دوره شانزدهم، شماره پاییز و زمستان ۱۴۰۲

مجله فناوری اطلاعات در طراحی مهندسی

Information Technology in Engineering Design

http://ited.sinaweb.net

یک مدل زمان‌بندی وظایف در منابع ناهمگن ابری با ترکیبی از الگوریتم‌های هوش جمعی

صفدر رستمی[۱] علی برومندنیا*[۲] احمد خادم‌زاده[۳]

(۱) گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد جنوب-تهران، تهران، ایران

(۲) گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد جنوب-تهران، تهران، ایران

(۳) گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد جنوب-تهران، تهران، ایران

(تاریخ دریافت :۱۴۰۲/۰۹/۲۰ تاریخ پذیرش: ۱۴۰۳/۰۲/۰۹)

چکیده

در رایانش ابری کاربران بر اساس تقاضا و پرداخت به ازای استفاده، به منابع محاسباتی اشتراکی دسترسی دارند. برای اجرای درخواست‌های متنوع و متغیر با زمان کاربران و محدودیت و پویایی منابع محاسباتی، رسیدگی به شرایط پویای سیستم، بهره‌وری منابع و رضایت کاربران، یک مکانیسم زمان‌بندی کارا، امری حیاتی می‌باشد. تخصیص وظایف به منابع، یک چالش اساسی در محیط های محاسباتی رایانش ابری به شمار می‌رود؛ بنابراین برای کاهش زمان اجرا، الگوریتم‌های مختلف با مکانیسم‌های زمان‌بندی متفاوتی ارائه شده است. تمامی الگوریتم‌های زمان‌بندی ارائه‌شده، با توجه به وضعیت فعلی سیستم و پویایی درخواست‌های کاربران، یک زمان‌بندی بهینه بین منابع و وظایف را فراهم آورند، اما بسیاری از روش‌های موجود، در طول زمان‌های طولانی نتوانسته‌اند نتیجه مطلوبی را ارائه دهند. به دلیل سرعت همگرایی پایین راه‌حل‌ها در الگوریتم‌های فرااکتشافی، روش پیشنهادی در این مقاله، الگوریتم جست‌وجوی کاپوچین چندهدفه، مبتنی بر صف‌های اولویت چندگانه و رتبه‌بندی نامغلوب است. با ایجاد یک گراف به کمک الگوریتم بهینه‌سازی پیشنهادی و یک تابع هدف زمان اجرا و زمان اتمام کل کارها، سعی در بهبود زمان‌بندی وظایف دارد. روش پیشنهادی در مقایسه با روش‌های مشابه توانسته است، بهبود چشمگیری (۲۵ درصد) در زمان اتمام کل کارها داشته باشد؛ نتایج به‌دست‌آمده نشان از برتری روش پیشنهادی دارد.

کلمات کلیدی: رایانش ابری، زمان‌بندی وظایف، رتبه‌بندی نامغلوب، صف‌های اولویت چندگانه

*عهده‌دار مکاتبات:

علی برومندنیا

نشانی: گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد جنوب-تهران، تهران، ایران

پست الکترونیکی: **yaghoobi@mshdiau.ac.ir**

۱- مقدمه

رایانش ابری مدلی برای فراهم کردن دسترسی آسان، بر اساس تقاضای کاربر، از طریق شبکه به مجموعه‌ای از منابع رایانشی قابل‌تغییر و پیکربندی (مثل شبکه‌ها، سرورها، فضای ذخیره‌سازی، برنامه‌های کاربردی و وظایف) است. این دسترسی می‌تواند با کمترین نیاز به مدیریت منابع و یا نیاز به دخالت مستقیم فراهم‌کننده سرویس به سرعت فراهم شده یا آزاد گردد. سیستم‌های رایانش ابری ناهمگن[1] نوعی از سیستم‌های رایانش توزیع‌شده هستند. در این نوع سیستم‌ها مواقعی که وظایف بر روی پردازنده‌های یکسان اجرا نمی‌شوند دارای زمان اجرا و مصرف انرژی متفاوتی هستند [۱]. سیستم‌های رایانش توزیع‌شده ناهمگن، توسط شبکه‌های پرسرعت به همدیگر متصل شده‌اند که مناسب پردازش‌های پرسرعت برنامه‌های حجیم محاسباتی گوناگون هستند [۲, ۳]. همچنین سیستم‌های رایانش توزیع‌شده اجرای موازی برنامه‌ها را ممکن می‌سازند.

الگوریتم زمان‌بندی روشی است که به وسیله‌ی آن، وظایف به منابع موجود در مراکز داده تخصیص داده می‌شوند [۴, ۵]. انتخاب یک زمان بندی نامناسب می‌تواند باعث مواردی از جمله کند شدن برنامه ابر و به دنبال آن، افزایش زمان اجرای برنامه‌ها شود [۶]. هدف اصلی الگوریتم‌های زمان‌بندی، برآوردن نیازهای چالشی سیستم‌های توزیع‌شده از جمله به‌دست‌آوردن عملکرد رایانشی بالا و بهترین توان عملیاتی سیستم است. با وجود تلاش‌های بسیاری که اخیراً در زمینه‌ی زمان‌بندی وظایف صورت گرفته است، همچنان این مسئله در محیط‌های رایانش ناهمگن یک چالش باقی مانده است. زمان‌بندی وظایف[2] در سیستم‌های توزیع‌شده، باید زیر وظایف[3] را به منابع مناسب جهت اجرا تخصیص دهد و الگوریتم‌های زمان‌بندی، ترتیب اجرای این وظایف را مشخص می‌کنند [۷]. از آنجایی که تخصیص وظایف به منابع یک چالش می‌باشد، الگوریتم‌های بسیاری جهت کاهش زمان اجرا و موازی‌سازی زیر وظایف ارائه شده است. در [۸]، یک روش زمان‌بندی وظایف، با استفاده از برنامه‌ریزی وظایف بر اساس صف‌های پویا و الگوریتم‌های ترکیبی فرااکتشافی شامل الگوریتم ازدحام ذرات-فازی و الگوریتم تبرید شبیه‌سازی شده-ازدحام ذرات، پیشنهاد شده است. هدف این رویکرد به‌دست‌آوردن بهترین ترتیب وظایف، برای به حداقل رساندن زمان انتظار در صف انتظار وظایف و توزیع وظایف در بین صف‌های پویای ایجاد شده، سپس تخصیص وظایف به مناسب‌ترین منابع و بهینه‌سازی معیارهای عملکرد ابر می‌باشد. در مقاله [۹]، از الگوریتم بهینه‌سازی ازدحام ذرات باینری با حداقل پیچیدگی زمانی و هزینه برای زمان‌بندی وظایف و توازن

[1] Heterogeneous distributed computing systems
[2] Task scheduler
[3] Subtasks

بار در محاسبات ابری، بهره برده‌اند. اهداف اصلی الگوریتم پیشنهادی، به حداقل رساندن زمان اتمام کل کارها[1]، زمان انتظار و درجه عدم تعادل و در عین حال بهینه‌سازی استفاده از منابع و به حداقل رساندن زمان اجرا و هزینه اجرا می‌باشد. پیچیدگی زمانی با زمان زمان‌بندی (زمان اجرا) اندازه‌گیری می‌شود. با بهبود روش به‌روزرسانی موقعیت ذرات با توجه به استراتژی متعادل‌سازی بار و تابع هدف تعریف‌شده، هزینه موردنیاز برای استفاده از منابع ابری کاهش‌یافته است. در [۱۰]، یک زمان‌بند مبتنی بر الگوریتم ترکیبی، شامل الگوریتم بهینه‌سازی کلونی مورچگان و ژنتیک معرفی کرده‌اند که خروجی زمان‌بندی را به‌عنوان ورودی خود می‌پذیرد. در الگوریتم ژنتیک، بهترین راه‌حل با اعمال تابع برازندگی[2] بر روی کروموزوم‌ها ارزیابی می‌شود و n بهترین راه‌حل‌ها یافت می‌شوند. nth-1 راه‌حل برازش الگوریتم ژنتیک به فرومن اولیه الگوریتم مورچگان تبدیل می‌شود. الگوریتم مورچگان با استفاده از اطلاعات اکتشافی و روش‌های به‌روزرسانی فرومن راه‌حل بهینه را پیدا می‌کند. عملیات متقاطع[3]، برای ترکیب بهترین راه‌حل الگوریتم ژنتیک و راه‌حل نهایی الگوریتم مورگان به کار گرفته می‌شود. در مقاله [۱۱] به منظور حل مشکلات عدم توازن بار، سرعت همگرایی پایین و استفاده ناکارآمد از منابع ماشین مجازی موجود در استراتژی‌های بهینه‌سازی زمان‌بندی، یک رویکرد زمان‌بندی وظایف، با استفاده از الگوریتم کلونی مورچگان ارائه کرده‌اند. تابع هدف زمان‌بندی وظایف، با ترکیب سه هدف کوتاه‌ترین زمان انتظار، درجه تعادل بار منبع و هزینه تکمیل کار برای جست‌وجوی راه‌حل بهینه زمان‌بندی ایجاد می‌شود. نتایج حاصل از شبیه‌ساز کلود سیم نشان می‌دهد که استراتژی پیشنهادی، نسبت به سایر روش‌ها، از سرعت همگرایی، زمان تکمیل، توازن بار بهتری برخوردار است. در [۱۲]، یک نسخه‌ی پیشرفته از بهینه‌ساز چند نظمی[4]، برای زمان‌بندی وظایف در محیط محاسبات ابری پیشنهاد شده است. بهبود الگوریتم، شامل افزودن عملیات جدید می‌باشد که برخی از بهترین راه‌حل‌ها، در هر تکرار ذخیره می‌شوند و پس از تعداد معینی از تکرار، این راه‌حل‌ها به‌عنوان یک راه‌حل جدید به الگوریتم بازگردانده می‌شوند. فاکتورهای اصلی برای ارزیابی رویکرد پیشنهادی، طول کار و هزینه و توان موردنیاز آن است و هدف اصلی به حداقل رساندن زمان اجرا است. به منظور بهبود عملکرد زمان‌بندی وظایف و غلبه بر محدودیت‌های الگوریتم‌های زمان‌بندی سنتی، رویکرد پیشنهادی از قابلیت‌های جست‌وجوی محلی و سراسری، به همراه اپراتورهای اصلی انجام‌شده توسط بهینه‌ساز چند نظمی، بهره می‌برد. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم پیشنهادی، به طور قابل توجهی از هر دو الگوریتم بهینه‌ساز چند نظمی و ازدحام ذرات از نظر دستیابی به حداقل زمان اتمام کل کارها و بهبود استفاده بهینه از منابع بهتر عمل می‌کند.

[1] Makespan
[2] Fitness Function
[3] Crossover
[4] Multi-Verse Optimizer

نویسندگان در مقاله [۱۳] یک روش بهینه‌سازی چندهدفه جدید را با استفاده از الگوریتم بهینه‌سازی شیر مورچه[۱]، برای مسائل زمان‌بندی وظایف در محیط‌های رایانش ابری با پیکربندی/توزیع متعادل وظایف معرفی کرده‌اند. الگوریتم معرفی‌شده با یک تکنیک جست‌وجوی محلی یعنی استراتژی تکامل تفاضلی برای بهبود توانایی جست‌وجوی الگوریتم بهینه‌ساز شیر مورچه ترکیب شده است. در مقاله [۱۴]، به مسئله زمان‌بندی مجموعه‌ای از وظایف با مجموعه‌ای از گروه، به مجموعه‌ای از ابر با هدف به حداقل رساندن زمان کلی توقف[۲] پرداخته‌اند. در این مقاله، یک الگوریتم زمان‌بندی وظایف مبتنی بر جفت و بر اساس الگوریتم بهینه‌سازی مجارستانی را برای محیط محاسبات ابری ارائه کرده‌اند. الگوریتم پیشنهادی، تعداد نابرابر وظایف و ابرها را در نظر می‌گیرد و وظایف را برای تصمیم‌گیری زمان‌بندی جفت می‌کند. الگوریتم پیشنهادی هم‌زمان اجاره و هم‌زمان اجاره معکوس را برای تصمیم‌گیری زمان‌بندی در نظر می‌گیرد و از زمان انتقال، برای واقعی‌تر کردن الگوریتم پیشنهادی استفاده می‌کند. ارزیابی‌های عملکرد نشان می‌دهد که الگوریتم پیشنهادی می‌تواند به طور مؤثر وظایف را زمان‌بندی کند و با حداقل زمان توقف پیاده‌سازی شود. در [۱۵] بهبود الگوریتم الهام گرفته از طبیعت (بهینه‌ساز چند نظمی) با ویژگی‌های چندهدفه، برای بهبود اثربخشی زمان‌بندی انتقال وظایف ابری ناهمگن است. در روش پیشنهادی در این مقاله به‌منظور افزایش قابلیت جست‌وجوی بهینه‌ساز چند نظمی و جلوگیری از گیرکردن در بهینه محلی، این بهینه‌ساز، با الگوریتم ژنتیک ترکیب شده است. روش پیشنهادی می‌تواند وظایف انتقال را بر اساس حجم کاری منابع ابری موجود برنامه‌ریزی کند. از سوی دیگر، الگوریتم ژنتیک می‌تواند بهینه‌ساز چند نظمی معمولی را با استفاده از فرآیندهای متقاطع و جهش برای بهبود زمان‌بندی وظایف آغازشده توسط بهینه‌ساز چند نظمی بهبود بخشد. همان‌طور که در بالا نیز بیان شد، زمان‌بندی مؤثر و کارآمد یک برنامه، برای رسیدن به کار آیی بالا در محیط‌های رایانش ناهمگن یک امر مبرم است [۱۶]. زمان‌بندی یک برنامه، در حالت عادی یک مسئله چندجمله‌ای غیر قطعی-کامل است. اخیراً برای به‌دست‌آوردن راه‌حل‌های نزدیک به بهینه[۳]در مسائل زمان‌بندی، الگوریتم‌های فرااکتشافی[۴] بسیاری ارائه شده‌اند [۱۷]. به‌طور خلاصه می‌توان گفت که الگوریتم‌های فرااکتشافی، راهکارهای پیشرفته و کلی جست‌وجو می‌باشند و گام ها و معیارهایی را پیشنهاد می‌کنند که در فرار از دام بهینه‌های محلی، بسیار مؤثر هستند. عامل اصلی در این روش‌ها، تعادل پویا بین استراتژی‌های تنوع‌بخشی و پرقدرت‌سازی است [۱۸]. تنوع‌بخشی، به جست‌وجوی گسترده در فضای جواب اشاره دارد و پرقدرت‌سازی، به معنی بهره‌برداری از تجربیات به‌دست‌آمده در فرآیند جست‌وجو و تمرکز بر نواحی پرامیدتر فضای جواب هست. بنابراین با ایجاد تعادل پویا بین این دو استراتژی از یک طرف، جست‌وجو به سمت محدوده‌هایی از فضای

[۱] Antlion optimizer algorithm

[۲] Layover time

[3] Near optimal

[4] Metaheuristic

جواب سوق داده می‌شود که جواب‌های بهتری در آنها یافت شده است، از طرف دیگر موجب عدم اتلاف زمان بیشتر در بخشی از فضای جواب می‌شود که پیش از این بررسی شده و یا شامل جواب‌های نامرغوب‌تری است [۱۹]. علی‌رغم موفقیت و کارایی الگوریتم‌های فرااکتشافی متعدد در مسائل بهینه‌سازی، در زمان‌بندی چالش‌هایی مثل حالت غیرخطی پیچیده، حالت چندگانه، ترکیبی و چند مقیاس کلان در این الگوریتم‌های وجود دارد [۲۰]. حل کردن این چالش‌ها و غلبه کردن بر این چالش‌ها، سبب می‌شود که هزینه‌ی محاسباتی و پیچیدگی محاسباتی به طرز چشم‌گیری افزایش پیدا کند. برای غلبه بر این چالش‌ها و این پیچیدگی‌ها، الگوریتم‌های فرااکتشافی یا ارائه قالب‌های کاری جست‌وجوی تصادفی راه‌حل‌های شبه‌بهینه را ارائه می‌کنند [۲۱]. این راه‌حل‌ها علاوه بر رسیدن به پاسخی معقول، پیچیدگی محاسباتی قابل قبولی دارند. از این رو در این مقاله، یک روش زمان‌بندی پیشنهادی[۱] بر پایه الگوریتم کاپوچین چندهدفه مبتنی صف‌های اولویت چندگانه و رتبه‌بندی نامغلوب برای سیستم‌های ابری ناهمگن ارائه‌شده است. در این روش درخواست توسط کاربران به زیرمجموعه‌ای از وظایف شکسته شده و با یک گراف مدل خواهد شد. الگوریتم بهینه‌سازی پیشنهادی این تحقیق، مبتنی بر الگوریتم کاپوچین با در نظر گرفتن تابع هدف مناسب، متغیرهای زمان اجرا، زمان پردازش، زمان اتمام کل کارها را بهبود بخشیده است. بر این اساس می‌توان نوآوری‌هایی که در این مقاله ارائه شده است را به شرح زیر بیان کرد:

- ارائه یک روش جهت زمان‌بندی در محیط سیستم‌های ابری ناهمگن با استفاده از صف‌های اولویت چندگانه مبتنی بر رتبه‌بندی نامغلوب
- ایجاد تعادل در سرعت همگرایی و جست‌وجو در فضای مسئله
- بهبود زمان اجرای کل کارها
- افزایش نرخ بهره‌برداری از منابع

این مقاله متشکل از چهار بخش است، در بخش ۲ پیشینه تحقیق و ابزارهای تحقیق معرفی شده‌اند. در بخش ۳ روش پیشنهادی ارائه‌شده و درنهایت در بخش ۴ شبیه‌سازی و ارزیابی نتایج، آورده شده است. در آخرین بخش نیز نتیجه‌گیری مطرح شده است.

۲- پیشینه تحقیق

[۱] Multi Objective Capuchin Search Algorithm (MOCapSA) without priority queue

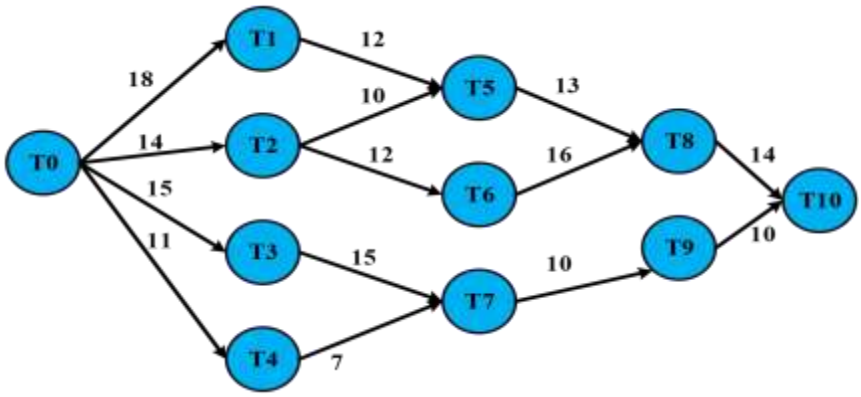
وقتی یک وظیفه‌ای از سمت کاربر به سیستم ارسال می‌شود، این وظایف بر اساس وضعیت منابع ابری، به هر کدام از این منابع تخصیص داده می‌شوند که این منابع در واقع همان ماشین‌های فیزیکی شما بوده، که هرکدام مجموعه‌ای از ماشین‌های مجازی را شامل می‌شود [۲۲]؛ پس وقتی وظایف از سمت کاربر رسید، این وظایف می‌تواند به چندین زیر وظیفه تقسیم شده و این زیر وظایف روی چندین ماشین فیزیکی توزیع می‌شوند. چگونگی تخصیص وظایف به منابع به صورت مناسب و مؤثر، زمان‌بندی کار نامیده می‌شود. هدف اصلی زمان‌بندی کار، کوچک کردن زمان اجرای کار و افزایش توان عملیاتی سیستم است [۲۳]. در مسئله‌ی زمان‌بندی در جهت توازن بار همواره باید زمان و ترتیب پردازش کارها، در نظر گرفته شود. هنگام درخواست کاربران مبنی بر پردازش کارها، سیستم باید تصمیم بگیرد که ترتیب پردازش کارها به چه صورت باشد تا زمان اتمام کل کارها کاهش یابد [۲۴]. در این بخش از مقاله، نحوه‌ی زمان‌بندی وظایف در ابر ناهمگن با استفاده از الگوریتم بهینه‌سازی کاپوچین چندهدفه[۱] مبتنی بر صف‌های اولویت و رتبه‌بندی نامغلوب، ارائه شده است؛ بتواند برای کاهش زمان اتمام، کاهش هزینه‌ی اجرای کارها و ایجاد توازن بار خوب روی منابع اعمال شود.

از آنجایی که در روش پیشنهادی منابع ناهمگن ابری در نظر گرفته شده است، بنابراین مدل سیستم مورد استفاده در روش پیشنهادی دارای یک مجموعه از پردازنده‌های ناهمگن P می‌باشد (منظور از پردازنده در واقع ماشین‌های فیزیکی است). سیستم‌های محاسباتی ناهمگن، مجموعه‌ای از ماشین‌های محاسباتی یا ماشین‌های فیزیکی است [۲۵]. این ماشین‌های فیزیکی از ماشین‌های مجازی موجود در خودشان، به عنوان پردازشگر درخواست‌ها یا همان وظایف کاربران استفاده می‌کنند. با توانمندی‌های متمایز بوده که به هم متصل شده‌اند تا برنامه‌های موازی یا وظایف درخواستی توسط کاربران را اجرا کنند. مسئله زمان‌بندی در این مقاله، به صورت مسئله زمان‌بندی گراف وظایف مدل شده است. مسئله زمان‌بندی در مدل گراف وظایف در سیستم‌های محاسباتی بدین‌صورت مطرح می‌گردد، برنامه‌های بزرگ از وظایف کاربران که به سیستم ارسال شده است به تعدادی از برنامه‌های کوچک‌تر تقسیم می‌شود. در این مدل‌ها در رایانش ابری وظایف بزرگ به تعدادی زیر وظیفه تقسیم می‌شوند و هرکدام از این زیر وظایف، به یک منبع محاسباتی (ماشین فیزیکی) تخصیص داده می‌شوند [۲۶]. وظایف یا زیر برنامه‌های ایجادشده معمولاً با همدیگر رابطه تقدم و تأخر دارند و هر وظیفه تا وظایف ماقبل آن اجرا نگردند، نمی‌تواند اجرا شود. زمانی که یک وظیفه به تعدادی زیر وظیفه تقسیم می‌شود، متناسب با اولویتی که دارند، باید به ترتیب اجرا شده و درنهایت نتیجه‌ی نهایی را تولید خواهند کرد [۲۷]. هدف اصلی از مسئله زمان‌بندی گراف وظایف این است که وظایف برنامه‌ها با توجه به ترتیب اجرای آن‌ها، به ماشین‌ها طوری تخصیص یابند که شرایط تقدم و تأخر مابین وظایف برآورده شود و کمترین زمان اتمام کلی برنامه‌ها به دست آید. در مدل پیشنهادی

[۱] Multi-objective capuchin search algorithm

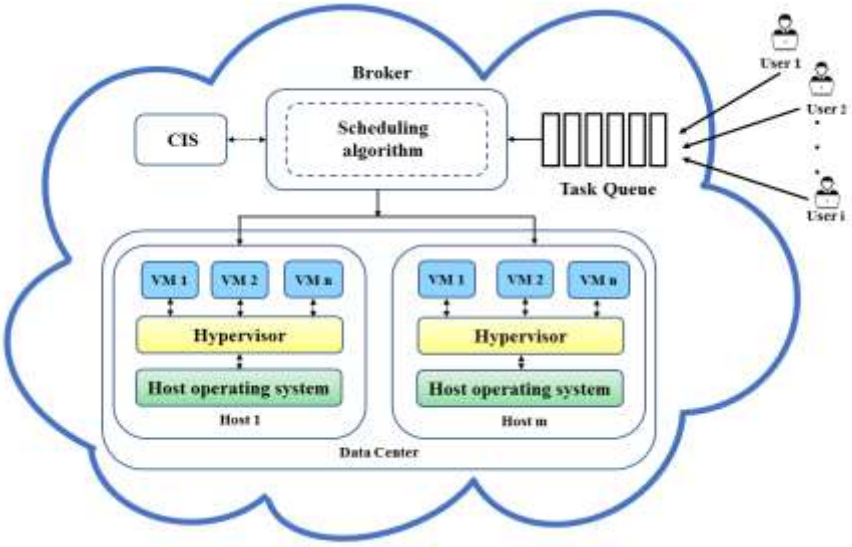
یک برنامه موازی می‌تواند با گراف جهت‌دار بدون دور $G(V,E)$ مدل شود که در آن V مجموعه‌ای از گره‌ها بوده و هر گره $n_i \in V$ یک وظیفه برنامه را نشان می‌دهد. این وظیفه دنباله‌ای از دستورات است که بایستی روی یک منبع به صورت ترتیبی اجرا شوند. E مجموعه‌ای از یال‌های ارتباطی و نشان‌دهنده لبه‌های گراف است که وابستگی‌های بین وظایف را مشخص می‌کنند [۲۸].

یال جهت‌دار $e_{i,j}$ گره‌های n_i و n_j را به هم متصل می‌کند؛ که گره n_i را گره والد و گره n_j را گره فرزند می‌گویند. n_j نمی‌تواند شروع شود تا زمانی که n_i تمام شود و داده‌هایش را به n_j بفرستد. مدل سیستم محاسباتی ناهمگن، یک مجموعه Q شامل q ماشین ناهمگن است که با استفاده از توپولوژی کاملاً متصل به هم وصل شده‌اند. در مدل سیستم پیشنهادی، هر وظیفه در گراف فقط بر روی یک منبع می‌تواند اجرا شود. در شکل ۱ شمایی از مدل توزیع وظایف به شکل گراف در سیستم پیشنهادی آورده شده است. در این مدل، مقدارهای لبه‌ها نشانگر مقدار هزینه ارتباطی بین دو وظیفه است و میانگین هزینه محاسباتی هر وظیفه در پردازنده‌ها (ماشین مجازی) بر روی هر گره مقدار گذاری شده است. طبق مدل سیستم پیشنهادی، درصورتی‌که دو وظیفه متصل به یکدیگر، بر روی ماشین فیزیکی یکسان زمان‌بندی شوند هزینه ارتباطی بین آن‌ها صفر در نظر گرفته می‌شود. گره‌های شروع و خروجی گراف به ترتیب نشان‌دهنده شروع و پایان برنامه هستند. مدل پیشنهادی را می‌توان یک مدل نامرتبط [1] نیز تعریف کرد، زیرا که یک منبع ممکن است بعضی از وظایف را در زمان کمتر و بعضی دیگر را در زمان بیشتر اجرا کند. همچنین برای شروع اجرای یک وظیفه می‌بایست تمام وظایف تقدم [2] آن زمان‌بندی اجرا شده باشند. شکل ۲ معماری زمان‌بندی وظایف در محیط محاسباتی ابر را با چندلایه (لایه وظایف، لایه ماشین‌های مجازی و لایه ماشین‌های فیزیکی) بیان می‌کند.



شکل ۱) نمودار گراف وظایف [3] با ۱۱ زیر وظیفه

[1] unrelated
[2] precedence
[3] Directed Acyclic Graph(DAG)



شکل ۲) معماری زمان‌بندی وظایف در محیط محاسباتی ابر

در ادامه، ساختمان داده راه‌حل‌ها ارائه شده است. در این مقاله هر راه‌حل به صورت یک بردار n بعدی در نظر گرفته شده که هر بعد نشان‌دهنده یک وظیفه می‌باشد. مقدار درج‌شده در هر بعد، نمایانگر شماره منبع داده شده به وظیفه می‌باشد. اگر هر راه‌حل از مسئله را به عنوان یک نقطه از فضای مسئله در نظر بگیریم، موقعیت این نقطه نشان‌دهنده زمان‌بندی بالقوه می‌باشد؛ که این موقعیت در طی فرآیند جست‌وجو تغییر می‌کند. برای تعیین یک راه‌حل می‌توان موقعیت این نقاط از فضای مسئله را کدگذاری کرد. نمونه‌ای از ساختمان داده راه‌حل‌ها در جدول ۱ نشان داده شده است؛ که در آن راه‌حل (موقعیت در فضای مسئله) دارای شش بعد بوده و هر بعد نشان‌دهنده یک وظیفه می‌باشد که در مجموع شش وظیفه در این راه‌حل وجود دارد. مقدار درج‌شده در هر بعد، نشان‌دهنده شماره منبعی است که به وظیفه مشخصی اختصاص داده شده است. به طور مثال وظیفه T2 به منبع ۱ و برای وظیفه T5 به منبع ۲ اختصاص داده شده است. در این نوع کدگذاری منابع تکراری می‌توان به وظیفه‌های مختلف اختصاص داد.

جدول ۱) تخصیص منابع به کارها

کارها	T1	T2	T3	T4	T5	T6
منابع	3	1	2	3	2	4

۱-۲- مدل ریاضی روش پیشنهادی

روشی که در این مقاله استفاده شده است یک روشی برای بهبود زمان‌بندی در رایانش ابری ناهمگن است. در این مقاله، بهینه کردن زمان اجرا، زمان اتمام کل کارها روی منابع برای زمان‌بندی کارها در رایانش ابری اهدافی هستند که به یک روش برای یافتن راه‌حل بهینه نیاز دارند. در ادامه مدل ریاضی این اهداف آورده شده است.

مشکل زمان‌بندی وظایف، در محیط‌های ابری شامل توزیع و تخصیص انواع وظایف در چندین ماشین مجازی و تکمیل تمام وظایف در یک‌زمان معقول است. ما یک سیستم ابری را در نظر می‌گیریم که از n تعداد ماشین مجازی[۱] (vm) و m تعداد ماشین فیزیکی[۲] (pm) تشکیل شده است. همان‌طور که در معادله ۱ نشان داده شده است. C ابر را بیان می‌کند، pm_1 مخفف اولین pm است و pm_m به mth ، pm اشاره دارد. هر vm را می‌توان با استفاده از معادله ۲ نشان داد؛ که در آن vm_1 و vm_n به ترتیب اولین و آخرین vm را نشان می‌دهند. ابر شامل k تعداد وظیفه ثبت‌شده توسط t تعداد کاربر است. (هر سیستم می‌تواند تعداد t تا کاربر داشته باشد که هرکدام از این کاربران می‌توانند، k وظیفه را به این سیستم ارسال کنند. پس یک ابر می‌تواند دارای k وظیفه باشد. اگر فرض کنیم هر وظیفه را به یک ماشین فیزیکی اختصاص دهیم، پس هر ابر می‌تواند مجموعه‌ای از ماشینی‌های فیزیکی داشته باشد.) هر کاربر را می‌توان با معادله ۳ نشان داد. هدف ما به حداقل رساندن زمان اجرا و زمان اتمام کل کارهاست، در حالی است که حجم کار در تمام ماشین‌های مجازی متعادل می‌شود.

$$C = \{pm_1, pm_2, \dots, pm_m\} \quad (۱)$$

$$VM_j = \{m_1, vm_2, \dots, vm_n\} \qquad 0 < j \leq n \quad (۲)$$

$$U_t = \{T_1, T_2, \dots, T_k\} \qquad 0 < t \leq k \quad (۳)$$

$$\mathrm{p\ m}_i = \sum_{j=1}^{n} \mathrm{v\ m}_j \qquad i = 1, 2, 3, \dots, m \qquad 0 < j \leq n \quad (۴)$$

پارامتر هدف اول: کاهش زمان اتمام کل کارها

این معیار، زمان اتمام پردازش کل کارها را مشخص می‌کند که هر چه مقدار آن کمتر باشد، نشان‌دهنده‌ی این است که الگوریتم توانسته زودتر کارها را پردازش کرده و در اختیار کاربران قرار دهد. زمان اتمام کل با استفاده از رابطه‌ی (۵) به دست می‌آید.

[1] virtual machine (VM)
[۲] physical machine (pm)

$$ET\left(T_i,R_j\right)=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{Lt_i}{Pc_j}\qquad 1\le i\le n,1\le j\le m \tag{۵}$$

که Lt_i، Pc_j و ET به ترتیب نشان‌دهنده‌ی طول هر کار i، ظرفیت پردازش هر منبع j و زمان پردازش هر کار i روی منبع j می‌باشند. اگر کار T_i اولین کاری باشد که روی منبع R_j اجرا می‌شود، زمان شروع آن برابر ۰ است، در غیر این صورت زمان شروع آن برابر مجموع زمان اتمام کار قبلی یعنی T_{i-1} روی منبع R_j و زمان راه‌اندازی کار T_i بعد از کار T_{i-1} روی منبع R_j است. رابطه‌ی زیر زمان شروع کار T_i را روی منبع R_j بیان می‌کند:

$$BT_{(T_i,R_j)}=FT_{(T_{i-1},R_j)}+ST_{(T_{i-1},T_i,R_j)} \tag{۶}$$

که FT، ST وBT به ترتیب بیانگر زمان اتمام، زمان بارگذاری و زمان شروع یک کار هستند. (زمان اتمام کار را که با رابطه ۷ برای هر وظیفه محاسبه کرده‌ایم؛ پس این را داریم. زمان شروع یک کار، بر اساس اولویت اجرای کارهای موجود، در صف وظایف معلوم است. مثلاً کار اول شما، وقتی سه ثانیه طول کشیده است، زمان شروع کار بعدی شما از زمان t=4 شروع خواهد شد. زمان بارگذاری، یک عدد بر اساس زمان در نظر گرفته شده است که فرض گرفته می‌شود که هر وظیفه نیاز دارد، بعد طی این زمان وارد فاز اجرایی شود. این عدد یک عدد ثابت است که بسته به دیتاست، انتخاب می‌شود.)

زمان اتمام هر کار T_i روی منبع R_j برابر است با مجموع زمان شروع و زمان اجرای هر کار روی آن منبع که به صورت رابطه‌ی زیر می‌باشد:

$$FT(T_i,R_j)=BT(T_i,R_j)+ET(T_i,R_j) \tag{۷}$$

زمان تکمیل تمام‌کارها روی هر منبع R_j برابر است با زمان اتمام آخرین کار $T_{i'}$ روی منبع R_j که با استفاده از رابطه‌ی زیر به دست می‌آید:

$$CT(R_j)=FT(T_{i'},R_j) \tag{۸}$$

که CT نشان‌دهنده‌ی زمان تکمیل تمام کارهای اجراشده روی یک منبع است.

زمان اتمام کل کارها برابر است با حداکثر زمان لازم برای تکمیل تمام کارها روی هر منبع R_j که به صورت زیر بیان می‌شود:

$$Makespan = Max\{CT(R_j)\} \qquad 1 \le j \le m \qquad (۹)$$

که زمان اتمام کل کارها بیانگر حداکثر زمان لازم برای تکمیل تمام‌کارها می‌باشد. در مسئله بهینه‌سازی چندهدفه پیشنهادی، در گام اول هدف، مینیمم سازی زمان اتمام کل کارها که توسط رابطه ۹ محاسبه می‌شود، می‌باشد.

۲-۲- مفهوم غلبه در الگوریتم‌های بهینه‌سازی

در این بخش از مقاله، قبل از پرداختن به مراحل روش پیشنهادی الزامات لازم در خصوص روش پیشنهادی تشریح شده است. در یک مسئله‌ی کمینه‌سازی با بیش از یک تابع هدف، گفته می‌شود نقطه‌ی $X = (x_1, \dots, x_{Nobj})$ بر نقطه‌ی $X^* = (x_1^*, \dots, x_{Nobj}^*)$ غلبه می‌کند، اگر و تنها اگر X^* از هیچ نظر بهتر از X نباشد و X حداقل از یک نظر اکیداً بهتر از X^* باشد. این مفهوم به صورت ریاضی به شکل زیر بیان می‌شود.

$$X < X^* \; if \; \forall_o \in 1,\dots,Nobj, \qquad x_o \le x_o^* \quad and \quad X \ne X^* \qquad (۱۰)$$

در این رابطه، X و X^* دو نمونه جواب برای حل مسئله، x_o و x_o^* مقدار هر درایه از جواب، o اندیس توابع هدف و $Nobj$ تعداد توابع هدف می‌باشند.

۲-۳- مرتب‌سازی نامغلوب در الگوریتم رتبه‌بندی‌شده

زمانی که مسئله در مورد بهینه‌سازی یک الگوریتم تک‌هدفه است، معیار برتری جواب‌ها نسبت به هم بسیار ساده و بدیهی است؛ زیرا تنها یک تابع هدف مدنظر می‌باشد؛ بنابراین، برای به‌دست‌آوردن بهترین جواب‌ها باید آن‌ها را بر اساس یک معیاری مرتب کرد. در این الگوریتم به هر جواب یک رتبه اختصاص داده می‌شود که این رتبه‌بندی بر اساس تعداد مغلوب شدن آن‌ها نسبت به سایر نقاط می‌باشد. در پایان الگوریتم، نقاطی که بهترین رتبه یعنی رتبه‌ی یک را دارا باشند، به عنوان مجموعه جواب یا نقاط سطح پارتو انتخاب می‌شوند.

۲-۴- حفظ تنوع پاسخ‌ها (فاصله ازدحامی) در الگوریتم‌های رتبه‌بندی‌شده

در الگوریتم‌های رتبه‌بندی‌شده، گاهی مجبور هستیم مقایسه‌ای در بین اعضای یک مجموعه که رتبه‌ی یکسانی دارند، انجام دهیم و برخی را حذف کنیم. این کار با استفاده از مفهوم حفظ تنوع پاسخ‌ها انجام می‌گیرد؛ به عبارت دیگر در حذف کردن چند عضو از یک مجموعه، سعی بر این است که در آن مجموعه، از هر بازه‌ای به طور منظم پاسخ وجود داشته باشد. مقدار فاصله‌ی ازدحامی برای هر جواب، تخمینی از چگالی جواب‌ها در اطراف آن است. فاصله‌ی ازدحامی[1] نسبت به همسایه‌ی قبل و همسایه‌ی بعد و اولین و آخرین عضو جمعیت به دست می‌آید. در مسئله‌ی مورد بررسی برای هرکدام، یک تابع هدف وجود دارد لذا، برای محاسبه‌ی فاصله ازدحامی منتسب به هر نقطه روی یک جبهه‌ی مشخص، ابتدا برای هر بعد، مقادیر تابع هدف آن ُعد به‌صورت نزولی، مرتب می‌شود و سپس نقاط قبل و بعد نسبت به توابع هدف مسئله انتخاب می‌شوند. کسری از آن بعد که عضو i اُم جمعیت آن را پوشش می‌دهد، توسط روابط ۱۱، ۱۲ و ۱۳ به دست می‌آید. فاصله‌ی ازدحامی نیز از طریق رابطه‌ی ۱۴ محاسبه می‌شود.

$$d_i^1 = \frac{\left|f_1^{i+1} - f_1^{i-1}\right|}{f_1^{max} - f_1^{min}} \quad (۱۱)$$

$$d_i^2 = \frac{\left|f_2^{i+1} - f_2^{i-1}\right|}{f_2^{max} - f_2^{min}} \quad (۱۲)$$

$$d_i^3 = \frac{\left|f_3^{i+1} - f_3^{i-1}\right|}{f_3^{max} - f_3^{min}} \quad (۱۳)$$

$$d_i = d_i^1 + d_i^2 + d_i^3 \quad (۱۴)$$

در روابط فوق، d_i^1، d_i^2 و d_i^3 به ترتیب کسری از توابع هدف اول، دوم و سوم هستند که عضو i اُم جمعیت، آن‌ها را تحت پوشش قرار داده است. d_i نشان‌دهنده‌ی فاصله‌ی ازدحامی عضو i اُم جمعیت هست. f_1^{min} ، f_2^{min} و f_3^{min} به ترتیب کمترین مقدار توابع هدف اول، دوم و سوم هستند و همچنین f_1^{max} ، f_2^{max} و f_3^{max} به ترتیب بیشترین مقدار توابع هدف اول، دوم و سوم هستند. بدیهی است هر چه عضو i اُم جمعیت، ناحیه بزرگ‌تری را پوشش دهد، بهتر است.

۵-۲- الگوریتم جست‌وجوی کاپوچین

[1] Crowding distance

این الگوریتم، برای حل مشکلات بهینه‌سازی سراسری و محلی ارائه شده است. الهام اصلی الگوریتم CapSA، رفتار پویای میمون‌های کاپوچین است که از رفتار اجتماعی کاپوچین‌ها در هنگام جست‌وجو بین درختان و حاشیه رودخانه‌ها در جنگل‌ها، هنگام جست‌وجوی منابع غذایی الهام گرفته شده است. در زیر، مراحل الگوریتم جست‌وجوی کاپوچین برای انتخاب بهترین ویژگی‌ها آورده شده است.

مرحله اول: مقداردهی اولیه در الگوریتم جست‌وجوی کاپوچین

مانند سایر الگوریتم‌های مبتنی بر هوش جمعی، الگوریتم جست‌وجوی کاپوچین را می‌توان به عنوان یک الگوریتم مبتنی بر جمعیت توصیف کرد که با مقداردهی اولیه تعداد از پیش تعیین‌شده افراد (کاپوچین‌ها) به طور تصادفی آغاز می‌شود. هر یک از این افراد یک راه‌حل کاندید برای مسئله بهینه‌سازی که در این مقاله مسئله زمان‌بندی می‌باشد، است. مجموعه‌ای از n کاپوچین در یک دامنه جست‌وجوی d بعدی را می‌توان با یک ماتریس، x در d نشان داد. ماتریس جمعیت اولیه در رابطه ۱۵ نشان داده شده است.

تابع برازندگی

عملکرد برازندگی هر کاپوچین، با تنظیم مقادیر متغیرهای تصمیم‌گیری در یک تابع برازندگی تعریف‌شده توسط کاربر ارزیابی می‌شود. این تابع برازندگی یک تابع با ضرایب وزنی بوده که بر اساس پارامترهای موردنظر تدوین می‌شود. سپس مقادیر مربوطه در ماتریس ذخیره می‌شوند، همان‌طور که در رابطه ۱۵ آورده شده است.

$$x = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & \cdots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^n & x_2^n & \cdots & \cdots & x_d^n \end{bmatrix} \quad (۱۵)$$

که در آن x موقعیت کاپوچین‌ها را نشان می‌دهد، d تعداد متغیرهای مسئله است، n تعداد کاپوچین‌ها است و x_d^i نشان‌دهنده ابعاد d اُمین بعد از موقعیت i اُمین کاپوچین است. از رابطه ۱۶، برای اختصاص موقعیت اولیه هر کاپوچین در جمعیت استفاده می‌شود.

$$x^{ij} = ub_{ij} + r \times (ub_{ij} - lb_{ij}) \quad (۱۶)$$

در رابطه بالا که lb_{ij}و ub_{ij} به ترتیب پایین‌ترین و بالاترین کران برای i اُمین کاپوچین در j اُمین بعد هستند، و r یک عدد تصادفی است که به طور یکنواخت در محدوده ۰ تا ۱ تولید می‌شود. X^i درواقع موقعیت جدید کاپوچین است. اندیس i معین کننده موقعیت جدید است و اندیس j معین کننده موقعیت فعلی است.

مرحله دوم: تولید راه‌حل توسط کاپوچین

تولید جمعیت جدید (راه‌حل‌های جدید) در CapSA به موقعیت کنونی کاپوچین و موقعیت بهترین کاپوچین‌ها و همچنین منبع غذایی موسوم به F، متکی است. منبع غذایی، F، هدف کاپوچین‌ها در دامنه جست‌وجوی d بعدی است که باید به صورت تکراری به روز شود. برای تولید راه‌حل جدید در الگوریتم پیشنهادی از رابطه زیر بهره برده است:

$$x_j^i = F_j + \frac{P_{bf}(v_j^i)^2 \sin(2\theta)}{g} \qquad (۱۷)$$

پارامترهای استفاده‌شده در رابطه ۱۷ به شرح زیر می‌باشد:

x_j^i موقعیت کاپوچین‌های آلفا و سایر کاپوچین‌های دنبال کننده در بعد j را مشخص می‌کند.

F_j موقعیت غذا در بعد j است.

P_{bf} احتمال تعادلی است که توسط دم کاپوچین‌ها، هنگام حرکت جهشی ایجاد می‌شود.

g نیروی گرانشی است که برابر با ۹/۸۱ است.

θزاویه پرش کاپوچین‌ها است.

τ یک پارامتر طول عمر است که به طور سیستماتیک در تمام تکرارها کاهش می‌یابد.

v_j^i سرعت کاپوچین i اُم در بعد j است.

زاویه پرش کاپوچین‌ها را می‌توان توسط رابطه ۱۸ محاسبه کرد:

$$\theta = \frac{3}{2} r \qquad (۱۸)$$

که در آن r یک عدد تصادفی است که در فاصله [۰، ۱]، به طور یکنواخت تولید می‌شود. طول عمرCapSA در این محدوده، یک تابع‌نمایی برای ایجاد تعادل بین جست‌وجو و انتخاب ویژگی‌ها در طی فرآیندهای جست‌وجوی سراسری و محلی است که توسط رابطه ۱۹ محاسبه می‌شود.

$$\tau = \beta_0 e^{-\beta_1 (\frac{k}{K})^{\beta_2}} \qquad (۱۹)$$

که در آن k و K به ترتیب، ماکزیمم مقدار تکرار و میزان تکرار فعلی را نشان می‌دهند. پارامترهای β_0، β_1 و β_2 به ترتیب، با مقادیر ۲ ، ۲۱ و ۲ مقداردهی می‌شوند.

الگوریتم پیشنهادی از تابع‌نمایی τ، برای کاوش در فضای جست‌وجو با تعداد معقولی از ویژگی (راه‌حل‌ها) استفاده می‌کند، جایی که کاپوچین‌ها، از هر منطقه جست‌وجو برای یافتن بهترین راه‌حل‌ها استفاده می‌کنند. مقادیر تولیدشده توسط این تابع تأثیر زیادی در جست‌وجو و انتخاب ویژگی دارد، زیرا مقادیر مناسب این فرمول قابلیت‌های جست‌وجو و انتخاب ویژگی در الگوریتم پیشنهادی را تقویت می‌کند. این تعادل با تعیین موقعیت سازگار پارامترهای β_0 ، β_1 و β_2در حین تکرار حفظ می‌شود. به عبارت خاص‌تر، این تابع (τ) به CapSA کمک می‌کند، تا موقعیت‌های کاپوچین‌ها را به طور مؤثر به روز کند تا با پیمایش دقیق فضای مسئله، منبع غذایی بهینه (مناسب‌ترین ویژگی) را به سرعت پیدا کند. سرعت کاپوچین i اُم در بعد j را می‌توان به‌صورت زیر محاسبه کرد.

$$v_j^i = \rho v_j^i + \tau a_1 \left(x_{best_j}^i - x_j^i\right) r_1 + \tau a_2 (F_j - x_j^i) r_2 \qquad (۲۰)$$

پارامترهای رابطه بالا به شرح زیر می‌باشد:

$i=0,1,2 ..,N$ اندیس‌های کاپوچین‌ها را در جمعیتی به اندازه n نشان می‌دهد.

$j=1,2,3,..,d$ نشان‌دهنده بعد مسئله است.

v_j^i سرعت فعلی کاپوچین i اُم در بعد j است.

x_j^i موقعیت فعلی عنصر j اُمین کاپوچین است.

$x_{best_j}^i$ نشان‌دهنده بهترین موقعیت کاپوچین i اُم در بعد j است.

F_j موقعیت غذا در بعد j است.

a_1و a_2 دو ثابت مثبت هستند که تأثیر $x_{best_j}^i$ و F_j را روی سرعت کاپوچین کنترل می‌کنند که به دلخواه، ۱.۰ انتخاب شده است.

r_1 و r_2 دو عدد تصادفی هستند که به طور یکنواخت در بازه [۰ ، ۱] تولید می‌شوند.

ضریب اینرسی ρ تأثیر سرعت قبلی را بر حرکت کاپوچین کنترل می‌کند که برابر ۰.۷ می‌باشد.

۳- فازهای روش پیشنهادی

روش پیشنهادی در این مقاله از مراحل زیر تشکیل شده است که در ادامه آورده شده است:

شکل ۳ شبه کد روش پیشنهادی را نشان می‌دهد.

Algorithm 1. Proposed method

1: Initial proposed method parameters:
- initialize resourcelist[Number of Resources]
- initialize joblist[Number of Jobs]
- Input r = number of iterations
- Input n= number of heuristics
- Initialize a random feasible solution

2: Create priority queue based on three categories
- New task queues
- The least work queue left
- Special task queues

3: Create new solution based on capuchin algorithm steps

Initial capuchins;
for i=3 to PopSize-1 do
for j=0 to ChSize-1 do
generate a solution j ∈ $(0, ChSize - 1)$randomly that not generated in pervious capuchin;
end for
move capuchin i from left to right in first valid place in the queue in order to has a valid topological order;
end for

4. Assigning subtasks to processors function

Fill the priority queue with subtasks;
while the priority queue is not empty **do**
Select the first subtask t_i from the priority queue;
for each processor p_k in the processor set **do**
Compute $EFT(t_i, p_k)$ value using the insertion-based HEFT scheduling policy;
Assign subtask t_i to the processor p_k that minimizes $EFT(t_i, p_k)$;
end for;
Remove t_i from the priority queue;
end while;
return makespan= $AFT(t_{exit})$.

شکل ۳) شبه کد روش پیشنهادی

۳-۱- اولویت‌بندی (رتبه‌بندی) وظایف کاربر

با وارد شدن هر کار به سیستم رایانش ابر، به صورت معمول، کار یا تقاضای واردشده به یک یا چند ماشین مجازی ارجاع داده می‌شود تا روند اجرای آن آغاز شود. این توافقات، اصطلاحاً توافقات سطح سرویس نامیده

می‌شوند. این امکان وجود دارد که کارهای زیادی در بازه زمانی یکسانی وارد سیستم شده باشند تا خدمات موردنظر خود را دریافت نمایند. این ورود به سیستم می‌تواند در همان لحظه باشد و یا پیش از این وارد سیستم شده باشند و طبق زمان‌بندی انجام‌شده توسط توزیع‌کننده تقاضا در سیستم ابر، اجرای آن به زمان فعلی موکول شده باشد. در این صورت کاربر می‌تواند با توجه به اولویتی، یک صف، برای کارکرد صحیح سیستم، در نظر بگیرد. در روش پیشنهادی، منظور از کیفیتی که کاربر می‌تواند انتظار داشته باشد، مدت زمان پاسخگویی سیستم به وی می‌باشد.

در روش پیشنهادی مقاله، هر کاربر با توجه به موارد مختلفی، از قبیل قابلیت پیش‌بینی در مورد وجود تقاضا در آینده‌ای معین، اهمیت زمانی کار به طور کلی، برآورد کاربر از منفعت ناشی از دریافت خدمات، قابلیت پرداختی کاربر و بسیاری موارد دیگر، از سطوح اولویتی مختلف، در کنار سایر ویژگی‌های موردنظر خود که از سیستم رایانش ابری انتظار دارد، استفاده می‌نماید. با توجه به آنچه گفته شد، طبق مدل پیشنهادی، هنگامی‌که یک کار به سیستم وارد می‌شود، علاوه بر توافقات رایج در سطح سرویس، مانند پهنای باند، فضای ذخیره‌سازی، قدرت پردازشی و مواردی از این دست، بر سر اولویت موردنظر برای کار نیز به توافق می‌رسد و با توجه به این توافق به یکی از صف‌های اولویتی وارد می‌شود. درروش پیشنهادی چند صف به شرح زیر وجود دارد:

- **صف کارهای جدید:** در این صف کارهای که تازه وارد سیستم شده‌اند، قرار می‌گیرند.
- **صف کمترین کار باقی‌مانده:** در این صف کارهای که مقدار خیلی کمی از آن‌ها باقی‌مانده است، قرار می‌گیرند.
- **صف کارهای ویژه:** در این صف کارهای بسیار حیاتی که در کوتاه‌ترین زمان باید اجرا شوند، قرار می‌گیرند.

۳-۲- انتخاب مناسب‌ترین ماشین (فیزیکی) برای تخصیص وظایف هر کاربر

برای انتخاب مناسب‌ترین ماشین برای تخصیص کارها از الگوریتم کاپوچین چندهدفه استفاده شده است. همان‌طور که در بخش قبلی نیز بیان شد، الگوریتم کاپوچین از دو مرحله اصلی، تولید جمعیت اولیه و تولید راه‌حل‌های جدید با استفاده از جست‌وجوی سراسری و محلی تشکیل شده است. در ادامه این زیر مراحل آورده شده است:

۳-۲-۱- تولید جمعیت اولیه

در این مرحله برای اجرای کارها، به صورت سراسر از الگوریتم کاپوچین چندهدفه استفاده شده است، در این روش، هر کاپوچین یک جواب برای مسئله در نظر گرفته می‌شود و این کاپوچین‌ها به صورت تصادفی توزیع شده‌اند. هدف در این مرحله، انتخاب مناسب‌ترین ماشین برای اختصاص وظایف به ماشین‌های مجازی می‌باشد، که برای این اختصاص از تابع هدف پیشنهادی که یک تابع چندهدفه است، استفاده شده است. در روش پیشنهادی، هر ماشین مجازی با موجودیت کاپوچین در نظر گرفته می‌شود، تا کارها را اجرا کنند.

۳-۲-۲- انتخاب مناسب‌ترین ماشین با استفاده از الگوریتم کاپوچین

در این مرحله با استفاده از الگوی حرکتی کاپوچین، ماشین‌ها پیمایش شده، تا مناسب‌ترین ماشین برای اجرای وظایف کاربر انتخاب شود. برای انتخاب مناسب‌ترین ماشین (مناسب‌ترین راه‌حل) از تابع برازندگی پیشنهادی که در ادامه آورده شده است، استفاده می‌شود. در الگوریتم پیشنهادی، انتخاب راه‌حل بهتر بر اساس عملگر مسابقه‌ای ازدحامی انجام می‌گیرد، به این صورت که:

راه‌حل S_1 ، راه‌حل S_2 را مغلوب می‌کند، اگر:

- رتبه‌ی S_1 کمتر از رتبه‌ی S_2 باشد یا
- رتبه‌ی S_1 و رتبه‌ی S_2 یکسان باشند و فاصله‌ی ازدحامی S_1 بیشتر از فاصله‌ی ازدحامی S_2 باشد.

همان‌طور که مطلع هستید، در الگوریتم‌های بهینه‌سازی، درنهایت یک جواب به دست نمی‌آید، بلکه مجموعه‌ای از جواب‌های نامغلوب که تقریبی از جبهه‌ی اول هستند، به دست می‌آیند، درصورتی‌که یک جواب نهایی نیاز است. لذا، باید از بین آن‌ها یک جواب به عنوان هدف زمان‌بندی کارها انتخاب شود. تاکنون روش‌های زیادی برای این کار پیشنهاد و مورداستفاده قرارگرفته‌اند که منطق فازی[۱] یکی از آن‌ها است. این روش برای اولین بار در [۲۸] ارائه شد و جزء پرکاربردترین روش‌ها برای انتخاب راه‌حل توافقی می‌باشد. در [۲۹] برای هر یک از توابع هدف، یک تابع عضویت به صورت رابطه‌ی ۲۱ در نظر گرفته شده است.

$$\mu_o^s = \begin{cases} 1 & f_o^s \le F_o^{min} \\ \dfrac{F_o^{max} - f_o^s}{F_o^{max} - F_o^{min}} & F_o^{min} < f_o^s < F_o^{max} \\ 0 & f_o^s \ge F_o^{max} \end{cases} \qquad (۲۱)$$

[۱] Fuzzy Set theory

که در آن F_o^{min}، F_o^{max}، f_o^s و μ_o^s به ترتیب نشان‌دهنده‌ی کمترین مقدار تابع هدف o اُم، بین تمام جواب‌های نامغلوب، بیشترین مقدار تابع هدف o اُم، بین تمام جواب‌های نامغلوب، مقدار تابع هدف o اُم برای جواب نامغلوب s اُم و تابع عضویت تابع هدف o اُم برای جواب نامغلوب s اُم می‌باشند.

برای هر راه‌حل نامغلوب s، تابع عضویت هنجارشده[1] با استفاده از رابطه‌ی (۲۲) به دست می‌آید.

$$\mu^s = \frac{\sum_{o=1}^{Nobj} \mu_o^s}{\sum_{s=1}^{S} \sum_{o=1}^{Nobj} \mu_o^s} \qquad (۲۲)$$

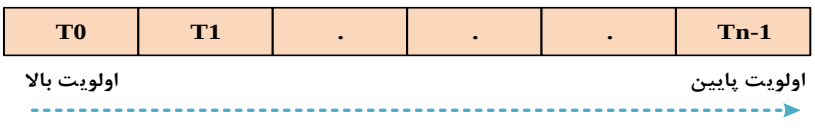
که μ^s تابع عضویت هنجارشده را نشان می‌دهد؛ بنابراین بهترین راه‌حل توافقی، جوابی است که دارای بیشترین مقدار برای μ^s می‌باشد.

۳-۳- تخصیص زیر وظایف به پردازنده‌ها (ماشین‌های مجازی) با استفاده از سیاست تخصیص پردازنده زودترین زمان پایان سیستم‌های ناهمگن

بعد از اینکه در مرحله قبل مناسب‌ترین ماشین فیزیکی، برای هرکدام از وظایف کاربر انتخاب شد، در این مرحله مناسب‌ترین پردازنده ماشین مجازی، برای اجرای زیر وظایف هر وظیفه انتخاب می‌شود. تخصیص زیر وظایف هر کدام از وظایف کاربر به پردازنده از چندین زیر مرحله تشکیل شده است؛ که در ادامه این زیر مراحل تشریح شده‌اند.

۳-۳-۱- مقداردهی اولیه

قبل از پرداختن به این مرحله باید جمعیت اولیه از راه‌حل‌ها تولید شود. همان‌طور که در بخش‌های قبلی نیز بیان شد، ساختمان داده راه‌حل‌ها یک آرایه با m بعد می‌باشد؛ که هرکدام از این راه‌حل‌ها، نشان‌دهنده یک راه‌حل از مسئله زمان‌بندی وظایف می‌باشد. ساختمان یک راه‌حل، شامل یک جایگشت از اعداد طبیعی ۰ تا n-1 می‌باشد که نشانگر ترتیب اولویت-های وظایف در گراف جهت‌دار بدون دور با تعداد n می‌باشد. شمایی از این کدبندی و همچنین ساختمان داده راه‌حل در شکل ۴ آورده شده است.



[1] Normalize

شکل ۴) رمزنگاری آرایه راه‌حل در الگوریتم پیشنهادی

در الگوریتم پیشنهادی، در مرحله مقداردهی جمعیت اولیه از سه سیاست رتبه‌بندی اکتشافی بنام‌های رتبه رو به بالا (رابطه ۲۳)، رتبه رو به پایین (رابطه ۲۴) و ترکیبی از این دو روش با رتبه سطح (رابطه ۲۵) برای مقداردهی سه راه‌حل اول جمعیت (سه کاپوچین اولیه جمعیت)، استفاده شده است. از این روش‌های رتبه‌بندی به‌کرات در مطالعات پیشین استفاده شده است [۳۰]. اولویت‌های راه‌حل‌های باقیمانده با جایگشت‌های تصادفی تولید می‌شوند. راه‌حل‌های تولیدشده تصادفی به دلیل اینکه اولویت‌هایشان معتبر نیست از سمت چپ به راست به طوری که زیر وظایف محدودیت‌های اولویت را نقض نکنند، مرتب‌سازی می‌شوند.

$$rank_b(t_i) = \overline{W(t_i)} + \max_{t_j \in succ(t_i)} \left(C(t_i . t_j) + rank_b(t_j) \right) \quad (۲۳)$$

در این رابطه $\overline{W(t_i)}$، میانگین هزینه محاسباتی وظیفه t_i ، $(t_i . t_j)$ مقدار هزینه ارتباطی بین وظایف t_i ,t_j و $rank_b(t_j)$ رتبه رو به بالای تأخر زیر وظیفه t_i است.

$$rank_t(t_i) = \max_{t_j \in pred(t_i)} \left(rank_t(t_j) + \overline{W(t_j)} + C(t_i, t_j) \right) \quad (۲۴)$$

در این رابطه $rank_t(t_j)$ رتبه رو به پایین زیر وظیفه تقدم t_i است.

$$Level(t_i) = \begin{cases} 0 \quad \text{if } t_i = t_{entry} \\ max_{t_j \in pred(t_i)} \left(Level(t_j) \right) + 1 \text{ otherwise} \end{cases} \quad (۲۵)$$

در این رابطه $Level(t_j)$ رتبه سطح تقدم زیر وظیفه t_i است.

۳-۳-۲- تخصیص زیر وظایف به پردازنده‌ها (ماشین‌های مجازی)

در روش پیشنهادی به منظور تخصیص وظایف به پردازنده‌ها، از سیاست تخصیص پردازنده زودترین زمان پایان سیستم‌های ناهمگن استفاده شده است [۳۱]. در این روش زیر وظیفه با بالاترین اولویت از بین زیر وظایف هر راه‌حل (وظیفه) انتخاب و به پردازنده‌ای تخصیص داده می‌شود که زمان اجرای آن از سایر پردازنده‌ها کمینه باشد. در ادامه این

بخش، به تشریح روش زودترین زمان پایان سیستم‌های ناهمگن پرداخته شده است. زودترین زمان شروع زیر وظیفه t_i بر روی پردازنده p_k به صورت $EST(t_i.p_k)$ نمادگذاری می‌شود و از رابطه ۲۶ به دست می‌آید.

$$EST(t_i.p_k) = max\left\{\max_{t_j \in pred(t_i)}\left(AFT(t_j) + C(t_j.t_i)\right)\right\} \quad (۲۶)$$

زمان شروع واقعی زیر وظیفه t_i بر روی پردازنده p_k با $AST(t_i.p_k)$ نمادگذاری می‌شود و از رابطه ۲۷ به دست می‌آید.

$$AST(t_i.p_k) = max(EST(t_i.p_k).Avail(p_k)) \quad (۲۷)$$

که $Avail(p_k)$ زمانی است که پردازنده p_k بیکار و آماده اجرای وظایف باشد.

زودترین زمان پایان زیر وظیفه t_i بر روی پردازنده p_k با $EFT(t_i.p_k)$ نمادگذاری شده است و از رابطه ۲۸ محاسبه می‌شود.

$$EFT(t_i.p_k) = W(t_i.p_k) + EST(t_i.p_k) \quad (۲۸)$$

که $W(t_i.p_k)$ هزینه محاسباتی زیر وظیفه t_i بر روی پردازنده p_k است. زمان واقعی پایان زیر وظیفه t_i بر روی پردازنده p_k با $AFT(t_i.p_l)$ نشان داده می‌شود و از رابطه (۲۹) به دست می آید.

$$AFT(t_i.p_k) = \min_{1 \le l \le P} EFT(t_i.p_l) \quad (۲۹)$$

که pk در رابطه بالا مناسب‌ترین[1] پردازنده برای زیر وظیفه t_i می‌باشد.

در این مرحله نیز، از یک تابع برازندگی برای انتخاب مناسب‌ترین پردازنده استفاده شده است. مقدار برازندگی در این مرحله بر اساس زمان اجرای برنامه در نظر گرفته شده است. زمان اجرای کل برنامه از رابطه (۳۰) به دست می‌آید.

$$makespan = max\{AFT(t_{exit})\} \quad (۳۰)$$

[1] fittest

همچنین رابطه (۳۱) برازندگی الگوریتم پیشنهادی را نشان می‌دهد.

$$fitness_i = makspan_i \quad (۳۱)$$

مقدار برازندگی کوچک، بیانگر کوتاه بودن زمان اجرای کل است؛ بنابراین بیشترین و کمترین مقدار برازندگی در میان جمعیت به ترتیب کاپوچین‌های بدتر و بهتر هستند. متناسب با مراحل بالا، تمامی زیر وظایف هر درخواست کاربر بین ماشین‌های فیزیکی و مجازی زمان‌بندی شده تا علاوه بر فراهم آوردن نیازمندی‌های منابع، نیازمندی‌های کاربر نیز فراهم شود.

۴- تحلیل و ارزیابی نتایج به‌دست‌آمده از روش پیشنهادی

در این بخش، نتایج به‌دست‌آمده برای روش پیشنهادی تحلیل شده است. الگوریتم کاپوچین با صف اولویت (روش پیشنهادی)، در دو سناریو مورد ارزیابی قرار می‌گیرد. روش پیشنهادی با الگوریتم کاپوچین چندهدفه بدون صف اولویت مورد ارزیابی قرار می‌گیرد. روش پیشنهادی بر اساس پارامترهای زمان اجرا و زمان اتمام کل کارها مورد مقایسه قرار گرفته شده است. این بخش از مقاله در دو سناریو مختلف تدوین شده است. در سناریو اول، هدف بررسی کردن عملکرد روش ترکیبی در مقایسه با تک‌تک روش‌های استفاده‌شده است.

۴-۱- تنظیمات مربوط به شبیه‌سازی مدل پیشنهادی

الگوریتم زمان‌بندی کار پیشنهادی، با استفاده از شبیه‌ساز متلب پیاده‌سازی شده است. جزئیات سخت‌افزار موردنیاز در جدول ۲ نشان داده شده است.

جدول۲) سخت‌افزار مورداستفاده

Component	Specification
Operating System	Windows ten **64** bit
CPU	Intel Core i**7**
RAM	**8.00** GB
VGA	**1.00 GB**

به منظور ساده‌سازی و پیاده‌سازی روش پیشنهادی، فرضیات زیر در نظر گرفته است:

- مراکز ابری یا مراکز داده از لحاظ جغرافیایی توزیع‌شده‌اند.
- از هزینه خواندن و نوشتن منابع، در زمان اجرای هر وظیفه صرفه نظر شده است.
- ماشین‌های مجازی، دارای توان‌های پردازشی متفاوت هستند.
- زمان‌بندی موردنظر ناهمگن است. به بیان دیگر روش پیشنهادی، وظایف متعددی را اولویت‌بندی زمان‌بندی خواهد کرد.
- زمان‌بندی انحصاری است و تا زمانی که یک سرویس رسانی به یک وظیفه اتمام پیدا نکرده، نمی‌توان منبع را از آن گرفت.
- زمان اجرای وظایف، بر اساس طول وظایف و سرعت ماشین‌های مجازی، بر اساس رابطه زیر محاسبه می‌شود:

$$ET = Mb/\left(Number\ of\ CPU * Mbps\right) \quad (۳۲)$$

در این رابطه (سرعت هر cpu در ماشین مجازی × تعداد cpu های هر ماشینی که این کار را پذیرفته) / حجم هر کار = زمان اجرای هر کار [۳۳].

۴-۲- داده‌های مربوط به شبیه‌سازی

به منظور ارزیابی روش پیشنهادی، از داده‌های موجود در پروژه CoMon که زیرساختی برای نظارت بر روی PlanetLab است، استفاده گردید. داده‌های موردنیاز به صورت فایل اکسل در متلب قابل بارگذاری و استفاده است. اطلاعات این داده‌ها مربوط به محیط ابر، در جدول 3 و اطلاعات مربوط به ماشین‌های مجازی، در جدول ۴ نشان داده شده است.

جدول ۳) اطلاعات محیط ابر موردنظر

VM_MI	1000 ~ 1000000
CLOUDLET_TYPES	**10**
CLOUDLET_FILESIZE	**300**
CLOUDLET_OUTPUTSIZE	**300**

جدول ۴) اطلاعات مربوط به ماشین مجازی

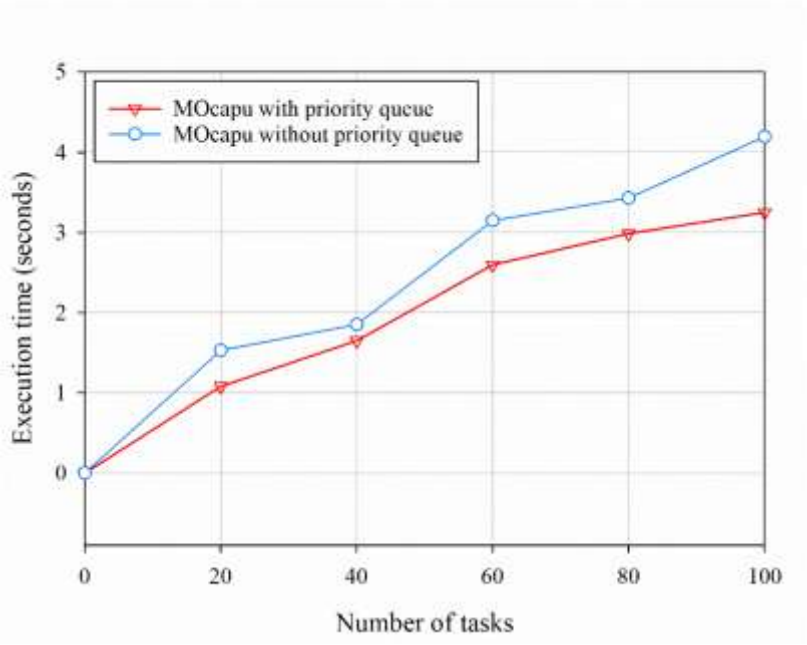
VM_MIPS	**1000, 1200, 1500, 2000**
maxhostVM	**2**
NumberOfHost	**2**

۴-۳- نتایج حاصل از شبیه‌سازی

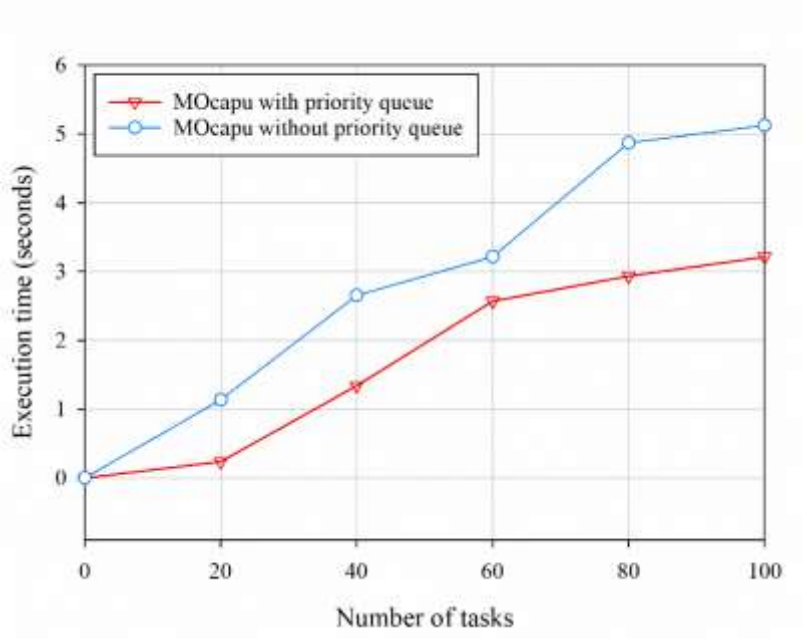
این بخش عملکرد روش پیشنهادی را مورد بحث قرار می‌دهد و کارایی آن را با کارهای قبلی با در نظر گرفتن معیارهای مختلف عملکرد از جمله زمان پردازش و زمان اجرا پاسخ، مقایسه می‌کند.

۴-۳-۱- زمان اجرا

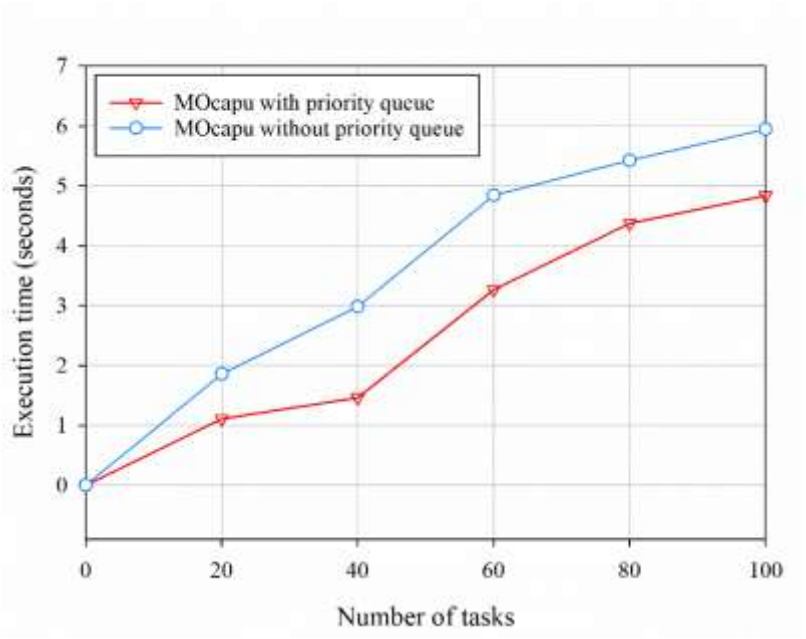
علاوه بر روش زمان‌بندی، تعداد وظایف ورودی نیز بر این پارامتر تأثیر می‌گذارد. به عنوان بخشی از کار، ما وظایف را در مهلت مقرر برنامه‌ریزی و پردازش می‌کنیم. همان‌طور که در شکل‌های ۵-۷ نشان داده شده است، ما الگوریتم خود را با الگوریتم جست‌وجوی کاپوچین چندهدفه بدون صف‌های اولویت، با تعداد مختلف ماشین‌های مجازی (VM) و فیزکی (PM) مقایسه می‌کنیم. همان‌طور که مشاهده می‌شود، الگوریتم ما زمان اجرا را کاهش می‌دهد. با تقسیم کل زمان اجرا بر تعداد کارها می‌توان میانگین زمان اجرا را محاسبه کرد. زمان اجرای روش پیشنهادی ما با افزایش تعداد کارها زیاد افزایش نمی‌یابد. با این حال، در کارهای پیشرفته، زمان اجرا به صورت خطی با تعداد کارها افزایش می‌یابد. علت این برتری نیز استفاده از تئوری صف، پردازش گراف و نیز تابع مناسب بهینه‌سازی است.



شکل۵) مقایسه زمان اجرا با تعداد وظایف (۵۰ PM و ۷۵ VM)



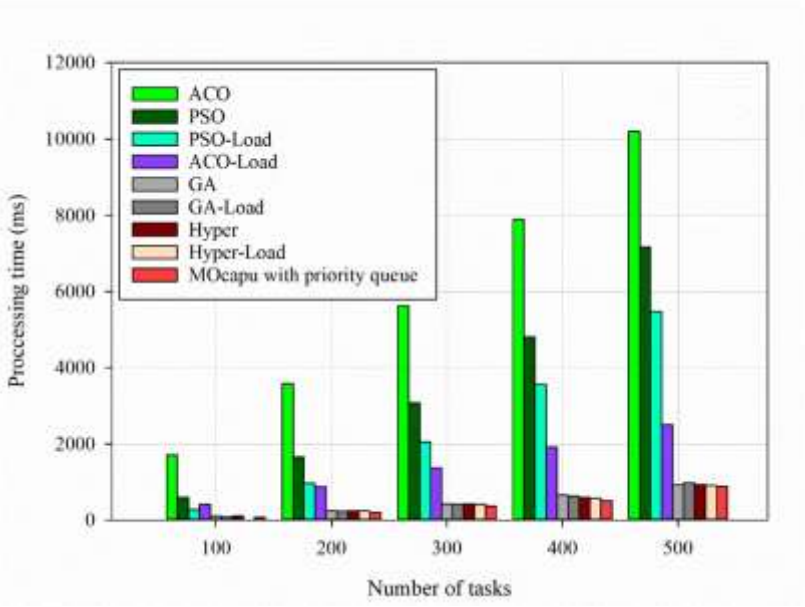
شکل۶) مقایسه زمان با تعداد وظایف (۱۰۰ PM و ۱۲۵ VM)



شکل۷) مقایسه زمان اجرا با تعداد وظایف (۱۵۰ PM و ۱۷۵ VM)

۴-۳-۲- زمان پردازش

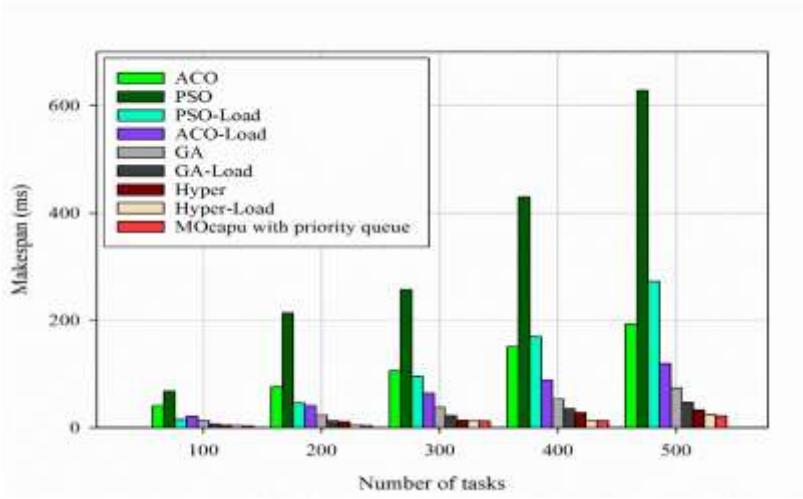
کل زمان پردازش تقریباً بازمان تکمیل کارها برابر است. زمان پردازش بین ماشین‌های مجازی بر اساس ظرفیت و پهنای باند آن‌ها متفاوت است. شکل ۸ زمان پردازش روش ما را با کارهای قبلی با نشان دادن مقدار راه‌حل بهینه با ۱۰۰-۵۰۰ کار مقایسه می‌کند. زمان پردازش برحسب میلی‌ثانیه محاسبه می‌شود. در اینجا، الگوریتم MOcapu پیشنهادی با ACO [1]، PSO [2]، PSO-Load [3]، ACO-Load [4]، GA [5]، GA-Load [6]، Hyper [7] موجود مقایسه شده است. الگوریتم‌های Hyper-Load [8]. در سناریوی زمان‌بندی ۳۰۰ کار، زمان صرف شده برای الگوریتم پیشنهادی، Hyper، Hyper-Load و GA-Load به ترتیب ۳۶۹ میلی‌ثانیه، ۴۲۲ میلی‌ثانیه، ۴۱۸ میلی‌ثانیه و ۴۱۲ میلی‌ثانیه است. به طور مشابه، هنگام برنامه‌ریزی ۵۰۰ کار، میانگین زمان پردازش به‌دست‌آمده برای الگوریتم MOcapu، Hyper، Hyper-Load و GA-Load به ترتیب ۸۸۶ ms، 929 ms، 9.4 ms و ۹۸۴ ms است. این نتایج برتری بالقوه الگوریتم ما را نسبت به کارهای قبلی نشان می‌دهد.



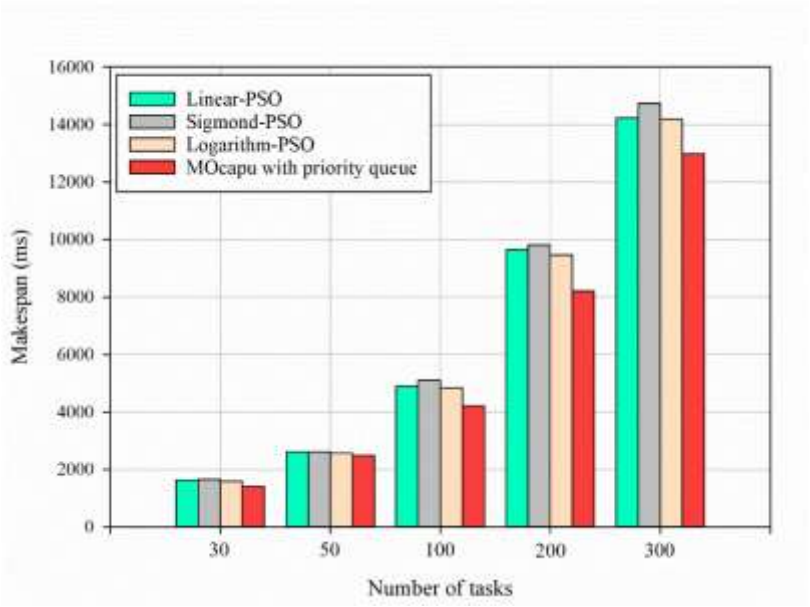
شکل۸) مقایسه زمان پردازش با تعداد کارها

۴-۳-۳- زمان اتمام کل کارها

زمان اتمام کل کارها، به فاصله زمانی بین لحظه‌ای که اولین کار برنامه‌ریزی می‌شود و لحظه‌ای که آخرین کار اجرا را کامل می‌کند، اشاره دارد. ویژگی‌های زمان اتمام کل کارها، با هر اجرای موازی وظایف کاهش می‌یابد. آزمایش‌های متعددی برای ارزیابی این پارامتر از نظر تعداد مختلف وظایف انجام شد. بر اساس گردش‌های کاری مختلف، شکل ۹ روش ما را با روش‌های قبلی در رابطه با زمان ساخت مقایسه می‌کند. در ابتدا، با تعداد کمی از وظایف، روش ما مانند روش‌های Hyper و Hyper-load عمل می‌کند، اما با افزایش تعداد وظایف، طول آن به مقدار کمی جمع می‌شود. شکل ۱۰ مقایسه‌ای از فاصله بین تکنیک ما و تکنیک‌های موجود برای ۱۰۰-۵۰۰ کار را نشان می‌دهد که در آن زمان تکمیل کار نهایی شامل هیچ سرباری در تکنیک ما نمی‌شود.



شکل۹) مقایسه زمان اتمام کل کارها با تعداد وظایف



شکل۱۰) مقایسه زمان اتمام کل کارها با تعداد وظایف

۴-۴- ارزیابی حاصل از نتایج شبیه‌سازی و مقایسه آن‌ها با سایر الگوریتم‌های فرااکتشافی

در این بخش نتایج به‌دست‌آمده برای روش پیشنهادی تحلیل شده است. روش پیشنهادی (الگوریتم کاپوچین چندهدفه با صف اولویت)، در حالات مختلف مورد ارزیابی قرار خواهد گرفت. به منظور ارزیابی بیشتر روش پیشنهادی با الگوریتم‌های ژنتیک GA[34]، الگوریتم بهینه‌سازی ازدحام ذرات PSO [۱۸]، زنبورعسل ABC [۱۷]، الگوریتم

دیفرانسیل تکاملی DEA [۳۵]، در معیار زمان اتمام کل کارها در سه سناریو مقایسه شده است. این سناریوها عبارت‌اند از: ۱) تغییر تعداد وظایف، ۲) تغییر تعداد ماشین‌های مجازی، ۳) تغییر هم‌زمان تعداد وظایف و ماشین‌های مجازی.

تنظیمات به کار گرفته‌شده در الگوریتم‌های فرااکتشافی مورداستفاده در این تحقیق برای ایجاد روش پیشنهادی در جدول‌های ۵ و ۶ نشان داده شده است. جدول ۵، نشان‌دهنده تنظیمات الگوریتم PSO و جدول ۶ مربوط به الگوریتم DEA است که بر اساس یک فرآیند آزمون‌وخطا به منظور تعیین بهترین پارامتر صورت گرفته است. در الگوریتم پیشنهادی تعداد عامل‌ها ۵۰ و تعداد منابع غذایی برای شکار کردن ۱۰۰ در نظر گرفته شده است.

جدول ۵) تنظیم پارامترهای الگوریتم PSO

نام پارامتر	مقدار پارامتر
تعداد ذره‌ها	۵۰
تعداد تکرار الگوریتم	۱۰۰
ضریب شتاب c_1	۰/۵
ضریب شتاب c_2	۰/۵
ضریب اینرسی ω	۰/۷۸

جدول ۶) تنظیم پارامترهای الگوریتم DEA

پارامتر	مقدار
جمعیت اولیه	۵۰
نوع انتخاب	نخبه‌گرایی و تصادفی
نوع ترکیب	تک نقطه‌ای
درصد ترکیب	۰.۴
نوع جهش	به طور تصادفی یکی از روش‌های جابه‌جایی یا درجا و یا معکوس کننده
درصد جهش	۰.۰۵
بیشترین تعداد تکرار	۲۰۰

۴-۴-۱- تغییر تعداد وظایف

به منظور تحلیل روش پیشنهادی به صورت جامع، تعداد وظایف موجود از ۲۵ تا ۳۰۰ سرویس افزایش داده شده است. خروجی موردنظر بر اساس زمان اتمام کل، بررسی شده است. جدول ۷ زمان اتمام کل و تعادل بار را برای روش پیشنهادی در ۲۵ وظیفه روی ابر را نشان می‌دهد. همچنین با الگوریتم‌های پایه PSO، DEA و GA و الگوریتم زنبورعسل ABC مقایسه شده است. نتایج جدول نشان می‌دهد که در تعداد وظایف پایین، روش پیشنهادی MOcapu، در زمان اتمام کل کارها و تعادل بار کارایی مشابهی با سایر الگوریتم‌ها دارد. به بیان دیگر در تعداد وظایف پایین، روش پیشنهادی در خصوص پارامتر زمان اتمام کل کارها و تعادل بار عملکردی نزدیک به الگوریتم ژنتیک با جمعیت اولیه تصادفی و الگوریتم ABC خواهد داشت. با توجه به اینکه روش پیشنهادی، الگوریتمی ترکیبی است، بدیهی است که در مواردی با تعداد وظایف کم مشابه با الگوریتم‌های دیگر عمل کرده است؛ اما از نقطه نظر تعادل بار، زمان اتمام کل کارها بهینه‌تر عمل کرده است.

جدول ۷) نتایج شبیه‌سازی با ۲۵ سرویس در روش‌های موردنظر

نوع الگوریتم	Makespan
DEA	۳۹.۲۶
ABC	۳۴.۷۶
PSO	۳۶.۷۶
GA	۳۳.۰۵
روش پیشنهادی	۳۰.۲۱

جدول ۸، نشان‌دهنده زمان اتمام کل برای روش پیشنهادی و ۴ الگوریتم موردنظر برای ۵۰ سرویس است. بر اساس نتایج ارائه‌شده در جدول ۵-۸، روش پیشنهادی به علت ترکیبی بودن دو الگوریتم به منظور بهینه‌سازی، بهترین عملکرد را ایجاد کرده است. زمـان اتمام کل در روش پیشنهادی نیز نسبت به الگوریتم‌های PSO و همچنین DEA کارایی بهتری دارد. بهبود عملکرد و همگرایی سریع‌تر، سبب شده است که زمان اتمام کل، نزدیک اما بهتر نسبت به دو الگوریتم ژنتیک و ABC داشته باشد. به بیان دیگر، با هر بار اجرای الگوریتم و افزایش تعداد وظایف، روش پیشنهادی روند بهتری به خود گرفته است. به صورت خلاصه، روش پیشنهادی با تعداد ۵۰ سرویس، از دیدگاه زمان اتمام کل کارها نسبت به الگوریتم‌های PSO و DEA کارایی بهتری را نشان داده است. تابع برازندگی مناسب و نوع عملکرد این الگوریتم و متناسب بودن با مسئله زمان‌بندی سبب این بهبود شده است.

جدول ۸) نتایج شبیه‌سازی با ۵۰ سرویس در روش‌های موردنظر

نوع الگوریتم	Makespan
DEA	۷۷.۶
ABC	۷۰.۱
PSO	۷۵.۹۳
GA	۶۸.۷۳
روش پیشنهادی	۶۱.۶۱

جدول ۹، زمان اتمام کل برای روش پیشنهادی و ۴ الگوریتم موردنظر برای ۱۲۵ سرویس نشان می‌دهد. از نتایج این جدول، می توان دریافت که با افزایش تعداد وظایف، روش پیشنهادی روند بهتری نسبت به سایر الگوریتم‌هایی مورد بحث دارد. به بیان دیگر، کاهش زمان اتمام کل کارها را بهتری نشان داده است. بر اساس نتایج جدول، الگوریتم DEA به علت در نظر نگرفتن وظایف با حداقل زمان و اولویت تخصیص منابع وظایف بر اساس ورود به ابر، بدترین کارایی را دارد. همچنین زمان اتمام کل نیز افزایش یافته است. این در حالی است که روش پیشنهادی با افزایش تعداد وظایف عملکرد و کارایی بهتری در خصوص کاهش زمان اتمام کل را دارد.

جدول ۹) نتایج شبیه‌سازی با ۱۲۵ سرویس در روش‌های موردنظر

نوع الگوریتم	**Makespan**
DEA	۱۹۳.۵۳
ABC	۱۸۱.۹
PSO	۱۸۵.۹۸
GA	۱۸۶.۱۴
روش پیشنهادی	۱۶۸.۱۴

جدول ۱۰، زمان اتمام کل کارها را برای روش پیشنهادی و ۴ الگوریتم موردبحث، بـرای ۲۰۰ سرویس نشان می‌دهد. بـر اساس نتایج این جدول، با افزایش تعداد وظایف، الگوریتم DEA باز هم بدترین عملکرد را ازنظر زمان اتمام کل کارها به خود اختصاص داده است؛ اما روش پیشنهادی عملکرد بهتری را با افزایش وظایف از خود نشان می‌دهد، الگوریتم بهینه‌سازی DEA وظایف را به ترتیب به منابع در دسترس، نگاشت می‌دهد. در این حالت احتمال اشغال بیش از حد یک

منبع و بیکاری منبع دیگر وجود دارد. این حالت باعث کاهش زمان اتمام کل کارها نیز شده است. روش پیشنهادی بر اساس نتایج به‌دست‌آمده و با توجه به تخصیص هدفمند منابع به وظایف، سبب کاهش زمان اتمام کل کارها شده است.

جدول ۱۰) نتایج شبیه‌سازی با ۲۰۰ سرویس در روش‌های موردنظر

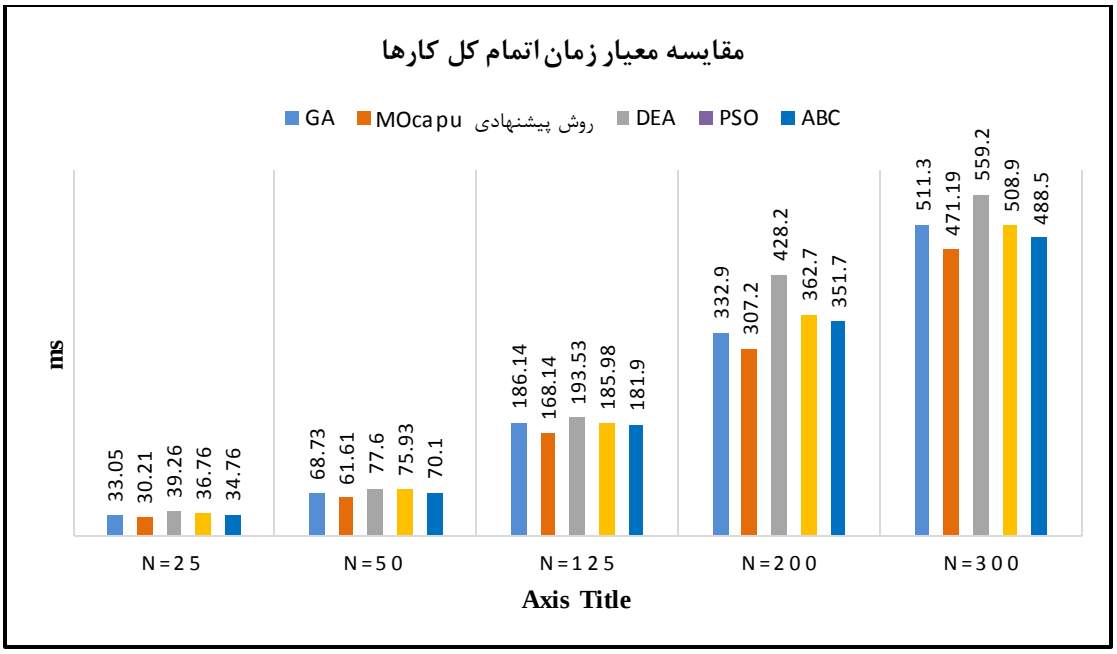
نوع الگوریتم	**Makespan**
DEA	**۴۲۸.۲**
ABC	**۳۵۱.۷**
PSO	**۳۶۲.۷**
GA	**۳۳۲.۹**
روش پیشنهادی	**۳۰۷.۲**

جدول ۱۱، زمان اتمام کل کارها در روش پیشنهادی و ۴ الگوریتم موردنظر، به ازای ۳۰۰ سرویس را نشان می‌دهد. نتایج این جدول، نشان‌دهنده برتری روش پیشنهادی با افزایش وظایف همانند حالات قبلی است؛ و با افزایش تعداد سرویس برتری روش پیشنهادی به‌خوبی ملموس است.

جدول ۱۱) نتایج شبیه‌سازی با ۳۰۰ سرویس در روش‌های موردنظر

نوع الگوریتم	**Makespan**
DEA	**۵۵۹.۲**
ABC	**۴۸۸.۵**
PSO	**۵۰۸.۹**
GA	**۵۱۱.۳**
روش پیشنهادی	**۴۷۱.۱۹**

شکل‌های ۱۱، نمودارهای میله‌ای را برای ۵ الگوریتم موردنظر شامل روش پیشنهادی، PSO، DEA، ABC و GA را در زمان اتمام کل کارها نشان می‌دهد.



شکل ۱۱) مقایسه معیار زمان اتمام کل کارها درروش پیشنهادی و سایر الگوریتم‌های موجود

با توجه به اینکه در محیط ابر، تعداد وظایف درخواستی توسط کاربران به صورت مداوم در حال افزایش است، بنابراین با افزایش تعداد وظایف، روش پیشنهادی بهترین بهبود را نشان داده است. همچنین با ارزیابی مناسب راه‌حل‌ها، هر چه مدت‌زمان اجرای وظایف روی ماشین‌ها مساوی و نزدیک به هم شود و تابع ارزیابی مقدار کم و نزدیک به صفر داشته باشد، از نظر زمانی، اتمام وظایف زودتر انجام می‌گردد. به بیان دیگر کاهش زمان اتمام کل کارها را در پی خواهد داشت.

جدول ۱۲، درصد بهبود زمان اتمام کل کارهای روش پیشنهادی را نسبت به سایر الگوریتم‌هایی بهینه‌سازی نشان می‌دهد. بر اساس نتایج ارائه‌شده در جدول، روش پیشنهادی نتیجه‌ای به نسبت بهتر در سایر روش‌ها دارد. بر اساس این جدول بهترین درصد زمان اتمام کل روش پیشنهادی، نسبت به الگوریتم DEA است. همچنین نتایج قابل‌قبول‌تری نسبت به سایر الگوریتم‌های موردبحث دارد.

جدول ۱۲) درصد بهبود پارامتر زمان اتمام کل کارها درروش پیشنهادی

نوع الگوریتم	۲۵ کار	۵۰ کار	۱۲۵ کار	۲۰۰ کار	۳۰۰ کار	درصد بهبودی
DEA	٪۳۰	٪۲۶	٪۱۵	٪۳۹	٪۱۹	**٪۲۵**
ABC	٪۱۵	٪۱۴	٪۸	٪۱۴	٪۴	**٪۱۱**
PSO	٪۲۲	٪۲۳	٪۱۱	٪۱۸	٪۸	**٪۱۶**
GA	٪۹	٪۱۲	٪۱۱	٪۸	٪۹	**٪۱۰**

۴-۴-۲- نتایج تغییر تعداد ماشین‌*های مجازی*

جهت بررسی دقیق‌تر روش پیشنهادی و میزان تأثیر تغییر تعداد ماشین‌های مجازی، تعداد وظایف ثابت نگه‌داشته شده و تعداد ماشین‌های مجازی از ۲ منبع به ۱۰ منبع افزایش یافته است. نتایج به‌دست‌آمده در شبیه‌سازی بر اساس افزایش تعداد ماشین‌های مجازی، در زمان‌بندی ۳۰۰ سرویس، برای روش پیشنهادی و الگوریتم‌های DEA، ABC، PSO و GA در جدول ۱۴ نشان داده شده است. در این جدول VMs نشان‌دهنده تعداد ماشین مجازی، LB نشان‌دهنده تعادل بار و MS نشان‌دهنده زمان اتمام کل است. ارزیابی روش پیشنهادی در مقیاس‌های مختلف و با استفاده از داده‌های واقعی مورد ارزیابی قرارگرفته است. نتایج جدول ۱۴ نشان می‌دهد که با افزایش تعداد منابع و ثابت نگه‌داشتن تعداد وظایف، همچنان روش پیشنهادی در خصوص تعادل بار و کاهش زمان اتمام کل، نسبت به سایر الگوریتم‌های موردبحث دارای برتری کارایی است. به بیان دیگر افزایش تعداد ماشین‌های مجازی به منظور تخصیص تعداد وظایف، تأثیر کمی درروند بهینه‌سازی الگوریتم پیشنهادی دارد.

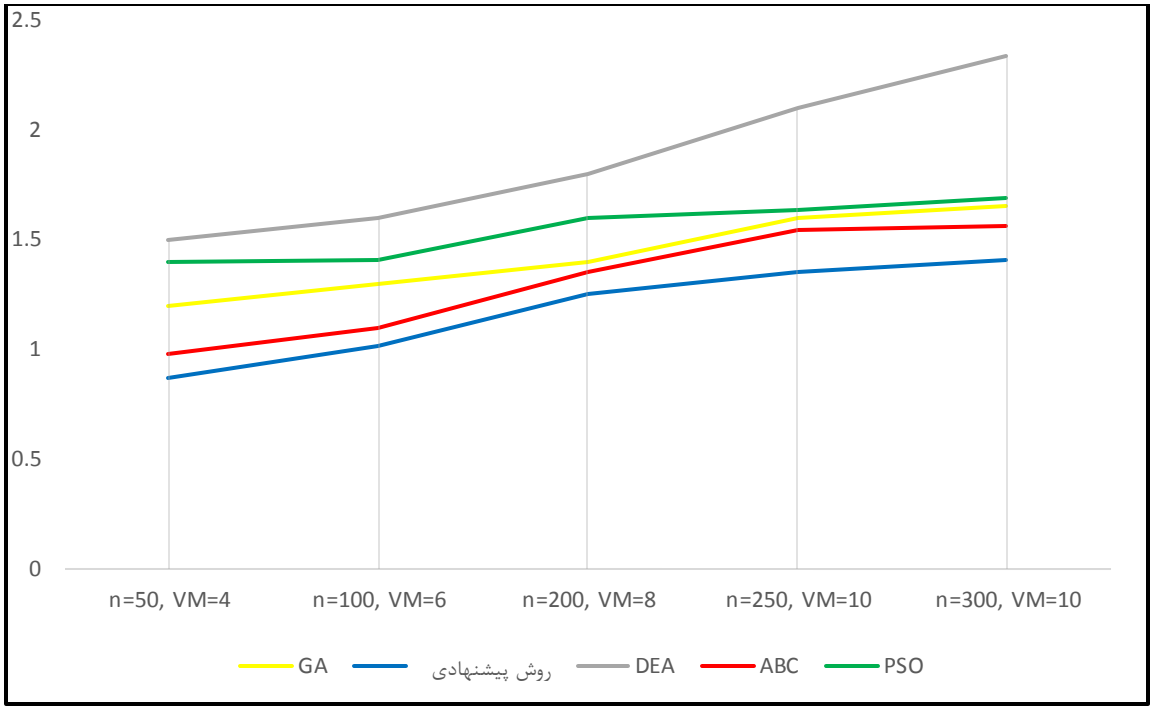
جدول ۱۴) نتایج شبیه‌سازی با تغییر تعداد ماشین‌های مجازی از ۲ تا ۱۰ ماشین

تعداد ماشین مجازی	DEA	ABC	PSO	GA	روش پیشنهادی
VMs=۲	۱۳۸۴	۱۲۵۲	۱۳۵۹	۱۲۵۸.۱	**۱۲۲۹.۱**
VMs=۵	۵۹۴.۲	۴۱۹.۲	۵۰۲.۳	۴۲۷.۴	**۳۸۹.۴**
VMs=۶	۴۹۳	۳۵۳.۵	۳۷۱.۶	۳۵۶.۳	**۳۴۳.۹**
VMs=۸	۳۳۳.۲۴	۲۴۸	۲۶۰	۲۵۱.۲	**۲۲۸.۱**
VMs=۱۰	۲۹۹.۳۵	۲۰۸.۵	۲۲۲.۷	۲۱۵.۰۳	**۱۹۷.۱۲**

۴-۴-۳- تغییر هم‌زمان تعداد وظایف و تعداد ماشین‌*های مجازی*

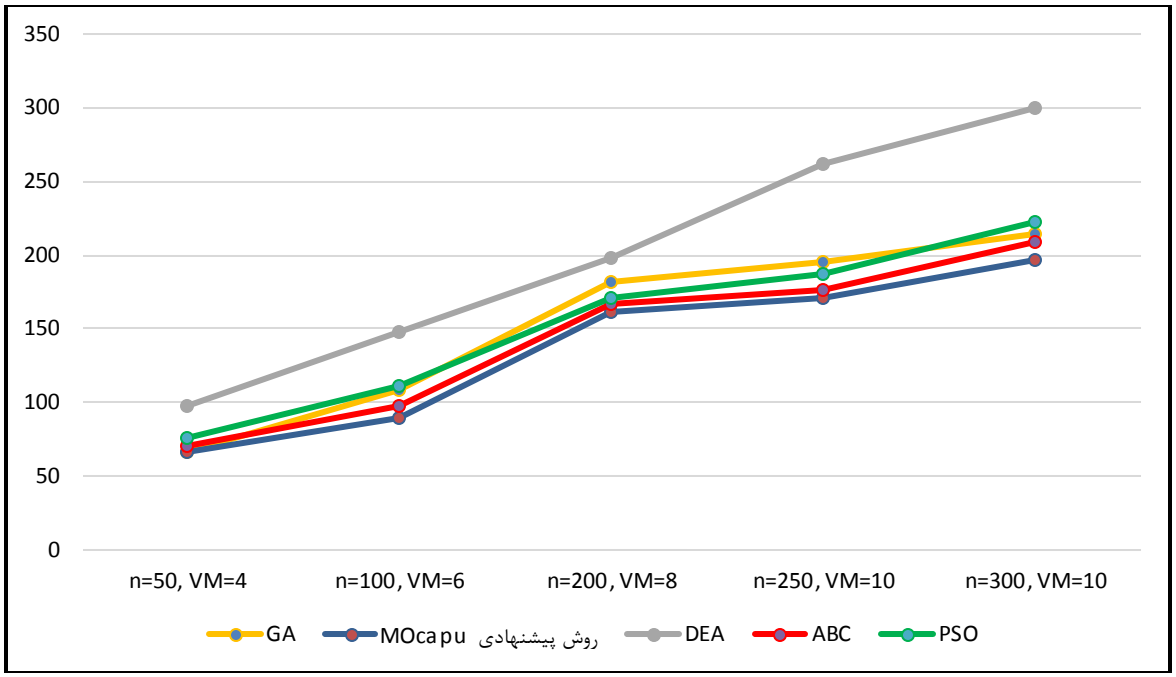
روش پیشنهادی و الگوریتم‌های DEA، ABC، PSO و GA در شرایطی یکسان و با تغییر هم‌زمان تعداد ماشین‌های مجازی و تعداد وظایف در محیط ابر مورد ارزیابی قرار گرفتند. نتایج حاصل از این ارزیابی برای معیار زمان اتمام کل کارها در نمودارهای شکل ۱۲ نشان داده شده است. الگوریتم‌های موردنظر در مقاله ۵ بار، با تغییر تعداد وظایف از ۵۰

سرویس به ۳۰۰ سرویس مورد ارزیابی قرار گرفت. جهت بررسی میزان تأثیر افزایش تعداد ماشین‌های مجازی هم‌زمان با تغییر وظایف، الگوریتم‌های موردبحث در تحقیق با ۴، ۶، ۸ و ۱۰ ماشین مجازی مورد آزمایش قرار گرفتند. نتایج به‌دست‌آمده نشان‌دهنده‌ی این است که افزایش تعداد وظایف هم‌زمان با افزایش تعداد ماشین‌های مجازی تأثیر زیادی بر کارایی روش پیشنهادی ندارد. با توجه به نتایج نمودار شکل ۸، روش پیشنهادی نسبت به سه الگوریتم PSO، DEA و GA، در با اختصاص بهینه منابع به وظایف، کاهش زمان اتمام کل کارها را در پی داشته است. روش پیشنهادی با تغییر هم‌زمان تعداد وظایف و افزایش تعداد ماشین‌های مجازی، راندمان تقریباً مشابه با بهبودی نسبی نسبت به الگوریتم ABC دارد.



شکل ۱۲) مقایسه پارامتر تعادل بار با تغییر هم‌زمان تعداد ماشین‌های مجازی و تعداد وظایف

بر اساس نتایج به‌دست‌آمده، روش پیشنهادی با تغییر هم‌زمان تعداد وظایف و افزایش تعداد ماشین‌های مجازی، کاهش زمان اتمام کل کارها اتفاق می‌افتد. شکل ۱۳ نتایج شبیه‌سازی پارامتر زمان اتمام کل کارها را بر اساس تغییر هم‌زمان تعداد کارها و ماشین‌های مجازی، نشان می‌دهد.



شکل ۱۳) مقایسه معیار زمان اتمام کل کارها با تغییر هم‌زمان تعداد ماشین‌های مجازی و تعداد وظایف

۵- نتیجه‌گیری

رایانش ابری به عنوان مدلی برای دسترسی و استفاده از منابع اشتراکی شناخته می‌شود. این منابع، می‌توانند شامل سرویس، کارهای پردازشی، ذخیره‌سازی و غیره باشند. رایانش ابری، مبتنی بر درخواست و تقاضا می‌باشد که در آن کاربران با ارسال درخواست خود به سرویس‌دهندگان، بخشی از منابع را در اختیار می‌گیرند. از آنجا که محیط ابر یک محیط پویا است و درخواست‌های کاربران در طول زمان متغیر هستند؛ لذا نیازمند روش زمان‌بندی بهینه‌ای هستیم که بتواند در زمان کمتر درخواست‌های بیشتری را پردازش کند. زمان‌بندی یکی از مسائل مهم برای بهبود بهره‌وری از تمام خدمات مبتنی بر ابر است. یکی از اهداف زمان‌بندی استفاده از منابع به طور مؤثر و به‌دست‌آوردن حداکثر بهره‌وری است. الگوریتم‌های زمان‌بندی باید وظایف را به‌گونه‌ای تنظیم کنند، که در آن تعادل بین بهبود عملکرد و کیفیت خدمات و درعین‌حال حفظ عدالت در میان وظایف باشد. در این مقاله یک روش زمان‌بندی ترکیبی متناسب با الگوریتم‌های فراابتکاری هوش جمعی ارائه شد. روش پیشنهادی در این مقاله شامل سه مرحله اصلی هست. در مرحله اول، وظایف کاربران در سه صف کارهای جدید، کمترین کار باقی‌مانده و کارهای ویژه اولویت‌بندی می‌شوند. در مرحله دوم، وظایف کاربران با استفاده از الگوریتم جست‌وجوی کاپوچین به مناسب‌ترین ماشین‌های فیزیکی تخصیص داده می‌شوند. نهایتاً در مرحله سوم، مناسب‌ترین پردازنده ماشین مجازی برای اجرای زیر وظایف هر وظیفه انتخاب می‌شود. ابتدا بر اساس تغییر

تعداد وظایف مورد ارزیابی قرار گرفتند، سپس با افزایش تعداد ماشین‌های مجازی و ثابت نگه‌داشتن تعداد وظایف مقایسه گردید. به منظور ارزیابی بیشتر، تغییر هم‌زمان تعداد وظایف و ماشین‌های مجازی نیز اجرا شد. نتایج مقایسه‌ها و ارزیابی‌ها نشان داد، روش پیشنهادی با استفاده از برازندگی مناسب، نسبت به الگوریتم‌های DEA، ABC، PSO و GA در شرایطی یکسان، عملکرد بهتری دارد. روش پیشنهادی، در زمان اتمام کل کارها توانسته است نتایج بسیار خوبی (کاهش ۲۵ درصدی) را کسب کند. همچنین این روش با ساختار متناسب با مسئله مکان‌یابی و یا تخصیص وظیفه سبب بهبود همگرایی و افزایش سرعت و رسیدن به پاسخ بهینه بوده است. نتایج پیاده‌سازی روش پیشنهادی و مقایسه آن با سایر روش‌ها نشان‌دهنده آن است؟ که الگوریتم پیشنهادی می‌تواند به کارایی نسبتاً خوبی در خصوص برقراری و تضمین کاهش زمان اجرا دست پیدا کند.

۶- منابع

[1] Sunyaev, A., Cloud computing, in Internet computing. 2020, Springer. p. 195-236.

[2] Srivastava, P. and R. Khan, A review paper on cloud computing. International Journal of Advanced Research in Computer Science and Software Engineering, 2018. 8(6): p. ۱۰۲۰-۷

[3] Tabrizchi, H. and M. Kuchaki Rafsanjani, A survey on security challenges in cloud computing: issues, threats, and solutions. The journal of supercomputing, 2020. 76(12): p. 9493-9532.

[4] Bohu, L., Z. Lin, and C. Xudong, Introduction to cloud manufacturing. ZTE Communications, 2020. 8(4): p. 6-9.

[5] Khemka, B., et al., Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system. Sustainable Computing: Informatics and Systems, (0).

[6] Xu, Y. ,et al., A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 2014. 270(0): p. 255-287

[7] Orhean, A.I., F. Pop, and I. Raicu, New scheduling approach using reinforcement learning for heterogeneous distributed systems. Journal of Parallel and Distributed

Computing, 2018. 117: p. 292-302.

[8] Arunarani, A., D. Manjula, and V. Sugumaran, Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems, 2019. 91: p. 407-415.

[9] Kumar, M., et al., A comprehensive survey for scheduling techniques in cloud computing. Journal of Network and Computer Applications, 2019. 143: p. 1-33.

[10] Bittencourt, L.F., et al., Scheduling in distributed systems: A cloud computing perspective. Computer science review, 2018. 30: p. 31-54.

[11] Khan, M.A., Scheduling for heterogeneous Systems using constrained critical paths. Parallel Computing, 2012. 38(4–5): p. 175-193.

[12] Daoud, M.I. and N. Kharma, A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2008. 68(4): p. 399-409.

[13] Ghafarian, T. and B. Javadi, Cloud-aware data intensive workflow scheduling on volunteer computing systems. Future Generation Computer Systems, 2015. 51: p. 87-97.

[14] Yang, B., et al. An utility-based job scheduling algorithm for cloud computing considering reliability factor. in 2011 international conference on cloud and service computing. 20 .۱۱ IEEE.

[15] Hussin, M., Y.C. Lee, and A.Y. Zomaya. Efficient energy management using adaptive reinforcement learning-based scheduling in large-scale distributed systems. in 2011 international conference on parallel processing. 2011. IEEE.

[16] Barrett, E ,.E. Howley, and J. Duggan, Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency and computation: practice and experience, 2013. 25(12): p. 1656-1674.

[17] Peng, Z., et al., Random task scheduling scheme based on reinforcement learning in cloud computing. Cluster computing, 2015. 18(4): p. 1595-1607.

[18] Abdullahi, M. and M.A. Ngadi, Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems,

2016. 56: p. 640-650.

[19] Kamalinia, A. and A. Ghaffari, Hybrid task scheduling method for cloud computing by genetic and DE algorithms. Wireless personal communications, 2017. 97(4): p. 6301-6323.

[20] Ben Alla, H., et al., A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. Cluster Computing, 2018. 21(4): p. 1797-1820.

[21] Mapetu, J.P.B., Z. Chen, and L. Kong, Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. Applied Intelligence, 2019. 49(9): p. 3308-3330.

[22] Senthil Kumar, A. and M. Venkatesan, Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment. Wireless Personal Communications, 2019. 107(4): p. 1835-1848.

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

[24] Shukri, S.E., et al., Enhanced multi-verse optimizer for task scheduling in cloud computing environments. Expert Systems with Applications, 2021. 168: p. 114230.

[25] Abualigah, L. and A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Computing, 2021. 24(1): p. 205-223.

[26] Panda, S.K., S.S. Nanda, and S.K. Bhoi, A pair-based task scheduling algorithm for cloud computing environment. Journal of King Saud University-Computer and Information Sciences, 2022. 34(1): p. 1434-1445.

[27] Abualigah, L. and M. Alkhrabsheh, Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing .The Journal of Supercomputing, 2022. 78(1): p. 740-765.

[28] Miettinen, K., Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science. 1999, Kluwer Academic Publishers, Dordrecht.

[29] Anuradha, V. and D. Sumathi. A survey on resource allocation strategies in cloud computing. in Information Communication and Embedded Systems (ICICES), 2014 International Conference on. 2014. IEEE.

[30] Xu, Y., et al., A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 2014. 270(0): p. 255-287.

[31] Topcuoglu, H., S. Hariri, and W. Min-You, Performance-effective and low-complexity task scheduling for heterogeneous computing. Parallel and Distributed Systems, IEEE Transactions on, 2002. 13(3): p. 260-۲۷۴.

[32] Alworafi, M.A. and S. Mallappa, An enhanced task scheduling in cloud computing based on deadline-aware model. International Journal of Grid and High Performance Computing (IJGHPC), 2018. 10(1): p. 31-53.

[33] Gupta, A., H. Bhadauria, and A. Singh, Load balancing based hyper heuristic algorithm for cloud task scheduling. Journal of Ambient Intelligence and Humanized Computing, 2021. 12(6): p. 5845-5852.