطه (CJS, OTSS, GTSA, JSSS) مقایسه می‌شود. نتایج حاصل از شبیه‌سازی نشان می‌دهد که الگوریتم پیشنهادی با در نظر گرفتن ویژگی‌های کارها و منابع، کارایی و اثربخشی قابل‌توجهی در محیط رایانش ابری خصوصاً در بار کاری بالا دارد.

کلیدواژه‌ها: رایانش ابری، الگوریتم‌های فرااکتشافی، زمان‌بندی کار، امنیت

۱- مقدمه

رایانش ابری، فناوری درحال‌توسعه در حوزه محاسبات توزیع‌شده و دامنه پردازش موازی است. محبوبیت رایانش ابری به‌دلیل ویژگی‌های منحصربه‌فرد آن مانند خدمات متنوع، امنیت، قابلیت ارتجاعی و مقیاس‌پذیری روزبه‌روز در حال افزایش است [۱]. ارائه‌دهندگان خدمات ابری خدماتی مانند نرم‌افزار، فضای ذخیره‌سازی، خدمات شبکه و غیره را به مشتریان خود ارائه می‌دهند. برای اینکه ارائه‌دهندگان خدمات ابری بتوانند چنین خدماتی را به مشتریان خود ارائه دهند، باید از همه منابع ابری به بهترین حالت ممکن استفاده کنند. این استفاده به‌طور مؤثری توسط الگوریتم‌های زمان‌بندی کار انجام می‌شود. از جمله اهداف مهم زمان‌بندی کار می‌توان به افزایش عملکرد، کیفیت خدمات و همچنین کاهش هزینه اشاره کرد [۲]. امنیت در زمان‌بندی در محیط‌های توزیع‌شده یکی از مهم‌ترین معیارها در خدمات ابری است. بسیاری از ویژگی‌هایی که رایانش ابری را جذاب می‌کنند، می‌توانند در مدل‌ها و کنترل‌های امنیتی سنتی مشکل‌ساز شوند. مشکلات امنیتی گاهی اوقات رایانش ابری را دشوار می‌کنند، خصوصاً برای آن دسته از ارائه‌دهندگان خدمات ابری که زیرساخت‌ها و منابع محاسباتی آن‌ها متعلق به یک طرف خارج از حوزه خودشان است که خدمات را به عموم ارائه می‌دهند.

*رایانامه نویسنده مسئول: najme.mansouri@gmail.com

ناشر: دانشگاه جامع امام حسین (ع)

خدمات امنیتی در محیط رایانش ابری در مقایسه با خدمات داده سنتی، بسیار پیچیده است.

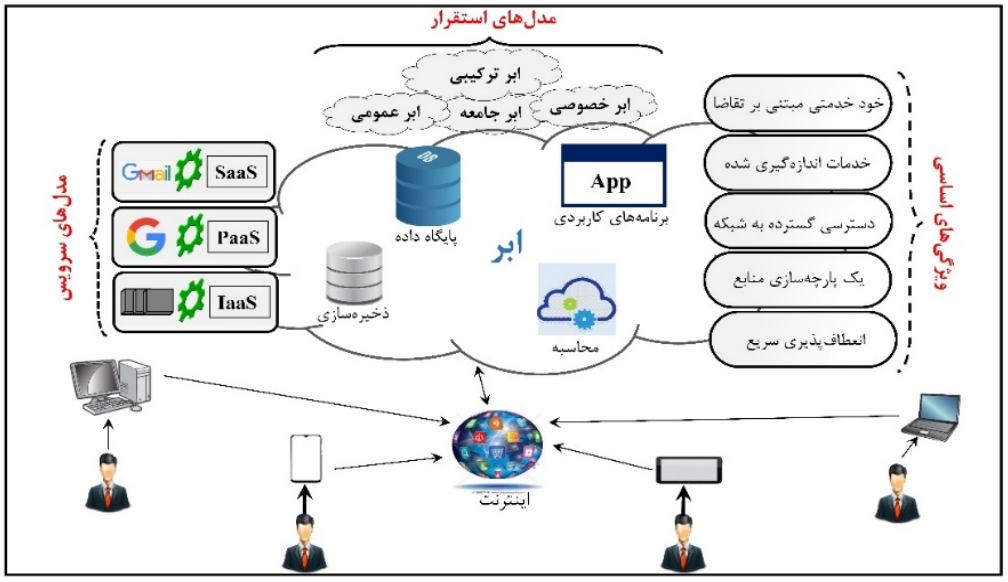
بسیاری از منابع محاسباتی در یک ابر می‌توانند منجر به یک سطح حمله بزرگ شوند. امنیت به‌عنوان بزرگ‌ترین مشکل نسبت داده‌شده به رایانش ابری است و در رتبه اول قرار دارد.

۱-۱- رایانش ابری

رایانش ابری مدلی است برای امکان دسترسی در همه‌جا، راحت و بر اساس درخواست کاربر از طریق شبکه به مجموعه مشترکی از منابع محاسباتی قابل تنظیم (به‌عنوان مثال شبکه‌ها، مراکز داده‌ها، فضای ذخیره‌سازی، برنامه‌ها و خدمات) که می‌تواند با حداقل تلاش مدیریتی یا تعامل ارائه‌دهنده خدمات به‌سرعت تهیه و آزاد شود. هدف اصلی رایانش ابری ارائه خدمات ابری به مصرف‌کنندگان از هرجای دنیا و بر اساس نیاز آن‌ها (از نظر مهلت، کیفیت خدمات، هزینه، قابلیت پردازش و غیره) است [۳]. رایانش ابری دارای مزیت‌های فراوانی است، از جمله مزیت‌های آن می‌توان به هزینه نگهداری پایین، مقیاس‌پذیری و انعطاف‌پذیری و همچنین کاهش هزینه اشاره کرد [۴]. همان‌طور که در شکل (۱) نمایش داده شده است از سه مدل خدمات[1] در ابر استفاده می‌شود: ۱) زیرساخت به‌عنوان یک خدمت که زیرساخت‌ها یا ماشین‌های مجازی را در صورت تقاضا فراهم می‌کند، ۲) سکو[2] به‌عنوان یک خدمت که برای برنامه‌ها و سایر توسعه‌ها استفاده می‌شود و ۳) نرم‌افزار به‌عنوان خدمت که از وب برای ارائه برنامه‌هایی که توسط یک فروشنده شخص ثالث مدیریت می‌شوند استفاده می‌کند.

ویژگی‌های منحصربه‌فردی که رایانش ابری دارد مانند خدمات اندازه‌گیری شده، دسترسی گسترده به شبکه، خودخدمتی مبتنی بر تقاضا، انعطاف‌پذیری سریع و یکپارچه‌سازی منابع، باعث جلب‌توجه بسیاری از کاربران به ابر شده است. چهار مدل استقرار ابر وجود دارد: ۱) ابر عمومی که خدمات زیرساختی (به‌عنوان مثال، مراکز داده‌ها، فضای ذخیره‌سازی، شبکه‌ها، سیستم‌عامل‌های توسعه و غیره) برای عموم ارائه می‌شوند، ۲) ابر خصوصی که منابع ابر را برای استفاده انحصاری توسط یک سازمان واحد ارائه می‌دهد، ۳) ابر جامعه که منابع ابر در بین چندین سازمان که نگرانی‌های مشترکی دارند به اشتراک گذاشته می‌شوند و ۴) ابر ترکیبی که ترکیبی از سایر مدل‌های ابر است [۵-۶].

به‌دلیل ماهیت پیچیده و پویای محیط ابر، روش‌های زمان‌بندی کار نقش مهمی در استفاده از مزایای ابر دارند. برای استفاده مناسب از منابع در مقیاس‌های مختلف، محاسبات ابری نیاز به الگوریتم‌های کارآمد زمان‌بندی کار برای مدیریت منابع دارند.



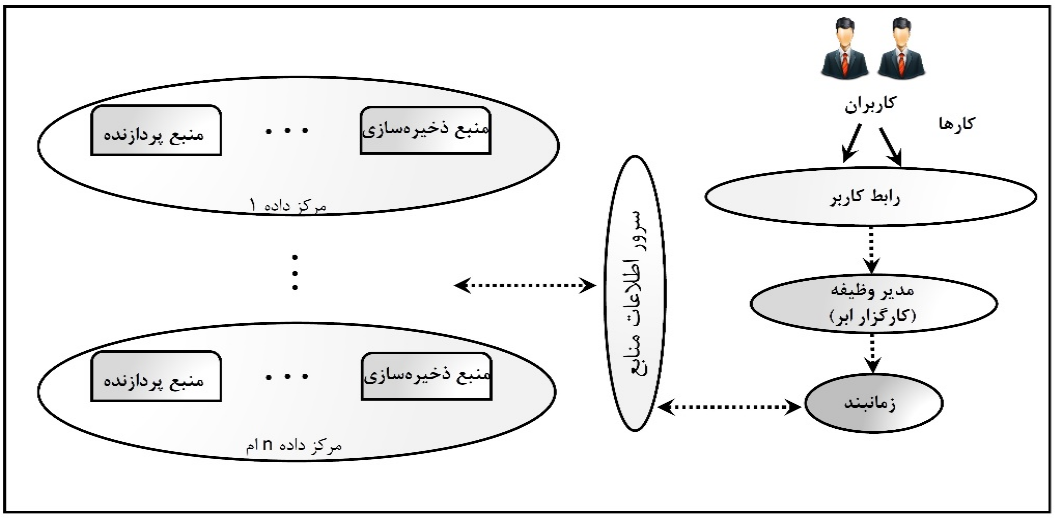
شکل (۱): رایانش ابری

۱-۲- زمان‌بندی کار

زمان‌بندی به‌عنوان سازوکاری تعریف می‌شود که تعیین می‌کند کدام کار برای اجرای به ماشین‌های یک محیط توزیع‌شده اختصاص یابد. شکل (۲) فرایند زمان‌بندی کار در محیط ابر را نشان می‌دهد. امروزه محیط ابری، طرح بهتری را برای انجام کارهای ارسالی از نظر پاسخگویی، مقیاس‌پذیری و انعطاف‌پذیری ارائه می‌دهد.

[1] Services

[2] Platform



شکل (۲): زمان‌بندی کار در محیط رایانش ابری [۷].

مسئله زمان‌بندی کار به‌دلیل پویایی محیط محاسبات ابری و همچنین به‌دلیل ناهمگون بودن محیط ابر به‌عنوان یک مسئله سخت شناخته می‌شود که در زمان چندجمله‌ای حل نمی‌گردد [۸].

الگوریتم‌های زمان‌بندی را می‌توان به دو قسمت دسته‌بندی کرد: ۱) الگوریتم زمان‌بندی ایستا و ۲) الگوریتم زمان‌بندی پویا.

الگوریتم‌های زمان‌بندی ایستا به اطلاعات مربوط به‌کار (مانند طول کار، تعداد کارها و مهلت انجام کار) و منابع (مانند ظرفیت پردازش ماشین، قدرت پردازش و حافظه) قبل از اجرا نیاز دارند. الگوریتم‌های ایستا وقتی تغییرات در حجم کار بسیار کم باشد و رفتار سیستم دائماً تغییر نکند، گزینه مناسبی هستند و به‌خوبی کار می‌کنند، اما اگر بار دائماً در محیط نوسان و تغییر کند، الگوریتم‌های ایستا گزینه مناسبی برای رایانش ابری نیستند. اجرای الگوریتم‌های ایستا بسیار آسان است، اما این الگوریتم‌ها پارامترهای کیفیت خدمات را بهینه نمی‌کنند و عملکرد خوبی را در محیط واقعی ارائه نمی‌دهند؛ بنابراین، برای محیط ابری به الگوریتم زمان‌بندی کار پویا نیاز داریم.

در روش زمان‌بندی پویا، الگوریتم نیازی به اطلاعات کامل قبلی در مورد کار و ماشین ندارد اما نیاز به نظارت دائمی بر ماشین‌های محیط وجود دارد. این الگوریتم‌ها برای محیط ابری کارآمدتر و دقیق‌تر هستند زیرا اگر بار ماشینی بیش از حد باشد، کار را از ماشین پربار به ماشین با بار کم منتقل می‌کند، یعنی شرایط الگوریتم اغلب با تغییرات بار (افزایش یا کاهش) در یک ماشین تغییر می‌کند [۹]. با استفاده از الگوریتم‌های پویا می‌توان حل تقریبی مسائل پیچیده را پیدا کرد. الگوریتم‌های ایستا و پویا هرکدام دارای مزایا و معایبی هستند که در جدول (۱) نشان داده شده است.

از جمله اهداف اصلی برای زمان‌بندی کار بهبود تعادل بار، توان عملیاتی سیستم، کیفیت خدمت، زمان اجرا و مسائل اقتصادی می‌باشد [۱۱-۱۰]. پارامترهای زیادی وجود دارد که بر زمان‌بندی کار در محیط ابر تأثیر می‌گذارند، مانند: زمان پاسخ، منابع ابری، مصرف انرژی، امنیت، هزینه، اولویت کار، قابلیت اطمینان و غیره. با این وجود، اکثر روش‌های زمان‌بندی یک یا دو پارامتر مانند زمان اجرا را در نظر می‌گیرند [۱۲]. از طرف دیگر، ماهیت محیط ابر اغلب منجر به مسائل مربوط به انرژی می‌شود که برای افزایش سود ارائه‌دهندگان خدمات و حفاظت از محیط زیست مورد بررسی قرار می‌گیرند. همچنین در محیط‌های ابری، ارائه خدمات امن در مقایسه با سامانه‌های رایانه‌ای سنتی بسیار دشوار است زیرا بسیاری از منابع توزیع‌شده می‌توانند منجر به یک سطح حمله بزرگ شوند. درنتیجه، امنیت به‌عنوان بزرگ‌ترین چالش در رایانش ابری شناخته می‌شود [۱۳].

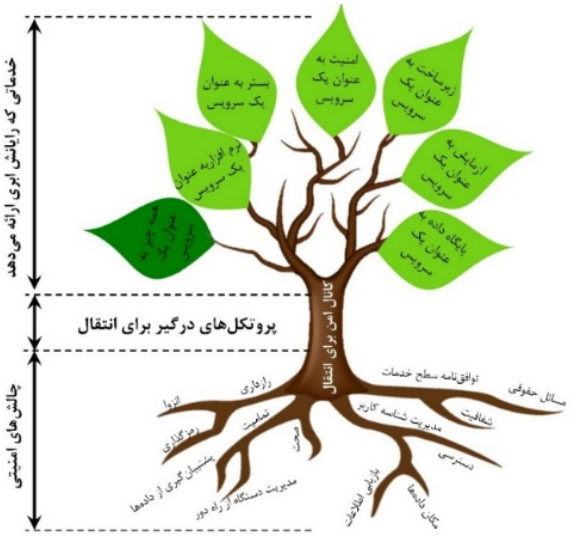
۱-۳- امنیت در رایانش ابری

شکل (۳) درخت امنیتی را نشان می‌دهد که اهمیت الزامات اساسی امنیتی را ذکر می‌کند. چالش‌های مشخص‌شده در ریشه باید به‌درستی حل شوند، دقیقاً همان‌طور که ریشه، درخت را در خاک ایمن می‌کند. وقتی این الزامات اساسی به‌طور مناسب برآورده شوند، درخت امنیت بازدهی سود را از نظر هر چیزی به‌عنوان خدمت[1]، (که به‌صورت استعاری با برگ درخت نشان داده شده است)، تضمین می‌کند.

[1] XaaS

جدول (۱): مقایسه الگوریتم‌های زمان‌بندی پویا و ایستا [۱۴].

الگوریتم پویا	الگوریتم ایستا
در مورد کارها و منابع نیازی به اطلاعات قبلی نیست.	قبل از اجرا به اطلاعاتی در مورد کارها نیاز است.
الگوریتم‌های پویا برای مسائل محاسباتی بزرگ مناسب هستند و نتیجه مطلوبی را ارائه می‌دهند.	الگوریتم‌های ایستا نمی‌توانند نتایج مطلوبی برای مسائل محاسباتی بزرگ ارائه دهند.
تصمیم زمان‌بندی در زمان اجرا گرفته می‌شود.	تصمیم زمان‌بندی در زمان کامپایل گرفته می‌شود.
الگوریتم‌های پویا در زمان کمتری مسائل محاسباتی را حل می‌کنند.	الگوریتم‌های ایستا برای حل مسائل محاسباتی زمان بیشتری را صرف می‌کنند.
الگوریتم‌های فرااکتشافی تحت الگوریتم پویا قرار می‌گیرند.	فقط الگوریتم‌های سنتی تحت الگوریتم ایستا قرار می‌گیرند.
الگوریتم‌های پویا، ماشین را به‌طور مداوم کنترل می‌کنند (یا بر اساس رویداد یا فاصله زمانی).	الگوریتم‌های ایستا به‌طور مداوم گره را کنترل نمی‌کنند.
اجرای آن ساده نیست زیرا پیچیدگی زیاد است.	به‌دلیل پیچیدگی کم، اجرای آن ساده است.
با استفاده از الگوریتم‌های پویا می‌توان نتایج بهینه را برای مسائل چندهدفه را پیدا کرد.	یافتن نتیجه بهینه برای مسائل چندهدفه توسط الگوریتم‌های ایستا سخت است.
الگوریتم‌های پویا، وقتی بار کار مداوم تغییر می‌کند، به‌خوبی کار می‌کنند.	الگوریتم‌های ایستا زمانی شرایط ثابت است، به‌خوبی کار می‌کنند.
الگوریتم‌های پویا بار کار را به روشی کارآمد در ماشین‌ها متعادل می‌کنند.	الگوریتم‌های ایستا در ماشین‌های مجازی در حال اجرا (گره) بار کار را به‌درستی متعادل نمی‌کنند.



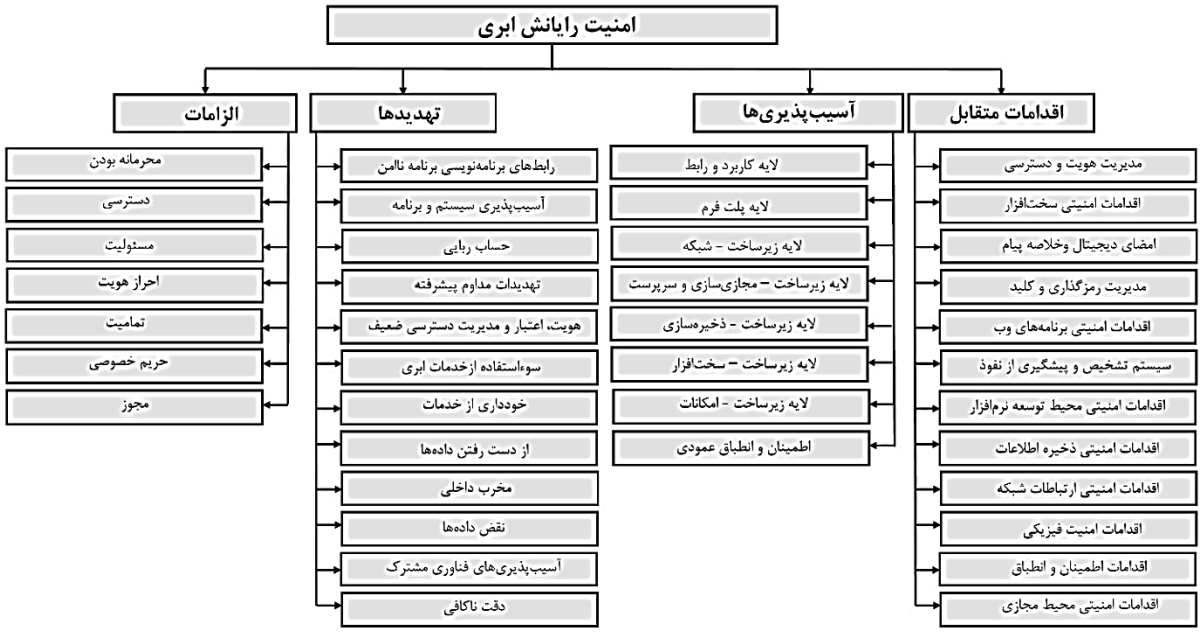
شکل (۳): درخت امنیت در رایانش ابری [۱۵].

درگذشته، عملکرد بالا اولین هدف در سامانه‌های توزیع‌شده بود و این نگرانی بدون توجه زیاد به پارامترهای امنیتی موردتوجه قرارگرفته بود. اما خدمات رایانش ابری عمومی بوده و مدیریت داده‌های امانت گذاشته‌شده به‌دلیل سهولت هدف‌گیری نفوذگران کار دشواری است.

با استفاده از رایانش ابری همه‌ی اطلاعات کاربران در ابر ذخیره می‌شود. در نتیجه سؤالاتی برای کاربران ابر پیش می‌آید، به‌عنوان مثال فاکتور امنیت در این ابر به چه نحوی است؟، آیا امکان دست‌یابی به اطلاعات شخصی و محرمانه توسط کاربران غیرمجاز وجود دارد یا خیر و ده‌ها سؤال دیگر، در نتیجه امنیت به یک نگرانی دائمی در سامانه‌های ابری تبدیل شده است.

با اولویت‌ترین جنبه برای هر نوع محاسبات امنیت است، و این نشان‌دهنده آن است که مسائل امنیتی برای محیط ابری نیز مهم است [۱۶]. ازآنجاکه رویکرد رایانش ابری می‌تواند با ذخیره‌سازی اطلاعات حساس کاربران هم در سمت کاربران و هم در مراکز داده ابری همراه باشد، مدیریت هویت و احراز هویت در رایانش ابری بسیار مهم است [۱۷]. مدل‌سازی امنیت در رایانش ابری شامل، شناسایی الزامات امنیتی ابر و تهدیدات مربوط به آسیب‌پذیری در اجزای معماری ابر است که بردارهای حمله را تشکیل می‌دهند و بر تحقق این الزامات تأثیر می‌گذارند. همچنین، این موضوع شامل تعیین اقدامات متقابل برای رفع این آسیب‌پذیری‌های شناخته‌شده برای کاهش یا از بین بردن بردارهای حمله برای تحقق امنیت ابر و الزامات حریم خصوصی است [۱۸]. شکل (۴) طبقه‌بندی امنیت ابری را نشان می‌دهد که برای شناسایی و توصیف الزامات مختلف امنیتی ابر، تهدیدات تأثیرگذار بر این الزامات، آسیب‌پذیری در اجزای معماری مرجع رایانش ابری و فناوری‌های اساسی که این تهدیدات را تشکیل می‌دهد و اقدامات متقابل برای رفع این آسیب‌پذیری‌ها است. این طبقه‌بندی بر اساس مطالعات رهنمودها و استانداردهای منتشرشده توسط سازمان‌هایی که در زمینه امنیت ابر فعالیت می‌کنند و کارهای منتشر شده محققان در حوزه ابر تعریف شده است [۱۹].

با توجه به اهمیت امنیت، خصوصاً برای کارهایی که از حساسیت بیشتری برخوردار هستند (مانند حوزه دفاعی یا پزشکی) در طراحی الگوریتم زمان‌بندی کارها باید به پارامتر امنیت توجه نمود [۲۰]. اگر روش مناسبی برای حملات غیرقابل پیش‌بینی در محیط ابری در نظر گرفته نشود، متوقف شدن اجرای کارها منجر به هزینه بیشتر می‌شود و اثرات منفی بر کیفیت ارائه خدمت خواهد داشت. راهبرد پاسخ به حملات را می‌توان در قالب یک عملیات زمان‌بندی دوباره تعریف کرد. متأسفانه، بیشتر روش‌های زمان‌بندی کار برای سامانه‌های مقیاس بزرگ نامناسب هستند زیرا نتایج این روش‌ها معمولاً از نظر مصرف انرژی و مسائل امنیتی بهینه نیستند [۲۱]. درحالی‌که امنیت یکی از پارامترهای مهم در محیط رایانش ابری می باشد و درنتیجه ایده اصلی در روش پیشنهادی این است که با در نظر گرفتن سطح امنیتی هر یک از ارائه‌دهنده‌ها در حین زمان‌بندی، منجر به بهبود امنیت شود درحالی‌که کارایی سیستم افت نکند.



شکل (۴): طبقه‌بندی امنیت ابری [۲۲].

در این مقاله، بر استفاده از منابع و امنیت اجرا تمرکز شده است. الگوریتم زمان‌بندی کار پیشنهادی با استفاده از روش فرااکتشافی ازدحام ذرات، تعادلی را بین چندین پارامتر از جمله (هزینه، زمان، بار، انرژی و امنیت) در حین تخصیص کارها بین ماشین‌ها برقرار می‌کند. سرانجام، الگوریتم پیشنهادی از طریق شبیه‌سازی‌های گسترده ارزیابی می‌شود. نتایج ثابت می‌کند که الگوریتم پیشنهادی در بهبود عملکرد بسیار مؤثر و کارآمد است و می‌تواند امنیت و خدمات کارآمد را برای برنامه‌ریزی کارها در محیط ابر ارائه دهد.

ادامه این مقاله به شرح زیر است. بخش ۲ بررسی مقالات مرتبط را ارائه می‌دهد. بخش ۳ الگوریتم ازدحام ذرات را شرح می‌دهد. بخش ۴ بهینه‌ساز ازدحام ذرات با یادگیری انطباقی چندگانه را معرفی می‌نماید. در بخش ۵، روش پیشنهادی زمان‌بندی کار برای بهبود امنیت توصیف می‌گردد. بخش ۶ ارزیابی ساز و کار پیشنهادی را توضیح می‌دهد و مزایای آن را در شرایط مختلف محیط نشان می‌دهد. سرانجام نتیجه‌گیری و پیشنهادها در بخش ۷ آورده شده است.

۲- کارهای مرتبط

امنیت یکی از مهم‌ترین اجزا در محیط‌های ابری است. الگوریتم‌های زمان‌بندی کارآمد می‌توانند نیازهای کاربر را برآورده کنند. اگرچه، تحقیقات زیادی در زمینه تخصیص منابع انجام شده است، اما زمان‌بندی وظایف همچنان مسئله اصلی در صنعت و حوزه تحقیقاتی است [۲۴-۲۳]. این بخش برخی از روش‌های زمان‌بندی موجود برای محاسبات ابری را مرور می‌کند.

سررو و همکاران [۲۵] یک روش زمان‌بندی کار برای بهبود مصرف انرژی و امنیت در محیط ابر ارائه کردند. الگوریتم پیشنهادی محدودیت‌های امنیتی مختلفی را در نظر می‌گیرد. هر محدودیت امنیتی بر اساس خواسته‌های امنیتی[1] کارها و سطح اعتماد[2] ارائه‌شده توسط مراکز داده فراهم می‌شود. سپس کاربر می‌تواند یکی از سطوح امنیتی احتمالی را بر اساس مدل امنیتی پیشنهادی انتخاب کند؛ بنابراین، کاربران بر اساس اطلاعات ارائه‌شده قادر به تعیین کلیدهای بلند یا کوتاه برای مراحل رمزنگاری هستند. نتایج آزمایش‌ها ثابت کرد که مدل پیشنهادی منجر به‌کارایی خوبی در محیط محاسباتی برای تخصیص کارها به ماشین‌های محاسباتی می‌شود.

شیشیدو و همکاران [۲۶] اثرات روش‌های فرااکتشافی مختلف (یعنی، بهینه‌سازی ازدحام ذرات[3]، الگوریتم ژنتیک[4] و الگوریتم ژنتیک چند جمعیتی[5]) را در الگوریتم زمان‌بندی کار در محیط ابر بررسی کردند. نویسندگان، یک الگوریتم زمان‌بندی گردش کار آگاه از امنیت و هزینه [۲۷] را به‌عنوان روش پایه استفاده کردند. نتایج شبیه‌سازی بر اساس کاربردهای واقعی علمی نشان داد که الگوریتم ژنتیک چند جمعیتی برای مسائل مربوط به زمان‌بندی با حجم بار کم مناسب است.

اسماعیل و ماتروالا [۲۸] برای کاهش مصرف انرژی در سیستم ابری، الگوریتم زمان‌بندی کار آگاه از انرژی مصرفی

[1] Security demands
[2] Trust levels
[3] Particle Swarm Optimization (PSO)
[4] Genetic-based Algorithms (GA)
[5] Multi-Population Genetic Algorithm (MPGA)

ماشین مجازی را ارائه دادند. الگوریتم معرفی‌شده، ماشین‌های مجازی فعال و غیرفعال را برای تعیین ماشین مجازی جهت اجرای کار در نظر می‌گیرد. همچنین افزایش مصرف انرژی وظایف در حال اجرا بر روی یک ماشین مجازی را کنترل می‌کند. نتایج شبیه‌سازی نشان داد که الگوریتم مطرح‌شده در مقایسه با روش اتحاد کار آگاه از انرژی[1] [۲۹] در مصرف انرژی بیشتر صرفه‌جویی می‌کند. زیرا الگوریتم مطرح‌شده میزان انرژی مصرفی ماشین‌های مجازی فعلی را در تصمیم‌گیری در نظر می‌گیرد.

ژانگ [۳۰] الگوریتم زمان‌بندی امنیت کار[2] را معرفی کرد. نویسنده یک معماری ابر با چهار لایه با عنوان‌های: نرم‌افزار میانه مدیریت، معماری مبتنی بر خدمت، مجازی‌سازی منابع و منابع فیزیکی را پیشنهاد کرد. سپس، برای تأمین سطح امنیتی زمان‌بندی کار، تقاضای امنیتی و سطح اعتماد را در نظر گرفت.

پس از آن، الگوریتم پیشنهادشده مسئله زمان‌بندی را بر اساس الگوریتم ژنتیک مدل می‌کند. در مدل پیشنهادی، هر کروموزوم مجموعه‌ای از کارها را در چندین مرکز داده نشان می‌دهد و سپس هر ژن به‌عنوان یک جفت برای نشان دادن روابط بین کارها و مراکز داده تعریف می‌شود. نتایج نشان داد که الگوریتم زمان‌بندی پیشنهادی می‌تواند زمان پردازش را برای تعداد مختلف کارها کاهش دهد.

لیو و ژو [۳۱] یک الگوریتم مبتنی بر بینایی خواب لایه‌ای خودسازگار برای حل مشکل زمان‌بندی ایمن در محیط ابر ارائه کردند. نویسندگان از یک ساختار درخت تصمیم‌گیری برای طبقه‌بندی ویژگی‌های امنیت منابع کارها استفاده کردند. علاوه بر این، آن‌ها یک رویکرد تحلیلی از بالا به پایین را برای بازسازی فضای چند رشته‌ای منابع در ذخیره‌سازی ابر در نظر گرفتند. سرانجام، آن‌ها یک روش انتخاب خود تطبیقی را برای حذف منابع اضافی ارائه نمودند. میانگین دقت زمان‌بندی الگوریتم پیشنهادی در حدود ۹۵/۵٪ است که از نظر صحت برای برنامه‌ریزی پویای امنیت منابع در محیط ابری برتر از روش‌های سنتی است.

لو و همکاران [۳۲] الگوریتم زمان‌بندی وظیفه مبتنی بر ژنتیک را ارائه دادند که نه‌تنها زمان اجرا را کاهش می‌دهد بلکه تعادل بار را نیز تضمین می‌کند. نویسندگان پنج شرط، یعنی: رسیدن کار جدید، انتظار طولانی‌مدت، بیکار بودن منبع، آزمایش خرابی و بارگذاری زیاد منابع در طی فرایند زمان‌بندی کار را در نظر گرفتند. نتایج شبیه‌سازی نشان داد که الگوریتم زمان‌بندی کار پیشنهادی می‌تواند باعث کاهش زمان و تحقق تعادل بار شود.

منصوری و جاویدی [۳۳] الگوریتم زمان‌بندی کار برای کاهش هزینه[3] در محیط ابر ارائه دادند. الگوریتم مطرح شده هم‌زمان ویژگی‌های اطلاعات فشرده و محاسبات فشرده کارها را در نظر می‌گیرد. علاوه بر این، ویژگی‌های شبکه مانند از بین رفتن و از دست دادن بسته را مدل می‌کند. در آخر، یک تابع هزینه خطی را بر اساس هزینه انتقال، هزینه شبکه و هزینه محاسبه تعریف می‌کند. نتایج شبیه‌سازی با کلودسیم نشان داد که الگوریتم مطرح شده می‌تواند باعث بهبود استفاده از پردازنده شود.

چار و همکاران [۳۴] الگوریتم زمان‌بندی کار بهینه[4] جدیدی را معرفی کردند که ساختار داده‌ای مبتنی بر درخت را برای اجرای کارها به روش کارآمد اعمال می‌کند. در ابتدا، نویسندگان وظایف را با توجه به‌اندازه و ماشین‌های مجازی بر اساس میزان توان پردازش اولویت‌بندی کردند؛ بنابراین، کار دارای بالاترین اندازه دارای بالاترین رتبه است.

آن‌ها درختی به نام درخت ماشین مجازی[5] ساختند که در آن هر ماشین مجازی با یک گره نشان داده می‌شود. سپس، الگوریتم پیشنهادی گروه‌بندی کارها را بر اساس تعداد برگ در درخت ماشین مجازی اجرا می‌کند. نتایج شبیه‌سازی با کلودسیم ثابت کرد که الگوریتم پیشنهادی عملکرد بهتری در مقایسه با دیگر الگوریتم‌های زمان‌بندی سنتی مانند خدمت به ترتیب ورود[6] دارد. جدول (۲) به‌صورت خلاصه الگوریتم‌های زمان‌بندی ذکرشده را مقایسه می‌کند.

۳- الگوریتم بهینه‌سازی ازدحام ذرات استاندارد[7]

الگوریتم بهینه‌سازی ذرات یک روش بهینه‌سازی تصادفی، مبتنی بر جمعیت است [۳۵]. این الگوریتم، رفتار جمعی پرندگان در یافتن بهترین مکان را شبیه‌سازی می‌کند؛ بنابراین، PSO در یافتن مقدار بهینه وابسته به حرکت ذرات در فضای جستجو است. در سال ۲۰۱۹، آروناران و همکاران [۳۶] بررسی جامعی از الگوریتم‌های زمان‌بندی کارها در ابر ارائه دادند و نتایج نشان داد که روش‌های بهینه‌سازی ازدحام ذرات (PSO) و الگوریتم ژنتیک (GA)، توسط محققان به‌شدت در توسعه رویکردهای زمان‌بندی و تخصیص پویای منابع برای محاسبات ابری مورداستفاده قرار گرفته است.

[1] Energy Conscious Task Consolidation (ECTC)
[2] Job Security Scheduling Strategy (JSSS)

[3] Cost-based Job Scheduling (CJS)
[4] Optimal Task Scheduling Strategy (OTSS)
[5] Virtual Machine Tree (VMT)
[6] First Come First Serve (FCFS)
[7] Standard PSO

جدول (۲): مقایسه الگوریتم‌های زمان‌بندی.

معیارها/ الگوریتم‌ها	سال	محیط	نوع کار	زمان اجرا	قابلیت اطمینان	تعادل بار	انرژی	امنیت	ابزار	ایده اصلی	معایب
سررو و همکاران [۲۵]	۲۰۱۷	ناهمگن	پویا	✓	×	×	✓	✓	Score simulator	ترکیب انرژی و سیاست‌های برنامه‌ریزی آگاه از عملکرد	- پارامترهای کیفیت خدمات مانند هزینه، بار و غیره را لحاظ نکرده است، - با هیچ الگوریتم پیشرفته‌ای مقایسه نشده است.
شیشیدو و همکاران [۲۶]	۲۰۱۷	ناهمگن	ایستا	✓	×	×	×	✓	CloudSim	ارزیابی سه الگوریتم فرااکتشافی	- بیشتر تمرکز بر روی دو پارامتر (یعنی امنیت و هزینه) است و پارامترهای مهمی مانند انرژی و بار را لحاظ نکرده است.
اسماعیل و ماتروالا [۲۸]	۲۰۱۸	ناهمگن	ایستا	✓	×	×	✓	×	-	در نظر گرفتن ماشین‌های مجازی فعال و غیرفعال و همچنین توجه به افزایش مصرف انرژی	- محدودیت مهلت و اولویت در نظر گرفته نشده است، - الگوریتم فقط مصرف انرژی و زمان اجرا را لحاظ کرده است و سایر پارامترهای کیفیت خدمات مانند هزینه، امنیت و در دسترس بود را در نظر نگرفته است.
ژانگ [۳۰]	۲۰۱۵	ناهمگن	ایستا	✓	×	×	×	✓	GridSim	تمرکز بر امنیت کار و توصیف روابط بین کارها و ماشین‌ها	- فرایند تحلیل روابط کارها زمان‌بر است.
لیو و ژو [۳۱]	۲۰۱۷	-	پویا	×	✓	×	×	✓	Matlab	استفاده از درخت تصمیم، روش تحلیلی از بالا به پایین و همچنین الگوریتم فیلتر خود انطباقی	-ساخت درخت تصمیم باعث پیچیدگی روش پیشنهادی شده است، - با الگوریتم‌های پیشرفته مقایسه نشده است.
لو و همکاران [۳۲]	۲۰۱۴	-	پویا	✓	×	✓	×	×	CloudSim	استفاده از الگوریتم ژنتیک	- فضای راه‌حل در الگوریتم ژنتیک زیاد است، - پارامترهایی مانند امنیت، هزینه و انرژی لحاظ نشده است.
منصوری و جاویدی [۳۳]	۲۰۱۹	-	پویا	✓	✓	✓	×	×	CloudSim	استفاده از داده‌ها، قدرت پردازش و خصوصیات شبکه برای تخصیص کار	- در الگوریتم مطرح شده از هیچ الگوریتم فرااکتشافی استفاده نشده است، - پارامترهای مهمی مانند انرژی و امنیت مورد توجه قرار نگرفته‌اند.
چار و همکاران [۳۴]	۲۰۱۲	-	ایستا	✓	×	×	×	×	CloudSim	به‌کار بردن ساختار داده‌ای درختی برای ماشین‌های مجازی	- به‌صورت ایستا کارها تخصیص می‌یابد، - پارامترهای با اهمیت کیفیت خدمات مانند امنیت، انرژی، زمان بازگشت، بار و غیره را لحاظ نکرده است.

الگوریتم PSO در دسته الگوریتم‌های هوش دست جمعی و الگوریتم GA در دسته‌بندی الگوریتم‌های تکاملی قرار گرفته‌اند.

الگوریتم‌های هوش دست جمعی بر پایه مفهوم تعامل چندین عامل (ذره) با یکدیگر و ایجاد تبادل اطلاعات بر طبق اصولی

ساده بنا شده‌اند. از طرفی الگوریتم‌های تکاملی مانند GA رفتار اجتماعی الگوریتم‌های هوش دست جمعی را دنبال نکرده و سعی می‌کنند با ترکیب، مخلوط یا انجام جهش در هر عامل، عاملی با ماهیت جدید ایجاد کنند و به اصطلاح تنوع در جمعیت ایجاد کنند. در روش PSO، به‌روزرسانی عامل‌ها (ذرات) توسط بهترین موقعیت هر عامل (Pbest) و بهترین موقعیت جمعی (Gbest) انجام می‌پذیرد به این صورت ذرات سعی می‌کنند در دو جهت حرکت کنند. این عملکرد این اجازه را به PSO می‌دهد تا تنوع و اکتشاف بیشتر در فرایند جستجو اتفاق بیفتد. همچنین، اثرات حرکت ذرات می‌تواند منجر به همگرایی سریع‌تر و تنوع بیشتر در مسیرهای جستجو شود. به همین دلیل در تحقیقات گذشته نشان داده شده است که روش PSO نسبت به بسیاری از روش‌های تکاملی از جمله GA از نظر سرعت پردازش و پیاده‌سازی بهتر عمل می‌کند و از آنجایی که در مسئله زمان‌بندی کارها برای رایانش ابری، زمان پارامتر مهمی محسوب می‌شود، در این مقاله از روش PSO استفاده شده است.

در این مقاله، دامنه فضای جستجو یا به عبارتی فضای مسئله D در نظر گرفته شده است و مکان و سرعت iامین ذره در زمان t به‌صورت $\left\{X_i^1(t),X_i^2(t),...,X_i^D(t)\right\}$ و $\left\{V_i^1(t),V_i^2(t),...,V_i^D(t)\right\}$, $i=1,2,3,...,N$ تعریف می‌شود که در آن N تعداد ذرات در جمعیت است. سرعت و مکان ذره iام به‌صورت زیر به‌روزرسانی می‌شود [۳۷]:

$$V_i(t+1)=\omega(t)V_i(t)+c_1 r_1(Pbest_i(t)-X_i(t))+ c_2 r_2(Gbest(t)-X_i(t)) \quad (۱)$$

$$X_i(t+1)=X_i(t)+V_i(t+1) \quad (۲)$$

که در معادلات شماره (۱) و (۲)، r_1 و r_2 دو عدد تصادفی بین صفر و یک می‌باشند و $c_1=c_2=2$ را به نام‌های پارامترهای شتاب معرفی می‌کنند. بهترین مقادیر شخصی (Pbest) و بهترین مقدار کلی (Gbest) به ترتیب توسط معادلات (۳) و (۴) حساب می‌شوند [۳۷].

$$Pbest_i(t)= \arg\min\{fit(X_i(1)), fit(X_i(2)),..., fit(X_i(t))\} \quad (۳)$$

$$Gbest_i(t)= \arg\min\{fit(Pbest_1(t)), fit(Pbest_2(t)),..., fit(Pbest_N(t))\} \quad (۴)$$

در معادله یک، پارامتر ω را وزن اینرسی[1] گویند که به‌صورت نزولی در طی دورهای مختلف مقدار آن تعیین می‌شود. این مقدار توسط معادله (۵) قابل محاسبه است [۳۷].

$$\omega(t)=\omega_{\max}-\frac{(\omega_{\max}-\omega_{\min})}{T_{\max}}t \quad (۵)$$

که در معادله (۵)، $\omega_{\max}=0.9$ و $\omega_{\min}=0.4$، $T_{\max}$ تعداد دورهایی که PSO باید انجام شود و t شماره دوری است که PSO در حال حاضر آن را انجام می‌دهد. با در نظر گرفتن معادلات شماره یک و دو مشخص است که با افزایش تعداد دورها، مکان ذرات نیز تغییر می‌یابد.

۴- بهینه‌ساز ازدحام ذرات با یادگیری انطباقی چندگانه (MAPSO)

یکی از مواردی که به PSO کمک می‌کند تا در بهینه محلی نیفتد، تنوع در جمعیت[2] است. امروزه یادگیری انطباقی[3] راهی مؤثر در بهبود تنوع در جمعیت محسوب می‌شود. با این وجود در PSO استاندارد، یادگیری عمدتاً بر روی بهترین ذره تمرکز[4] دارد و این باعث از دست رفتن تنوع در جمعیت می‌شود. برای افزایش تنوع جمعیت و بالا بردن قدرت جستجو در PSO استاندارد، این مقاله نسخه جدیدی از PSO استاندارد به نام الگوریتم بهینه‌ساز ازدحام ذرات با یادگیری انطباقی چندگانه[5] یا به‌اختصار MAPSO معرفی کرده است.

به‌عبارت‌دیگر، در این بخش الگوریتمی به نام بهینه‌ساز ازدحام ذرات با یادگیری انطباقی چندگانه یا به‌اختصار MAPSO معرفی شده است که این الگوریتم نسخه جدیدی از PSO استاندارد است. در الگوریتم پیشنهادی جمعیت تشکیل‌شده ابتدایی به چند زیرجمعیت تقسیم می‌شود. این عمل مشابه روش PSO-ALS بیان‌شده در [۳۷] است. اما روش یادگیری انطباقی معرفی شده در [۳۷] گرچه توانسته به بهبود PSO استاندارد کمک کند و با شکستن فضای مسئله توسط زیرجمعیت‌ها، باعث تنوع در ذرات شده و راه‌حل‌های بیشتری را مورد ارزیابی قرار دهد اما به‌دلیل استفاده از وزن اینرسی خطی در هنگام به‌روزرسانی ذرات و متعاقب آن برداشتن گام‌های جستجوی ثابت توسط هر ذره، این الگوریتم دچار همگرایی سریع و افتادن در دام بهینه محلی شده است. در روش پیشنهادی مقاله، از نقطه قوت PSO-ALS یعنی شکستن فضای مسئله و ایجاد تنوع در ذرات بهره گرفته شده و نقطه ضعف این الگوریتم در وزن اینرسی را با استفاده از یک تابع آشوب به نام تابع منطقی[6] حل کرده است. بنابراین در الگوریتم پیشنهادی MAPSO، ذرات با گام جستجوی

[1] Inertia weight

[2] Diversity population
[3] Adaptive learning
[4] Global best particle
[5] Multi adaptive learning for particle swarm optimization (MAPSO)
[6] Logistic

متغیر در فضای مسئله حرکت کرده و برخلاف PSO-ALS احتمال افتادن در بهینه محلی و همگرایی زودرس بسیار کم می‌شود.

دو یادگیری انطباقی در بدنه MAPSO گنجانده شده است: ۱) به‌منظور ارتقای هر چه‌بهتر یادگیری، جمعیت کلی به‌صورت انطباقی به چندین زیرجمعیت[1] تقسیم می‌شود. سپس ذرات در هر زیرجمعیت به دو دسته ذرات معمولی[2] و بهترین ذره محلی[3] گروه‌بندی شده و به‌منظور افزایش تنوع جمعیت، دو گروه ذرات به دو شکل متفاوت به‌روزرسانی می‌شوند. ۲) برای ایجاد تعادل بین عملیات اکتشاف[4] و بهره‌برداری[5]، وزن اینرسی خطی در PSO استاندارد، به وزن اینرسی غیرخطی و بر پایه آشوب[6] تغییر داده شده است. در ادامه و در بخش‌های بعد به تشریح جزئیات الگوریتم پیشنهادی می‌پردازیم.

۴-۱- تشکیل زیرجمعیت‌ها

یک الگوریتم یادگیری مبتنی بر روش چند جمعیتی که در آن تعیین اندازه زیرجمعیت‌ها نقش مهمی را هم ایفا می‌کند، می‌تواند باعث ایجاد تنوع در جمعیت شده و در نتیجه از همگرایی زودهنگام یک الگوریتم فرااکتشافی جلوگیری کند. در روش‌های یادگیری موجود، اندازه زیرجمعیت‌ها اغلب به‌صورت از قبل تنظیم شده، یا به کمک پارامترهای متغیر تعیین می‌شوند، این نوع روش‌های تعیین اندازه با توجه به ماهیت محیط‌های الگوریتم‌های فرااکتشافی که پیچیده و پویا هستند نمی‌توانند مؤثر باشند.

ایده اصلی الگوریتم خوشه‌بندی بر پایه این فرض است که ذرات مرکزی در احاطه همسایگان با تراکم محلی پایین و نسبتاً با فاصله از دیگر ذرات با تراکم محلی بالا باشد. دو کمیت برای ذره i معرفی می‌شود: تراکم محلی (ρ_i) و فاصله (δ_i). تراکم محلی نشان‌دهنده تعداد ذراتی است که در یک فاصله معین تا ذره i وجود دارند و با معادله (۶) می‌شود [۳۸]:

$$\rho_i = \sum_{j \in I, j \neq i} e^{-(\frac{d_{ij}}{d_c})^2} \qquad (۶)$$

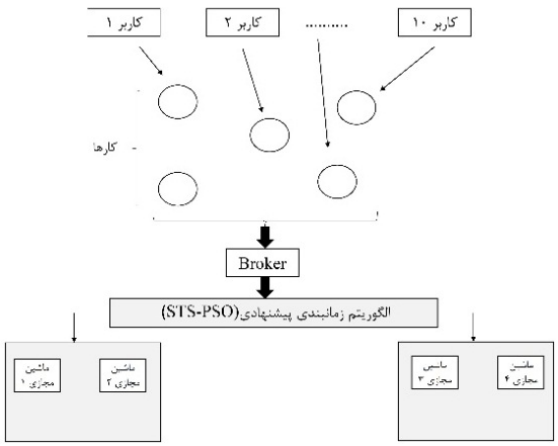
که I مجموعه تمام ذرات است، d_{ij} نشان‌دهنده فاصله اقلیدسی بین ذره i و ذره j و به d_c پارامتر قطع فاصله اطلاق می‌شود. فاصله δ_i به‌عنوان کمترین فاصله بین ذره i و هر ذره دیگر با بالاترین تراکم شناخته می‌شود و با معادله (۷) محاسبه می‌شود [۳۸]:

$$\delta_i = \min_{j \in I, \rho_j > \rho_i}(d_{ij}) \qquad (۷)$$

برای ذره‌ی با بالاترین تراکم $\delta_i = \max_{j \in I}(d_{ij})$ به‌علاوه برای ذراتی با تراکم محلی یکسان، تمام ρ_iهای آن‌ها به‌صورت نزولی مرتب شده سپس δ_i برای هر ذره محاسبه می‌شود. با محاسبه این دو پارامتر، مرکز هر زیرجمعیت ذره‌ای با بالاترین δ_i و متناسب با آن با بالاترین ρ_i خواهد بود. بعد از مشخص شدن مرکز هر زیرجمعیت، دیگر ذرات باقیمانده به نزدیک‌ترین زیرجمعیت به خود و با تراکم محلی بالا ملحق می‌شوند.

۵- الگوریتم MAPSO برای زمان‌بندی کار برای بهبود امنیت (STS-PSO[7])

در این مقاله، مسئله زمان‌بندی به‌صورت دسته‌ای از کارهای مستقل از هم در یک محیط رایانش ابری در نظر گرفته شده است، مدل پیشنهادی و نحوه استفاده از الگوریتم زمان‌بندی در شکل (۵) قابل‌مشاهده است، همان‌طور که در شکل (۵) ملاحظه می‌شود، در مرحله اول کارهای کاربران جمع‌آوری‌شده، سپس با الگوریتم پیشنهادی (STS-PSO) و بر پایه چهار پارامتر زمان بازگشت، امنیت، انرژی مصرف‌شده و هزینه زمان‌بندی کارها زمان‌بندی شده و در آخر کارها اجرا می‌شوند.



شکل (۵): شمای کلی از نحوه کارکرد الگوریتم زمان‌بندی پیشنهادی.

در الگوریتم پیشنهادی، کارها شامل دو مشخصه‌اند: ۱) اندازه (S) و ۲) سطح روش امنیتی درخواستی (sd). مشخصه اول برای

[1] Subswarms
[2] Ordinary particles
[3] Localy best particle
[4] Exploration
[5] Exploiation
[6] Chaos-based
[7] Security based-Task Scheuling-PSO (STS-PSO)

n کار به‌صورت $S=[S_1,...,S_n]$ و مشخصه دوم برای i امین کار به‌صورت $sd_i=[sd_i^1,sd_i^2,sd_i^3]$ تعریف می‌شود. به همراه ارسال کارها توسط کاربران، سطح خدمت امنیتی درخواست شده (sd) در قالب یک آرایه ارسال می‌شود، این آرایه ۳ سلول دارد (برای هر خدمت امنیتی یک سلول) که نمایانگر الگوریتم امنیتی درخواستی کاربر است. هر سلول با عددی مابین یک و صفر مقداردهی می‌گردد که صفر نمایانگر عدم اختصاص و یک نمایانگر اختصاص قوی‌ترین الگوریتم امنیت درخواستی برای اجرای کار است. برای ارائه‌دهنده‌ها (ماشین‌های مجازی) نیز چنین مشخصاتی می‌توان تعریف کرد، در این مقاله هر ماشین مجازی با چهار مشخصه، سرعت پردازنده (CS)، پهنای باند (b)، ظرفیت (CP) و سطح امنیت (sl) تعریف می‌شود.

۵-۱- ساخت جمعیت اولیه

در بخش ۵، از الگوریتم MAPSO برای حل مسئله زمان‌بندی با رویکرد امنیتی در محیط رایانش ابری استفاده شده است. پس از پیدا کردن مناسب‌ترین ماشین‌های مجازی برای اجرای کارها توسط این الگوریتم، کارهایی در ماشین مجازی که نرخ خطر[1] اجرای زیاد داشته باشد شناسایی شده و زمان‌بندی دوباره اجرا می‌شود تا به ماشین‌های مجازی ایمن‌تر با در نظر گرفتن هزینه مناسب منتقل شده و اجرا شوند.

به عبارت دیگر، در این مقاله هدف حل مسئله زمان‌بندی کار است و هر راه‌حل احتمالی به‌صورتی که در ادامه گفته می‌شود به یک ذره نگاشت می‌شود. هر ذره با یک بردار n بعدی یعنی به‌اندازه تعداد کارهای ما تعریف می‌شود. در هر بعد عددی اعشاری تصادفی مابین ۱ تا vn که vn تعداد ماشین‌های مجازی است مقداردهی می‌شود. به‌عنوان مثال، فرض کنید ۵ کار داریم که باید توسط ۳ ماشین مجازی اجرا شوند، بنابراین هر ذره دارای ۵ بعد است و مقدار اولیه در هر بعد، عددی اعشاری بین ۱ تا ۳ قرار داده می‌شود، شکل (۶) نمونه راه‌حلی برای این مسئله را نشان می‌دهد.

کار ۱	کار ۲	کار ۳	کار ۴	کار ۵
۰.۸۲	۱.۹۴	۰.۹۱	۲.۷۶	۱.۶۶

شکل (۶): نمونه‌ای از یک راه‌حل برای مسئله‌ای با ۵ کار و ۳ ماشین.

برای اینکه مشخص شود هر کار به کدام ماشین مجازی اختصاص‌یافته، عدد موجود در هر بعد به سمت بالاگرد می‌شود. چگونگی تخصیص کارها به ماشین‌های مجازی واقع در ذره شکل (۶)، توسط شکل (۷) قابل مشاهده است.

[1] Risk

کار ۱	کار ۲	کار ۳	کار ۴	کار ۵
۰.۸۲	۱.۹۴	۰.۹۱	۲.۷۶	۱.۶۶
ماشین مجازی ۱	ماشین مجازی ۲	ماشین مجازی ۱	ماشین مجازی ۳	ماشین مجازی ۲

شکل (۷): نمونه‌ای از چگونگی اختصاص کارها به ماشین‌های مجازی.

۵-۲- توابع ارزیابی

اکثر الگوریتم‌های زمان‌بندی گذشته، پارامترهای زمان بازگشت، هزینه اجرا و بار را در نظر گرفته‌اند که تأثیر مستقیم بر کارایی سیستم دارند. اما همان‌طور که در مقدمه بیان شد برای کارهایی که از حساسیت بیشتری برخوردار هستند در محیط رایانش ابری بایستی پارامتر امنیت را در طراحی الگوریتم‌های زمان‌بندی لحاظ کرد تا کیفیت ارائه خدمت بهبود یابد. از طرف دیگر، مصرف انرژی از مشکلات مهم می‌باشد که با سایر پارامترهای کارایی در تعارض است در نتیجه علاوه‌بر پارامترهای رایج، امنیت و مصرف انرژی نیز در اهداف گنجانده شدند تا با استفاده از الگوریتم تکاملی تعادلی بین آن‌ها برقرار شود. به عبارت دیگر الگوریتم زمان‌بندی پیشنهادی پنج پارامتر اصلی (امنیت، انرژی، بار، هزینه و زمان بازگشت) را که برای تخصیص کارها در محیط رایانش ابری اهمیت دارند را هم‌زمان در نظر می‌گیرد.

زمان بازگشت[2]: یکی از پارامترهای مهم در زمان‌بندی کارها زمان بازگشت می‌باشد که نشان‌دهنده زمان کامل کل فرایند شامل عملیات انتقال، پذیرش و پیاده‌سازی است. در رابطه (۸)، ERT_{ij} زمان بازگشت برای کار i در ماشین مجازی j را نشان می‌دهد [۳۹]:

$$ERT_{ij}=(\frac{S_j}{b})+\frac{l_i}{n_j}+d \qquad (۸)$$

که در آن، S_j اندازه کار i، b پهنای باند، d تأخیر، l_i تعداد دستورالعمل‌های لازم برای اجرای کار i، n_j تعداد دستورالعمل‌های در ثانیه توسط ماشین j را نشان می‌دهند.

هزینه: یکی دیگر از پارامترهایی که تأمین‌کنندگان خدمات ابری به آن توجه ویژه دارند، هزینه کلی می‌باشد که بر اساس رابطه (۹) محاسبه می‌شود [۳۹]:

$$EC_{ij}=(\frac{l_i}{n_j})\times RC+(\frac{f_i}{b_j})\times TC \qquad (۹)$$

که در آن، RC هزینه منابع، f_i اندازه فایل و TC هزینه انتقال را نشان می‌دهد.

[2] Expected round trip

انرژی مصرف‌شده[1]: پارامتر سومی که پژوهشگران ابر به آن می‌پردازند کاهش مصرف انرژی و آلاینده‌های سوختی در محیط زیست است. مصرف انرژی را می‌توان با استفاده از رابطه (۱۰) تعریف کرد [۴۰]:

$$\mathrm{Cost}_{Pow} = k \times P_{\max} + (1-k) \times P_{\max} \times U \qquad (۱۰)$$

که در آن، U استفاده از پردازنده را نشان می‌دهد و k کسری از توان مصرف‌شده در وضعیت بیکار را نشان می‌دهد. علاوه‌بر این، P_{max} توان مصرفی در حداکثر بار را نشان می‌دهد.

بار[2]: بار کاری یک ماشین به‌صورت میانگین بار کلیه پردازنده‌هایش بر اساس رابطه (۱۱) محاسبه می‌گردد [۳۳]:

$$Load = \frac{\sum_{i=1}^{n} load(\mathrm{Pr}ocessor\_Element_i)}{n} \qquad (۱۱)$$

که در رابطه (۱۱)، n نشان‌دهنده تعداد پردازنده‌ها می‌باشد و بار هر پردازنده با رابطه (۱۲) به‌دست می‌آید.

$$load(\mathrm{Pr}ocessor\_Element_i) = \frac{(Total\_MIPS) - (Free\_MIPS)}{Total\_MIPS} \qquad (۱۲)$$

در رابطه (۱۲)، مشخص است که سرعت پردازنده از مشخصه معماری سخت‌افزار می‌باشد که بر حسب دستورالعمل‌های میلیونی در ثانیه[3] (*MIPS*) نمایش داده می‌شود.

امنیت[4]: امروزه امنیت نقش مهمی در محیط‌های محاسبات ابری ایفا می‌کند، بنابراین، اختصاص خدمات امنیتی به‌کاربران امری حیاتی است. کارها در هنگام اجرا در محیط ابری همواره مورد تهدید حملات متفاوتی‌اند، سه حمله مهم و متداول در محیط ابری عبارت‌اند از تجسس[5]، تغییر[6] و جعل[7]. به‌منظور فائق آمدن بر این حملات، خدمات امنیتی مانند، محرمانه کردن[8]، جامعیت بخشیدن[9] و احراز هویت[10] به‌کار گرفته شده است. کاربران برای کارهای خود نوع الگوریتم امنیتی را با آرایه‌ای شامل سه سلول $sd_i = [sd_i^1, sd_i^2, sd_i^3]$ مشخص می‌کنند و سطح امنیتی هر یک از ارائه‌دهنده‌های خدمت توسط آرایه سه‌گانه‌ای دیگر $sl_i = [sl_i^1, sl_i^2, sl_i^3]$ مشخص می‌شود. به‌عنوان مثال، sl_i^1 نشان‌دهنده سطح امنیتی تضمین‌شده برای خدمت محرمانه کردن برای کار t_i در نظر گرفته شده است. جدول (۳)، جدول (۴) و جدول (۵) به ترتیب تعدادی از الگوریتم‌های رمزنگاری (برای خدمت محرمانه کردن)، درهم‌سازی (برای خدمت جامعیت بخشیدن) و احراز هویت به همراه سطح امنیتی آن‌ها را نمایش می‌دهد، این سطح امنیتی بازه‌ای بین ۰/۰۸ تا ۱ دارد. که شماره یک نشان‌دهنده قدرتمندترین الگوریتم است.

این الگوریتم‌های امنیتی اغلب حاوی سربار محاسباتی[11] می‌باشند که عموماً وابستگی نزدیکی به‌اندازه داده و سطح امنیتی آن الگوریتم دارد، سربار ناشی از جامعیت بخشیدن و محرمانه کردن توسط رابطه (۱۳) محاسبه می‌شود [۲۶]:

$$SO^{lv}(t_i) = F(sd_i^{lv}, d_i^{lv}),\ lv \in \{auth, ineg, confi\} \qquad (۱۳)$$

که در آن، sd_i^{lv} نشان‌دهنده سطح درخواستی خدمت امنیتی و d_i^{lv} نشان‌دهنده اندازه کار t_i است. سپس سربار امنیتی کلی توسط معادله (۱۴) محاسبه می‌شود [۲۶].

$$SO(t_i) = \sum_{lv \in \{auth, iteg, confi\}} SO^{lv}(t_i) \qquad (۱۴)$$

بنابراین، مشخص شد که کاربران محیط ابری می‌توانند سطح امنیتی برای اجرای کار خود را تعیین کنند. با توجه به این موضوع ارزیابی امنیت ارائه‌دهنده خدمت نیز مهم می‌شود. یک روش برای ارزیابی این مورد، محاسبه نرخ خطر یا خطر برای اجرای کارها مهم است.

جدول (۳): روش‌های رمزنگاری [۲۶].

نام روش	سطح امنیت
SEAL	۰/۰۸
RC4	۰/۱۴
Blowfish	۰/۳۶
Knufu/Khafre	۰/۴۰
RC5	۰/۴۶
Rijndael	۰/۶۴
DES	۰/۹۰
IDEA	۱

جدول (۴): روش‌های درهم‌سازی [۲۶].

نام روش	سطح امنیت
MD4	۰/۱۸
MD5	۰/۲۶
RIPEMD	۰/۳۶
RIPEMD-128	۰/۴۵
SHA-1	۰/۶۳
RIPEMD-160	۰/۷۷
Tiger	۱

جدول (۵): روش‌های احراز هویت [۲۶].

نام روش	سطح امنیت
HMAC-MD5	۰/۵۵
HMAC-SHA-1	۰/۹۱
CBC-MAC-AES	۱

[1] Energy consumption
[2] Load
[3] Millions Instruction Per Second (MIPS)
[4] Security
[5] Snooping
[6] Alteration
[7] Spoofing
[8] Confidentiality
[9] Integrity
[10] Authentication
[11] Computation overhead

در این مقاله از توزیع پوآسون برای آنالیز نرخ خطر استفاده می‌شود. نرخ خطر یک کار با یک ارائه‌دهنده خدمت با معادله (۱۵) قابل محاسبه است [۲۶]:

$$RR(t_i, sl_i^l) = 1 - e(-(sd_i^v, sl_i^{lv})) \quad lv \in \{auth, \text{int}eg, confi\} \tag{۱۵}$$

با بالاتر رفتن تفاوت مابین سطح امنیتی درخواستی کاربر برای کارش و سطح تعریف شده برای ارائه‌دهنده خدمت، احتمال شکست در اجرای آن کار افزایش می‌یابد. نرخ خطر برای کار t_i توسط معادله (۱۶) محاسبه می‌شود [۲۶].

$$RR(t_i) = 1 - \prod_{lv \in \{auth, \text{int}eg, confi\}} (1 - RR(t_i, sl_i^l)) \tag{۱۶}$$

و سپس نرخ خطر برای کارهای تخصیص داده شده به ارائه‌دهنده خدمت (ماشین مجازی) j توسط معادله (۱۷) به‌دست می‌آید. با داشتن نرخ خطر هر یک از ارائه‌دهنده خدمت، نرخ خطر برای ذره iام در جمعیت با معادله (۱۸) قابل محاسبه است. در این مقاله به دنبال کم‌ترین خطر برای اجرای کارها هستیم.

$$RR(VM_j) = \sum_{\forall t_i \in VM_j} RR(t_i) \tag{۱۷}$$

$$RR(P_i) = \sum_{j=1}^{vn} RR(VM_j) \tag{۱۸}$$

که در آن، vn تعداد کل ماشین‌های مجازی است.

۵-۳- به‌روزرسانی ذرات

در الگوریتم PSO استاندارد، عملیات به‌روزرسانی به‌گونه‌ای است که تنوع در جمعیت تا حدی از بین می‌رود و احتمال افتادن در بهینه محلی افزایش می‌یابد. برای حل این مسئله دو نوع به‌روزرسانی برای دو نوع مختلف از ذرات بدون استفاده از هیچ بردار سرعتی پیشنهاد شده است. همان‌طور که در بخش ۴ گفته شد ذرات در زیرجمعیت‌ها به دو گونه تقسیم‌بندی شدند، ذرات معمولی و بهترین ذره محلی، ذرات معمولی در یک زیرجمعیت موظف به انجام عملیات بهره‌برداری در فضای مسئله و توسعه تنوع جمعیت است. به‌روزرسانی برای ذرات معمولی با معادله (۱۹) انجام می‌شود [۳۸].

$$xc_i^d = \omega x_i^d + c_1 rand_1^d (pBest_i^d - x_i^d) + c_2 rand_2^d (cgBest_c^d - x_i^d) \tag{۱۹}$$

در معادله (۱۹)، ω وزن اینرسی، ثابت‌های c_1 و c_2 ثابت‌های شتاب، $rand_1$ و $rand_2$ دو عدد تصادفی بین صفر و یک و در نهایت $cgBest_c^d$ بهترین ذره محلی در زیرجمعیت c است.

از طرفی بهترین ذره محلی در هر زیرجمعیت موظف به یاددهی و اثرگذاری بر روی ذرات معمولی است، در کنار این امر، مسئولیت اکتشاف اطلاعات از دیگر زیرجمعیت‌ها را نیز بر عهده دارد که این کار موجب افزایش تنوع جمعیت می‌شود. چگونگی به‌روزرسانی برای بهترین ذرات محلی در معادله (۲۰) قابل مشاهده است [۳۸].

$$xg_i^d = \omega x_i^d + c_1 rand_1^d (pBest_i^d - x_i^d) + c_2 rand_2^d \left(\frac{1}{C}\sum_{c=1}^{C} cgBest_c^d - x_i^d\right) \tag{۲۰}$$

که در آن، C نمایانگر تعداد زیرجمعیت‌ها است. می‌توان مشاهده کرد که $cgBest_c^d$ واقع در معادله (۲۰) با میانگین اطلاعات برآمده از تمام زیرجمعیت‌ها جایگزین شده است. این جایگذاری نقش مؤثری در بهبود تنوع در جمعیت و متعاقب آن افزایش سرعت همگرایی[1] دارد، به‌دلیل اینکه اطلاعات همه زیرجمعیت‌ها در میان آن‌ها به اشتراک گذاشته می‌شود.

پارامتر مشترک در معادله (۱۹) و معادله (۲۰)، وزن اینرسی می‌باشد که در محیط‌های متفاوت، با تنظیم‌های متفاوت، ذرات متفاوتی را ایجاد کند. بنابراین سهم عمده‌ای در PSO و در تنظیم دو عملیات اکتشاف و بهره‌برداری دارد. به‌طور معمول نسخه خطی[2] وزن اینرسی در مسائل مورداستفاده قرار می‌گیرد و این در حالی است که نسخه غیرخطی[3] آن قوی‌تر و دارای توانایی تنظیم‌کنندگی بالاتری است. در سال‌های گذشته، توابع آشوب[4] به‌عنوان یکی از نگاشت‌های غیرخطی که توانایی تولید عددهای تصادفی و نامنظم را دارند مطرح شده‌اند و به‌صورت گسترده در عرصه محاسبات تکاملی مورد استفاده قرار گرفته‌اند. یکی از شناخته‌شده‌ترین این نوع توابع، تابع منطقی است که توانایی تولید عدد تصادفی مابین صفر و یک را دارد.

چگونگی تولید اعداد تصادفی توسط این تابع در معادله (۲۱) قابل‌مشاهده است [۴۱].

$$r(t+1) = 4r(t)(1-r(t)), \quad r(0) = rand, \quad r_0 \notin \{0, 0.25, 0.5, 0.75, 1\} \tag{۲۱}$$

در این مقاله، از تابع آشوب منطقی به‌جای وزن اینرسی معمول استفاده شده است، بنابراین یک وزن اینرسی غیرخطی داده شده است.

معادله (۲۲) نشان‌دهنده وزن اینرسی غیرخطی است [۴۱].

$$\omega(t) = r(t).\omega_{\min} + \frac{(\omega_{\max} - \omega_{\min}).t}{T_{\max}} \tag{۲۲}$$

که در معادله (۲۲)، $\omega_{\max} = 0.9$، $\omega_{\max} = 0.4$ و $r(t)$ عدد تصادفی تولیدشده توسط تابع منطقی است.

[1] Convergence speed
[2] Linear
[3] Non-linear
[4] Chaotic functions

۵-۴- تأمین امنیت برای کارهای پر خطر

برای تأمین امنیت کارهای پرخطر بعد از مشخص شدن بهترین ذره و مناسب‌ترین ماشین مجازی برای اجرای هر کار، ابتدا نرخ خطر اجرای کارها توسط ماشین مجازی با توجه به سطح امنیتی درخواستی و سطح امنیتی که ماشین مجازی می‌تواند فراهم کند محاسبه می‌شود (معادلات ۱۵-۱۸)، اگر سطح امنیتی ماشین مجازی بیش از سطح امنیتی درخواستی یک کار بود، نرخ خطر صفر محاسبه می‌شود به‌غیراز این مورد کارها بر اساس نرخ خطر اجرا از زیاد به کم مرتب می‌شوند و کارهای پرخطر مشخص می‌شوند. وقتی حمله‌ای غیرقابل پیش‌بینی در محیط ابری اتفاق می‌افتد مقرون‌به‌صرفه نیست اجرای تمامی کارها را متوقف کرده و آن‌ها را به ماشین مجازی ایمن‌تر منتقل کنند، از طرفی اگر ارائه‌دهنده خدمت ابری پاسخ مناسبی در جواب حمله پیش‌آمده تعبیه نکرده باشد این موضوع می‌تواند اثر منفی بر خدمت ابری وارد کند. راهبرد پاسخ به حملات را می‌توان در قالب یک عملیات زمان‌بندی دوباره تعریف کرد. فرض بر این است که مسئول خدمت ابری حرکت خرابکارانه و غیرعادی در ماشین مجازی که در حال اجرای کارهایی است را شناسایی می‌کند، به‌منظور پاسخ به این رفتار خرابکارانه، کارها به دو دسته تقسیم می‌شوند: کارهای تکمیل‌شده و کارهای در حال اجرا. برای کارهای تکمیل‌شده راهبرد امنیتی وجود ندارد. اما برای کاهش تهدیداتی که یک کار آلوده می‌تواند در قبال دیگر کارها در یک ماشین مجازی داشته باشد (ایجاد تغییرات نا به هنجار یا از دسترس خارج کردن) کارهای مشخص‌شده با خطر بالا در ماشین‌های مجازی دیگر دوباره زمان‌بندی شده تا اجرا شوند. این زمان‌بندی دوباره با مهاجرت کارهای پرخطر و با توجه به پارامتر هزینه (معادله ۹) انجام می‌شود؛ بنابراین کارهایی که احتمال آلوده شدن فراوانی دارند (کارهای پرخطر) به ماشین مجازی امن‌تر منتقل می‌شوند. با این انتقال نرخ خطر اجرای کارها در ماشین‌های مجازی جدید دوباره محاسبه شده و امنیت آن‌ها مورد ارزیابی قرار می‌گیرد.

۶- نتایج

در این بخش، الگوریتم پیشنهادی بهینه‌ساز ذرات با یادگیری انطباقی چندگانه (MAPSO) در دو قسمت مورد ارزیابی قرار گرفته است. در قسمت اول، عملکرد این الگوریتم بر روی مجموعه توابع تست استاندارد cec 2017 به‌دست آمده و با الگوریتم‌های PSO، CPSO، PSO-ALS و BAT در دو معیار میانگین و انحراف معیار مورد مقایسه قرار گرفته است. در قسمت دوم، الگوریتم زمان‌بندی پیشنهادی که مبتنی بر MAPSO است با چهار الگوریتم زمان‌بندی مختلف (CJS, OTSS, GTSA, JSSS) مورد مقایسه قرار گرفته است.

۶-۱- عملکرد بر روی مجموعه توابع cec2017

توابع تست cec2017، مجموعه‌ای از ۳۰ تابع با انواع مختلف از جمله تک‌نمایی، چندنمایی ساده، مخلوط و مرکب می‌باشند. مجموعه‌ای ۱۲تایی انتخاب‌شده از این توابع که برای ارزیابی الگوریتم پیشنهادی استفاده شده است در جدول (۶) قابل مشاهده است. برای همه این ۱۲ تابع فضای جستجوی مابین ۱۰۰- و ۱۰۰ در نظر گرفته شده است.

جدول (۶): جزئیات دوازده تابع cec2017 مورداستفاده [۴۲].

نوع توابع	شماره	توضیحات	بهترین جواب
تک‌نمایی	۱	Shifted and Rotated Bent Cigar Function	۱۰۰
	۲	Shifted and Rotated Zakharov Function	۳۰۰
چندنمایی ساده	۳	Shifted and Rotated Rosenbrock's Function	۴۰۰
	۴	Shifted and Rotated Expanded Scaffer's Function	۶۰۰
	۵	Shifted and Rotated Rastrigin's Function	۸۰۰
مخلوط	۶	Hybrid Function 1 (N=3)	۱۱۰۰
	۷	Hybrid Function 4 (N=4)	۱۴۰۰
	۸	Hybrid Function 6 (N=5)	۱۷۰۰
مرکب	۹	Composition Function 2 (N=3)	۲۲۰۰
	۱۰	Composition Function 4 (N=4)	۲۴۰۰
	۱۱	Composition Function 6 (N=5)	۲۶۰۰
	۱۲	Composition Function 8 (N=6)	۲۸۰۰

توابع تک‌نمایی دارای مسیرهای پیوسته و هموار و درعین‌حال دارای پستی‌های باریک هستند. توابع چندگانه دارای تعداد زیادی بهینه محلی است که فاصله آن‌ها با بهینه کلی زیاد است. برای توابع مخلوط، متغیرها به‌صورت تصادفی به چند زیرمجموعه تقسیم شده‌اند سپس برای زیرمجموعه‌های متفاوت، تعدادی توابع پایه مثل (Rastrigin و Rosenbrock) مورداستفاده قرار می‌گیرد. توابع مرکب نیز ترکیب چند زیر تابع هستند که به‌صورت معمول پیدا کردن بهینه کلی در آن‌ها زمان‌بر است.

عملکرد ۵ روش مورد مقایسه را از نظر میانگین و انحراف معیار در جدول (۷) می‌توان مشاهده کرد، در ۸ مورد از این توابع شامل ۲ تابع مرکب (شماره ۱۱ و ۱۲)، ۲ تابع مخلوط (شماره ۶ و ۷) دو تابع چندنمایی ساده (شماره ۳ و ۵) و ۲ تابع تک نمایی (شماره ۱ و ۲) روش پیشنهادی عملکرد بهتری داشته است. در ۴ تابع دیگر (شماره‌های ۴، ۸، ۹ و ۱۰) هم عملکرد قابل قبولی در مقایسه با دیگر روش‌ها از خود به نمایش گذاشته است. با توجه به جدول (۷)، الگوریتم پیشنهادی به‌صورت میانگین ۲ برابر بهتر

از الگوریتم خفاش عمل کرده است که یکی از دلایل برتری این است که برخلاف الگوریتم خفاش که بیشتر وقت خود را معطوف به عملیات بهره‌برداری می‌کند و از وجود ابزاری برای رهایی از بهینه محلی بی‌نصیب است، الگوریتم پیشنهادی با وجود چندین زیرجمعیت علاوه بر عملیات بهره‌برداری، به‌خوبی فضای مسئله را برای شناسایی بهینه کلی پیمایش می‌کند. در مقایسه با الگوریتم‌های CPSO و PSO-ALS روش پیشنهادی به ترتیب ۱/۵ برابر و ۱۲ درصد بهتر عمل کرده است.

جدول (۷): نتایج عملکرد روش‌های مورد مقایسه بر روی cec2017.

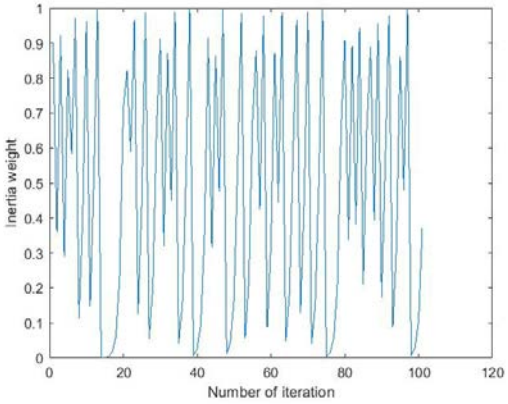
	PSO		MAPSO		CPSO		BAT		PSO-ALS	
تابع	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین	انحراف معیار	میانگین
1	1.62E+06	3.33E+06	1.05E+05	**2.31E+05**	3.11E+05	3.18E+06	1.21E+07	4.28E+08	2.11E+05	2.55E+05
2	2.11E+02	5.11E+02	2.18E+01	**3.17E+02**	2.39E+02	4.06E+02	1.62E+02	5.08E+02	1.69E+01	3.75E+02
3	3.81E+02	4.24E+03	1.83E+02	**4.72E+02**	3.76E+01	4.97E+02	3.41E+03	4.76E+04	2.35E+02	5.34E+02
4	1.65E+05	8.65E+05	3.35E+01	6.83E+02	5.65E+02	7.12E+02	2.88E+02	8.43E+03	3.00E+01	**6.47E+02**
5	3.33E+01	9.65E+02	2.73E+02	**8.54E+02**	4.83E+02	9.11E+02	3.22E+02	1.08E+03	3.65E+02	8.80E+02
6	3.28E+02	15.12E+02	5.83e+02	**11.55E+02**	7.43E+01	12.00E+02	4.25E+03	14.65E+02	2.54E+03	11.89E+02
7	8.76E+04	44.85E+04	7.11E+02	**32.81E+02**	4.00E+02	34.98E+02	1.76E+05	3.971E+05	7.54E+03	33.42E+02
8	2.21E+02	**18.33E+02**	4.69E+01	19.65E+02	3.61E+02	20.75E+02	3.39E+03	21.05E+02	4.45E+02	18.67E+02
9	3.16E+03	27.34E+02	1.18E+03	24.13E+02	4.42E+03	26.54E+02	8.88E+04	34.85E+03	6.88E+03	**23.75E+02**
10	9.23E+03	30.09E+02	3.12E+03	27.08E+02	4.65E+03	**26.81E+02**	2.75E+03	31.18E+02	5.95E+02	27.58E+02
11	2.19E+03	34.76E+02	3.05E+03	**29.18E+02**	1.77E+03	30.83E+02	8.66E+03	40.05E+02	5.43E+03	30.18E+02
12	6.31E+02	41.77E+02	5.03E+03	**30.42E+02**	8.11E+03	31.07E+02	6.71E+03	41.35E+02	1.91E+02	31.75E+02

یکی از دلایل پیمایش بهتر فضای مسئله توسط الگوریتم پیشنهادی تنظیم وزن اینرسی با تابع آشوب در دورهای مختلف است. شکل (۸) و شکل (۹) به ترتیب نمایانگر وزن اینرسی به‌دست‌آمده توسط الگوریتم پیشنهادی (MAPSO) و PSO است. همان‌طور که مشخص است تابع آشوب استفاده‌شده موجب تولید اعداد متفاوت و غیرخطی برای وزن اینرسی الگوریتم پیشنهادی شده که این امر در هنگام به‌روزرسانی ذرات موجب ایجاد تنوع بهتری نسبت به PSO می‌شود.

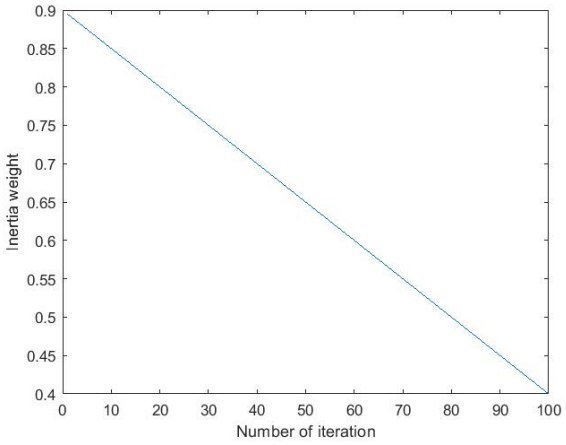
برای نشان دادن بهتر عملکرد روش پیشنهادی بر روی توابع تست cec2017 علاوه بر محاسبه میانگین و انحراف معیار برای این روش و مقایسه آن با دیگر روش‌های مشابه (جدول (۷))، از یک آزمون نا پارامتری به نام ویلکاکسون رتبه علامت‌دار استفاده شده است. این آزمون نا پارامتری به‌منظور انجام بررسی دو نمونه وابسته یا انطباق بین دو نمونه و همچنین مشخص کردن تفاوت آماری بین دو مجموعه از راه‌حل‌ها به‌کار گرفته می‌شود. نتیجه به‌دست‌آمده از آزمون ویلکاکسون پارامتری به نام P-value است که سطح برتری الگوریتم‌ها را ارزیابی می‌کند. در این مقاله بررسی میزان بهبود میانگین جواب نهایی در الگوریتم‌های فرااکتشافی مورد مقایسه در طی ۳۰ اجرا پرداخته شده است. فرضیه در این مقاله به قرار زیر است:

به نظر می‌رسد میانگین بهترین راه‌حل برای دو الگوریتم فرااکتشافی "الف" و الگوریتم فرااکتشافی "ب" در طی ۳۰ اجرا متفاوت نباشد".

برای آزمون این فرضیه، در گام اول بهبود میانگین در اجراهای متفاوت برای هر الگوریتم به‌صورت ۱ تا ۵ رتبه‌دهی می‌شود.



شکل (۸): وزن اینرسی به‌دست‌آمده با تابع آشوب.



شکل (۹): وزن اینرسی خطی.

در جدول (۸) میانگین رتبه هر الگوریتم قابل‌مشاهده است.

همان گونه که در جدول (۸) قابل مشاهده است به نظر می‌رسد الگوریتم پیشنهادی عملکرد بهتری نسبت به سایر

الگوریتم‌های مشابه دارد، برای اطمینان از وجود تفاوت معنادار بین الگوریتم پیشنهادی و سایر الگوریتم‌ها مقادیر P-value الگوریتم پیشنهادی در برابر سایر الگوریتم‌ها برای هر تابع مورد محاسبه قرار گرفته است. در جدول (۹) این مقادیر قابل مشاهده است. از جدول (۹) با اطمینان ۹۵ درصد می‌توان نتیجه گرفت برای توابع ۱-۳، ۵-۷ و ۱۱-۱۲ بین میانگین جواب نهایی الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها در طی ۳۰ اجرا تفاوت معنی‌داری وجود دارد. اما این قطعیت وجود تفاوت برای تابع ۴ و ۹ در مقایسه با PSO-ALS، تابع ۸ در مقایسه با PSO و تابع ۱۰ در مقایسه با CPSO وجود ندارد.

جدول (۸): رتبه هر یک از الگوریتم‌های مورد مقایسه در طی ۳۰ بار اجرای هر یک از توابع.

روش / تابع	PSO	MAPSO	CPSO	PSO-ALS	BAT
۱	۳	۱	۴	۲	۵
۲	۵	۱	۳	۲	۴
۳	۴	۱	۲	۳	۵
۴	۵	۲	۳	۱	۴
۵	۶	۱	۳	۲	۵
۶	۷	۱	۳	۲	۴
۷	۸	۱	۳	۲	۴
۸	۹	۳	۴	۲	۵
۹	۴	۲	۳	۱	۵
۱۰	۴	۲	۱	۳	۵
۱۱	۴	۱	۳	۲	۵
۱۲	۵	۱	۲	۳	۴
میانگین رتبه	۴/۰۸	۱/۴۱	۲/۸۳	۲/۰۸	۴/۵۸

جدول (۹): مقدار P-value به‌دست‌آمده از توابع مختلف توسط MAPSO در مقایسه با الگوریتم‌های مشابه.

روش / تابع	PSO	CPSO	PSO-ALS	BAT
۱	2.31E-09	2.08E-07	4.75E-06	4.06E-11
۲	2.28E-10	1.21E-06	2.78E-02	3.42E-08
۳	3.01E-11	7.11E-06	5.31E-03	3.02E-11
۴	3.02E-11	6.08E-04	2.21E-01	1.82E-07
۵	4.11E-08	6.81E-04	5.27E-04	3.02E-11
۶	3.14E-11	2.44E-06	7.85E-02	8.81E-10
۷	3.69E-11	2.91E-09	1.75E-05	1.75E-11
۸	4.41E-01	3.77E-03	1.77E-03	5.16E-04
۹	3.18E-07	4.98E-05	8.27E-01	4.36E-10
۱۰	1.68E-09	4.29E-01	5.47E-03	3.03E-11
۱۱	1.38E-09	5.08E-04	2.20E-03	3.38E-11
۱۲	3.02E-11	5.66E-07	1.69E-07	1.93E-10

می‌توان از جدول‌های (۱۰-۷) این‌طور نتیجه گرفت که به تنهایی تقسیم‌بندی جمعیت کلی به چند زیرجمعیت یا تغییر وزن اینرسی از خطی به غیرخطی نمی‌تواند باعث ایجاد توازن بین عملیات‌های اکتشاف و بهره‌برداری شده و بهینه کلی را از جستجوی فضای مسئله نتیجه دهد. در الگوریتم پیشنهادی، هر زیرجمعیت تلاش به یافتن بهینه‌های محلی در قسمت‌های مختلف فضای مسئله کرده، سپس با به‌روزرسانی ذرات به کمک بهینه‌های محلی یافت شده و با استفاده از وزن اینرسی تغییریافته تلاش می‌شود همسایگان آن‌ها مورد ارزیابی قرار گرفته تا بهینه کلی پیدا شود.

به‌منظور ارزیابی سرعت اجرای هر الگوریتم، پیچیدگی هر یک در جدول (۱۰) مورد ارزیابی قرار گرفته است. پیچیدگی الگوریتم‌ها طبق روش و گام‌های چالش [۴۳] محاسبه می‌شود. در این راستا پیچیدگی الگوریتم‌های مورد مقایسه در جدول (۱۰) نشان داده شده است. الگوریتم‌های اشاره‌شده با استفاده از متلب ۲۰۱۸ نوشته شده و بر روی سیستمی با پردازنده Intel CPU (3.40GHz) و ۸ GB حافظه اجرا شده‌اند.

جدول (۱۰)، نمایانگر زمان اجرای برای قطعه کد زیر است [۴۴]:

الگوریتم (۱): آنالیز تجربی پیچیدگی

```
for (i=1; i++; i=100000)
{
 X=X+X; X=X/2; X=X*X;
 X=sqrt(X); X=log(X);
 X= exp(X); X=X/(X+2);
}
```

در الگوریتم (۱)، T_1 نمایانگر زمان مورد نیاز برای محاسبه ۲۰۰ هزار بار انجام تابع شماره ۸ است. T_2 نمایانگر زمان کامل اجرای هر الگوریتم بر روی تابع ۸ در طی ۲۰۰ هزار دور است. T_2 در طی ۵ بار اجرا برای هر الگوریتم محاسبه شده و میانگین آن‌ها در جدول (۱۰) با نماد $\hat{T}_2$ قابل مشاهده است.

جدول (۱۰): پیچیدگی به‌دست آمده برای الگوریتم‌های مختلف.

Method	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2 - T_1)/T_0$
PSO	۰/۰۹۵۵	۰/۶۷۲۸	۱/۸۸۵۵	۱۲/۶۹
MAPSO	۰/۰۹۵۵	۰/۶۷۲۸	۲/۵۱۹۲	۱۹/۳۳
CPSO	۰/۰۹۵۵	۰/۶۷۲۸	۲/۱۸۷۵	۱۵/۸۶
PSO-ALS	۰/۰۹۵۵	۰/۶۷۲۸	۲/۳۱۶۷	۱۷/۲۱
BAT	۰/۰۹۵۵	۰/۶۷۲۸	۱/۶۶۸۳	۱۰/۱۵

در جدول (۱۰) سرعت بیشتر PSO و BAT در انجام محاسبات و رسیدن به جواب بهینه تابع شماره ۸ قابل نتیجه گیری است. اما در جدول‌های (۹-۷) همگرایی زودهنگام این الگوریتم‌ها و گیرکردن در بهینه محلی قابل ملاحظه می‌شود، این پیچیدگی برای الگوریتم پیشنهادی به‌واسطه تشکیل زیرجمعیت‌ها و محاسبه تابع آشوب مقدار بالاتری دارد اما با این وجود نتیجه قابل رقابتی در مقایسه با دیگر الگوریتم‌ها به‌دست آورده است.

بنابراین، می‌توان گفت همواره الگوریتم سریع‌تر و با پیچیدگی کمتر جواب بهینه را برنمی‌گرداند، اما این امکان وجود دارد که با اضافه کردن عملیات‌هایی به این نوع الگوریتم‌ها و بالا بردن پیچیدگی به حد معقول، دقت الگوریتم را افزایش داده و به جواب بهینه‌تری رسید.

۶-۲- معماری شبیه‌ساز کلودسیم

برای ارزیابی الگوریتم‌های تکرار داده از متلب و شبیه‌ساز کلودسیم استفاده شده است که مبتنی بر زبان جاوا است و یک ابزار متن‌باز است که در محیط‌های لینوکس و ویندوز قابل اجراست. شکل (۱۰) لایه‌های شبیه‌ساز کلودسیم را نشان می‌دهد.

۶-۲-۱- تنظیمات شبیه‌ساز

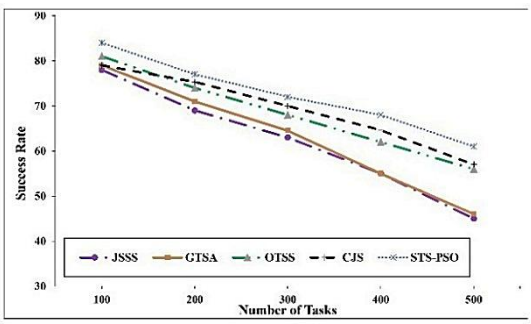
پارامترهای اصلی شبیه‌سازی و مقادیر آن‌ها در جدول (۱۱) نشان داده شده‌اند. به‌عنوان مثال تعداد مراکز داده‌ها برابر ۴۰ و تعداد پردازنده در هر ماشین ۶-۱ در نظر گرفته شده است.

شکل (۱۰): لایه‌های شبیه‌ساز کلودسیم.

جدول (۱۱): مقادیر پارامترهای شبیه‌سازی.

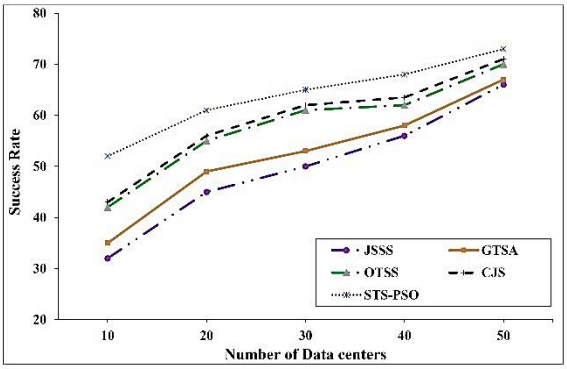
پارامترها	مقادیر
تعداد مراکز داده	۴۰
تعداد ماشین‌های مجازی	۷۰
توان پردازشی	(MIPS)۴۰۰-۲۵۰۰
تعداد پردازنده در هر ماشین مجازی	۱-۶
حافظه ماشین مجازی	(MB) ۵۱۲-۲۰۴۸
تعداد کارها	۱۰۰-۵۰۰
اندازه کارها	۱۰۰-۵۰۰

در مرحله اول، نرخ موفقیت[1] را که به‌عنوان نسبت تعداد کار اجراشده با موفقیت به کل کار ارسالی به سیستم تعریف می‌شود، مطالعه شده است. شکل (۱۱) نرخ موفقیت در تعداد مختلف کار را نشان می‌دهد. همان‌طور که مشاهده می‌شود الگوریتم STS-PSO ازآنجاکه پارامترهای امنیتی و هزینه‌ای را هم‌زمان در نظر می‌گیرد، به موفقیت بالاتری در سایر الگوریتم‌های زمان‌بندی می‌رسد. شکل (۱۲) نشان‌دهنده نرخ موفقیت تعداد مختلف مراکز داده است. بدیهی است که وقتی تعداد مرکز داده بیشتر باشد، منابع در سیستم کافی هستند بنابراین، همه الگوریتم‌ها عملکرد تقریباً خوبی را نشان می‌دهند. برای ۱۰ مرکز داده، الگوریتم STS-PSO نرخ موفقیت را در مقایسه با ساز و کار JSSS حدود ۴۰٪ افزایش می‌دهد.

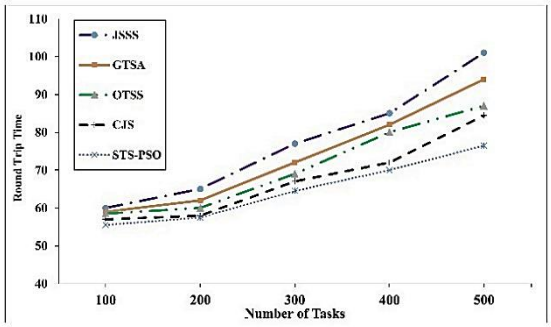


شکل (۱۱): نرخ موفقیت برای تعداد کار متفاوت.

[1] Success rate

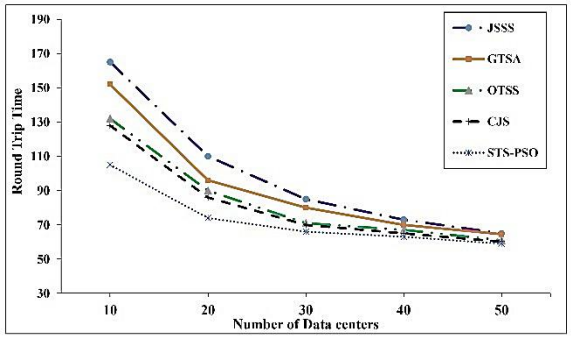


شکل (۱۲): نرخ موفقیت برای تعداد مراکز متفاوت.



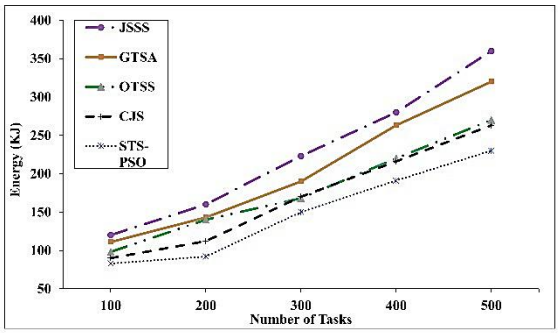
شکل (۱۳): زمان بازگشت برای تعداد کار متفاوت.

زمان بازگشت به‌عنوان متداول‌ترین معیار عملکرد ارزیابی برای الگوریتم‌های زمان‌بندی در نظر گرفته شده است که توسط رابطه (۸) محاسبه می‌گردد. به‌طور کلی، همان‌طور که از شکل (۱۳) مشاهده می‌شود، الگوریتم STS-PSO در مقایسه با OTSS، GTSA و JSSS زمان بازگشت کمتری دارد زیرا در تعریف تابع برازندگی زمان بازگشت را لحاظ کرده است. با افزایش تعداد کارها در مراکز داده، بهبود الگوریتم STS-PSO از الگوریتم‌های دیگر محسوس‌تر می‌شود. در شکل (۱۴)، زمان بازگشت روش‌های مختلف زمان‌بندی کار با تعداد زیادی از مراکز داده نشان داده شده است. از شکل (۱۴) می‌توان نتیجه گرفت که الگوریتم STS-PSO در مقایسه با الگوریتم OTSS (حدود ۱۱٪) از زمان بازگشت کمتری برخوردار است زیرا الگوریتم OTSS در فرایند توزیع کارها پارامتر زمان بازگشت را در نظر نمی‌گیرد و بعضی از کارها زمان انتظار و یا اجرای طولانی دارند. اما دلیل اصلی کارایی الگوریتم STS-PSO این است که کارها را بر اساس شرایط سیستم مانند پهنای باند، تأخیر و طول کارها زمان‌بندی می‌کند. بنابراین، منابع قدرتمند را برای کارهایی با نیاز به محاسبه کم هدر نمی‌دهد. قابل ذکر است که در شکل (۱۴)، CJS در مقایسه با JSSS حدود ۱۴٪ را کاهش می‌دهد. دلیل این امر این است که CJS کارها را از نظر داده محور بودن و یا محاسبه محور بودن در هنگام زمان‌بندی در نظر می‌گیرد.



شکل (۱۴): زمان بازگشت برای تعداد مراکز داده متفاوت.

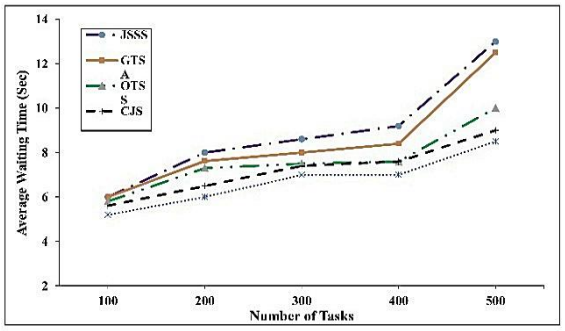
شکل (۱۵) کل مصرف انرژی را برای تعداد مختلف کار نشان می‌دهد که می‌توان دو نتیجه را مشاهده کرد: ۱) مصرف انرژی با افزایش تعداد کارها افزایش می‌یابد و ۲) الگوریتم زمان‌بندی پیشنهادی، مصرف انرژی کمتری را در مقایسه با سایر روش‌ها نشان می‌دهد. به‌عنوان مثال، الگوریتم STS-PSO در مقایسه با CJS حدود ۶٪ مصرف انرژی را کاهش می‌دهد زیرا الگوریتم STS-PSO هنگام ارسال کار به مراکز داده، هزینه انرژی را در نظر می‌گیرد.



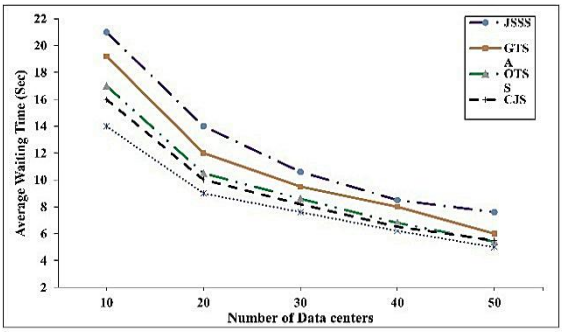
شکل (۱۵): انرژی مصرف شده برای تعداد کار مختلف.

شکل (۱۶) زمان انتظار را گزارش می‌کند که به‌عنوان زمانی که یک کار از زمان ارسال[1] تا تکمیل آن در صف انتظار می‌کشد محاسبه می‌شود. در نتایج به‌دست آمده، الگوریتم STS-PSO در مقایسه با JSSS به‌طور متوسط ۲۷٪ و در مقایسه با GTSA، زمان انتظار را به‌طور متوسط ۲۷٪ کاهش می‌دهد. این می‌تواند مربوط به توانایی STS-PSO در انتخاب مراکز داده با زمان آماده‌سازی کم باشد. شکل (۱۷) زمان انتظار الگوریتم‌های زمان‌بندی کار برای تعداد مختلف مراکز داده را نشان می‌دهد. با کاهش تعداد مراکز داده، روند افزایشی زمان انتظار توسط روش زمان‌بندی پیشنهادی به‌طور قابل‌توجهی کمتر از سایر روش‌ها است. از آنجا که الگوریتم STS-PSO زمان تأخیر را در طول فرایند زمان‌بندی در نظر می‌گیرد لذا کارها سریع‌تر اجرا می‌شوند.

[1] Submission



شکل (۱۶): زمان انتظار برای تعداد کار متفاوت.

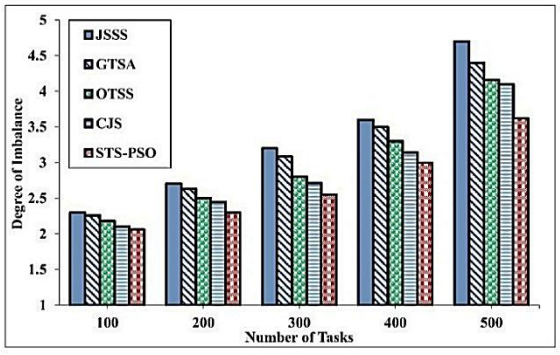


شکل (۱۷): زمان انتظار برای تعداد مراکز داده متفاوت.

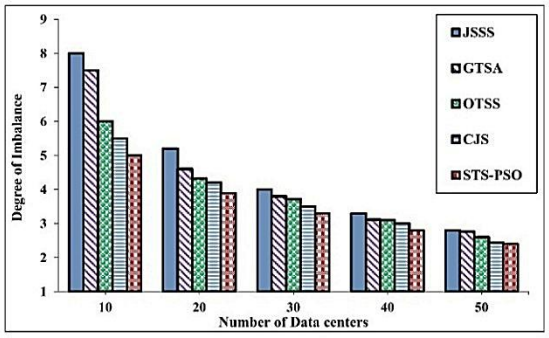
شکل (۱۸) میزان عدم تعادل بار را برای تعداد کار مختلف نشان می‌دهد که توسط معادله‌ی زیر تعریف شده است [۳۳].

$$D_i = \frac{Leng\_Tasks}{Num\_CPU \times CPU\_MIPS} \qquad (۲۳)$$

در این رابطه، $Leng\_Tasks$ طول کل کارهایی را که به VM_i اختصاص داده شده نشان می‌دهد، Num_CPU تعداد پردازنده را نشان می‌دهد و CPU_MIPS توانایی پردازنده است. طبق شکل (۱۹)، واضح است که الگوریتم پیشنهادی EAST ما به‌طور مداوم از الگوریتم‌های زمان‌بندی دیگر در پارامتر درجه عدم تعادل پیشی می‌گیرد. به‌عنوان مثال، الگوریتم STS-PSO در مقایسه با JSSSبه‌طور متوسط ۲۰٪ و در مقایسه با GTSA ۲۰٪ درجه عدم تعادل را بهبود می‌بخشد. زیرا الگوریتم STS-PSO پارامتر بار را در تابع برازندگی در نظر می‌گیرد و سعی می‌کند کارها را بر روی ماشین‌هایی با بار کم توزیع نماید.



شکل (۱۸): میزان عدم تعادل بار برای تعداد کار متفاوت.

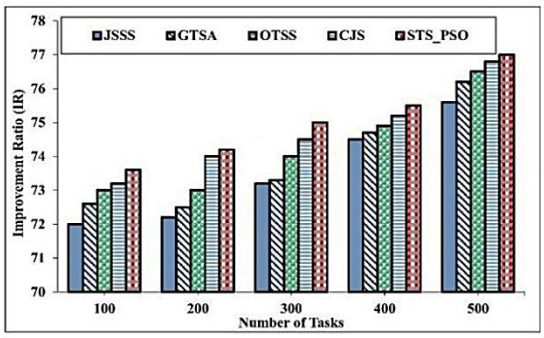


شکل (۱۹): میزان عدم تعادل بار برای تعداد مراکز داده متفاوت.

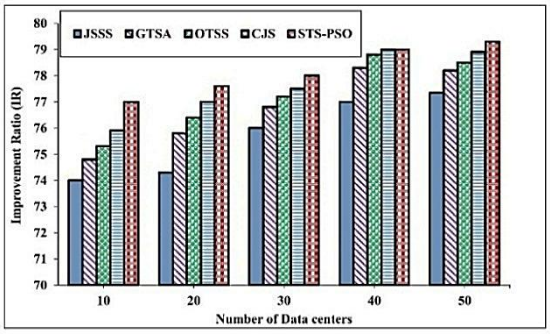
شکل (۲۰) نرخ بهبود[1] را ارائه می‌دهد که کارایی الگوریتم را بر اساس رابطه زیر نشان می‌دهد [۴۵]:

$$IR_j\,\% = \frac{\sum_{i=1,i\neq j}^{n} ExT_i - ExT_j}{\sum_{i=1,i\neq j}^{n} ExT_i} \times 100 \qquad (۲۴)$$

در این رابطه، ExT_i زمان اجرای الگوریتم i را نشان می‌دهد. از شکل (۲۰) و شکل (۲۱)، می‌توان مشاهده کرد که الگوریتم STS-PSO در مقایسه با الگوریتم‌های JSSS، GTSA و OTSS بهترین نرخ بهبود را دارد. این نتایج به این دلیل است که الگوریتم پیشنهادی با در نظر گرفتن پارامترهای مهمی مانند بار و زمان بازگشت زمان اجرای کمتری را به همراه دارد. علاوه بر این، بدیهی است که الگوریتم STS-PSO عملکرد بهتری نسبت به سایر روش‌ها در زیر بارهای سنگین به‌دست می‌آورد.



شکل (۲۰): نرخ بهبود برای تعداد کار متفاوت.



شکل (۲۱): نرخ بهبود برای تعداد مراکز متفاوت.

[1] Improvement ratio

۷- نتیجه‌گیری

در محیط رایانش ابری، امنیت داده‌ها اهمیت ویژه پیدا می‌کند زیرا داده‌ها ممکن است در مکان‌های گوناگونی از جهان قرار داشته باشند. رایانش ابری با ارائه مدل مبتنی بر تقاضا و ویژگی خدمت خودکار درخواست مشتریان و کارها را با هزینه معقولی و شیوه مدیریتی مناسب اجرا می‌نماید. از طرف دیگر، با توجه به ماهیت پویا و توزیع‌شده ابر و تعداد زیاد کارها، زمان‌بندی کار به مشکل اصلی تبدیل گشته است. در این مقاله، الگوریتم زمان‌بندی کار برای بهبود امنیت با استفاده الگوریتم بهینه‌سازی ازدحام ذرات بهبودیافته معرفی شده است. الگوریتم بهبودیافته، تنوع جمعیت را با به‌کار بردن یادگیری انطباقی افزایش می‌دهد تا تعادلی بین عملیات اکتشاف و بهره‌برداری حاصل شود. سپس الگوریتم زمان‌بندی پیشنهادی با در نظر گرفتن هم‌زمان پنج پارامتر (زمان بازگشت، بار، مصرف انرژی، هزینه و امنیت) منجر به توزیع بار و کاهش مصرف انرژی می‌گردد. الگوریتم زمان‌بندی پیشنهادی با استفاده از شبیه‌ساز کلودسیم با روش‌های مرتبط (CJS, OTSS, GTSA, JSSS) ارزیابی شد. نتایج حاصل از آزمایش‌ها حاکی از آن بود که الگوریتم پیشنهادی با در نظر گرفتن الگوریتم فرااکتشافی در حل مسئله زمان‌بندی کارها در محیط رایانش ابری موفق بوده است. اما می‌توان با استفاده از سایر روش‌ها چون نظریه بازی‌ها مسئله زمان‌بندی کار را بررسی نمود و سایر پارامترها مانند اولویت کارها را نیز لحاظ نمود.

۸- مراجع

[1] N. Mansouri and M. M. Javidi, “A review of data replication based on meta-heuristics approach in cloud computing and data grid,” Soft Comput., vol. 24, pp. 14503-14530, 2020.

[2] M. Bansal and S. K. Malik, “A multi-faceted optimization scheduling framework based on the particle swarm optimization algorithm in cloud computing,” Sustainable Comput. Inf. Syst., 2020.

[3] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[4] F. Jauro, H. Chiroma, A. Y. Gital, M. Almutairi, S. M. Abdulhamid, and J. H. Abawajy, “Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend,” Appl. Soft Comput., vol. 96, 2020.

[5] N. Mansouri and M. M. Javidi, “A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers,” J. Supercomput., vol. 74, pp. 5349-5372, 2018.

[6] N. Mansouri, R. Ghafari, and B. Mohammad Hasani Zade, “Cloud computing simulators: A comprehensive review,” Simul. Modell. Pract. Theory, vol. 104, 2020.

[7] A. A. Zubair, S. B. A. Razak, M. A. Bin Ngadi, A. Ahmed, and S. H. H. Madni, “Convergence-based task scheduling techniques in cloud computing: A review,” In: International Conference of Reliable Information and Communication Technology, pp. 227–234, 2020.

[8] S. R. Pakize, S. M. Arefi nejad, “A new scheduling algorithm to reduce computation time in Hadoop environment,” Journal of Electronical & Cyber Defence, vol. 8, 2020. (In Persian)

[9] F. Xin and L. Zhang, “The review of task scheduling in cloud computing,” In: International Conference on Geo-informatics in Sustainable Ecosystem and Society, pp. 119–126, 2019.

[10] M. Mehravaran, M. R. Pajoohan, and F. Adibnia, “Secure and confidential workflow scheduling in hybrid cloud with improved particle swarm optimization algorithm,” Journal of Electronic and cyber defense, vol. 7, pp. 131-145, 2019. (In Persian)

[11] N. Mansouri, “Network and data location aware approach for simultaneous job scheduling and data replication in large-scale data grid environments,” Front. Comput. Sci., vol. 8, pp. 391-408, 2014.

[12] 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 Gener. Comput. Syst., vol. 108, pp. 361-371, 2020.

[13] 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, pp. 48-61, 2021.

[14] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, “A comprehensive survey for scheduling techniques in cloud computing,” J. Netw. Comput. Appl., vol. 143, pp. 1-33, 2019.

[15] N. Subramanian and A. Jeyaraj, “Recent security challenges in cloud computing,” Comput. Electr. Eng., vol. 71, pp. 28-42, 2018.

[16] H. Tabrizchi and M. Kuchaki Rafsanjani, “A survey on security challenges in cloud computing: issues, threats, and solutions,” J. Supercomput., 2020.

[17] A. Bansal, A. K. Bairwa, and S. Hiranwal, “Security issues in cloud computing: A Review,” In: Proceedings of International Conference on Communication and Computational Technologies, pp. 515-521, 2021.

[18] H. Mouratidis, S. Shei, and A. Delaney, “A security requirements modelling language for cloud computing environments,” Softw. Syst. Model., vol. 19, pp. 271-295, 2020.

[19] D. A. B. Fernandes, L. F. B. Soares, J. V Gomes, M. M. Freire, and P. R. M. Inácio, “Security issues in cloud environments: a survey,” Int. J. Inf. Secur., vol. 13, pp. 113-170, 2014.

[20] S. Meng, W. Huang, X. Yin, M. R. Khosravi, Q. Li, S. Wan, and L. Qi, “Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications,” In: IEEE Transactions on Industrial Information, 2020.

[21] J. Zhou, J. Sun, P. Cong, Z. Liu, X. Zhou, T. Wei, and S. Hu, “Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT,” In: IEEE Transactions on Services Computing, 2020.

[22] R. Kumar and R. Goyal, “On cloud security requirements, threats, vulnerabilities and countermeasures: A survey,” Comput. Sci. Rev., vol. 33, pp. 1-48, 2019.

[23] N. Mansouri, G. Dastghaibyfard, and A. Horri, “A novel job scheduling algorithm for improving data grid’s performance,” In: 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 142-147, 2011.

[24] J. Gąsior and F. Seredyński, “Security-aware distributed job scheduling in cloud computing systems: A game-theoretic cellular automata-based approach,” In: International Conference on Computational Science, pp. 449-462, 2019.

[25] D. Fernández-Cerero, A. Jakóbik, D. Grzonka, J. Kołodziej, and A. Fernández-Montes, “Security supportive

[36] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," Future Gener. Comput. Syst., vol. 91, pp. 407-415, 2019.

[37] G. Xu, Q. Cui, X. Shi, H. Ge, Z. H. Zhan, H. P. Lee, Y. Liang, R. Tai, and C. Wu, "Particle swarm optimization based on dimensional learning strategy," Swarm Evol. Comput., vol. 45, pp. 33-51, 2019.

[38] Y. Zhang, X. Liu, F. Bao, J. Chi, C. Zhang, and P. Liu, "Particle swarm optimization with adaptive learning strategy," Knowledge-Based Syst., vol. 196, 2020.

[39] M. S. Sanaj and P. M. Joe Prathap, "Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere," Eng. Sci. Technol. an Int. J., vol. 23, pp. 891-902, 2020.

[40] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future Gener. Comput. Syst., vol. 28, pp. 755-768, 2012.

[41] H. Liu, X.W. Zhang, and L.P. Tu, "A modified particle swarm optimization using adaptive strategy," Expert Syst. Appl., vol. 152, 2020.

[42] I. B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems," Appl. Soft Comput., vol. 66, pp. 232-249, 2018.

[43] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report, 2017.

[44] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 145-152, 2017.

[45] N. Mansouri, "Adaptive data replication strategy in cloud computing for performance improvement," Front. Comput. Sci., vol. 10, pp. 925-935, 2016.

energy-aware scheduling and energy policies for cloud environments," J. Parallel Distrib. Comput., vol. 119, pp. 191-202, 2018.

[26] H. Y. Shishido, J. C. Estrella, C. F. M. Toledo, and M. S. Arantes, "Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds," Comput. Electr. Eng., vol. 69, pp. 378-394, 2018.

[27] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," Future Gener. Comput. Syst., vol. 65, pp. 140-152, 2016.

[28] L. Ismail and H. Materwala, "EATSVM: Energy-aware task scheduling on cloud virtual machines," Procedia Comput. Sci., vol. 135, pp. 248-258, 2018.

[29] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," J. Supercomput., vol. 60, pp. 268-280, 2012.

[30] H. Zhang, "Research on job security scheduling strategy in cloud computing model," In: International Conference on Intelligent Transportation, Big Data & Smart City, pp. 649-652, 2015.

[31] X. Liu and Y. Zhou, "A Self-adaptive layered sleep-based method for security dynamic scheduling in cloud storage," In: 4th International Conference on Information Science and Control Engineering, pp. 99-103, 2017.

[32] Y. Lou, T. Zhang, J. Yan, K. Li, Y. Jiang, H. Wang, and J. Cheng, "Dynamic scheduling strategy for testing task in cloud computing," In: Sixth International Conference on Computational Intelligence and Communication Networks, pp. 633-636, 2014.

[33] N. Mansouri and M. M. Javidi, "Cost-based job scheduling strategy in cloud computing environments," Distrib. Parallel Databases, pp. 1-36, 2019.

[34] R. Achar, P. S. Thilagam, D. Shwetha, and H. Pooja, "Optimal scheduling of computational task in cloud using virtual machine tree," In: Third International Conference on Emerging Applications of Information Technology, pp. 143-146, 2012.

[35] M. A. Kacimi, O. Guenounou, L. Brikh, F. Yahiaoui, and N. Hadid, "New mixed-coding PSO algorithm for a self-adaptive and automatic learning of Mamdani fuzzy rules," Eng. Appl. Artif. Intell., vol. 89, 2020.