مقاله پژوهشی

حل مسئله بهینه‌سازی چندهدفه جایگذاری ماشین‌های مجازی در مراکز داده‌ای ابر با رویکرد کمینه‌سازی مصرف انرژی و هدررفت منابع با الگوریتم تبرید فلزات

میرسعید حسینی شیروانی

چکیده: در عصر حاضر، صنعت رایانش ابری به یک زنجیره تأمین جدید بین ارائه‌دهندگان سرویس محاسباتی و درخواست‌دهندگان سرویس تبدیل شده است. برای این منظور، مراکز داده‌ای ابر به طور گسترده از تکنولوژی مجازی‌سازی استفاده می‌کنند که به طور بالقوه قابلیت افزایش بهره‌وری منابع محاسباتی در سطح زیرساخت ابر را فراهم می‌کند. طرح‌های ناکارامد جایگذاری ماشین‌های مجازی منجر به کاهش بهره‌وری سیستم، افزایش هدررفت منابع و در نتیجه مصرف بالای انرژی در مراکز داده‌ای ابر می‌شوند. بنابراین، این مقاله مسئله جایگذاری ماشین‌های مجازی روی ماشین‌های فیزیکی مرکز داده‌ای ابر را به یک مسئله بهینه‌سازی چندهدفه با رویکرد کمینه‌سازی دو هدف مصرف انرژی و هدررفت منابع فرمول‌بندی می‌کند که از لحاظ محاسباتی در رده مسایل NP-hard قرار دارد. از آنجایی که اکثر الگوریتم‌های فراابتکاری برای حل مسایل بهینه‌سازی پیوسته طراحی شده‌اند و نیز کیفیت راه حل آنها با خطر گیرافتادن در بهینه محلی تهدید می‌شود، برای حل این مسئله ترکیبی و پیچیده، یک الگوریتم بهینه‌سازی مبتنی بر تبرید فلزات متناسب با فضای جستجوی گسسته تعریف‌شده در مسئله، توسعه داده می‌شود تا امکان گیرافتادن در بهینه محلی را کاهش دهد. جهت اعتبارسنجی روش پیشنهادی، سناریوهای مختلفی معرفی و هدایت می‌شوند. نتایج به دست آمده از شبیه‌سازی در سناریوهای مختلف، برتری روش پیشنهادی را نسبت به سایر روش‌های موجود از لحاظ کاهش مصرف انرژی، هدررفت منابع و تعداد سرویس‌دهنده‌های فعال نشان می‌دهد.

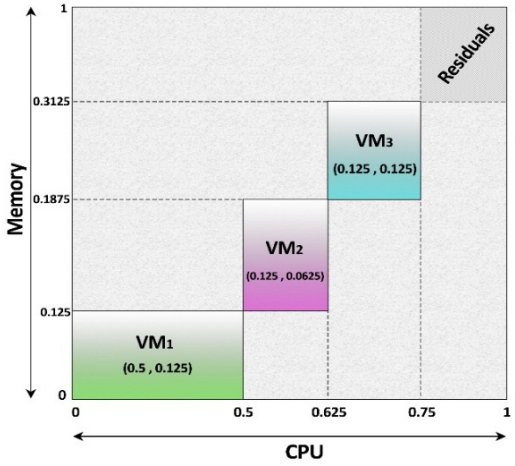
کلیدواژه: **رایانش ابری، مجازی‌سازی، جایگذاری ماشین مجازی، تبرید فلزات.**

۱- مقدمه

امروزه صنعت رایانش ابری به یک زنجیره تأمین جدید بین ارائه‌دهندگان سرویس محاسباتی و درخواست‌دهندگان سرویس تبدیل شده است [۱]. در صنعت رایانش ابری سرویس‌ها در قالب زیرساخت، بستر و برنامه‌های کاربردی ارائه می‌شوند و آمازون، گوگل و سیلزفورس به ترتیب در این حوزه‌ها جزء شاخص‌ترین ارائه‌دهندگان سرویس‌های ابر شناخته می‌گردند [۲]. در این راستا، سنگ بنای کار، به کارگیری تکنولوژی مجازی‌سازی است، به طوری که قابلیت اجرای هم‌زمان چندین ماشین مجازی را روی هر سرویس‌دهنده فیزیکی فراهم می‌کند [۳]. یک

این مقاله در تاریخ ۲۹ آبان ماه ۱۳۹۹ دریافت و در تاریخ ۶ فروردین ماه ۱۴۰۰ بازنگری شد.

میرسعید حسینی شیروانی (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، واحد ساری، دانشگاه آزاد اسلامی، ساری، ایران، (email: mirsaeid_hosseini@iausari.ac.ir).



شکل ۱: هم‌میزبانی چند ماشین مجازی روی یک سرویس‌دهنده فیزیکی.

ماشین مجازی، یک واحد پردازشی است که روی یک سرویس‌دهنده فیزیکی گمارده می‌شود تا نیازمندی‌های محاسباتی و پردازشی یک کاربر را از لحاظ زمان محاسبات، تعداد پردازنده و اندازه حافظه تأمین کند. این کار از طریق تسهیم‌سازی منابع فیزیکی[1] سرویس‌دهنده بین ماشین‌های مجازی امکان‌پذیر است، مگر این که پردازنده فیزیکی لایه زیرین با محدودیت منابع جهت پذیرش ماشین مجازی جدید مواجه شود. به عنوان مثال در شکل ۱، یک پردازنده فیزیکی را که قدرت پردازشی ۸۰۰ MIPS و حافظه‌ای برابر ۱۶ GB دارد در نظر بگیرید. به منظور سهولت درک مسئله، این منابع به شکل بردار منابع (۱ و ۱) نرمال‌سازی می‌شوند، به طوری که به ترتیب از چپ با راست به پردازش و حافظه اختصاص می‌یابد.

سه درخواست اولیه ماشین‌های مجازی با منابع مورد نیاز VM_1 (۲ GB، ۴۰۰ MIPS)، VM_2 (۱ GB، ۱۰۰ MIPS) و VM_3 (۲ GB، ۱۰۰ MIPS) توسط یک یا چند کاربر به کارگزار ابر اعلام می‌گردد که با توجه به ظرفیت سرویس‌دهنده فیزیکی مورد نظر به بردارهای منابع (۰/۱۲۵ و ۰/۵)، (۰/۰۶۲۵ و ۰/۱۲۵) و (۰/۱۲۵ و ۰/۱۲۵) نرمال‌سازی می‌شوند. با توجه به منابع موجود، ماشین‌های مجازی درخواست‌شده به همراه منابع مورد نیاز تخصیص داده می‌شوند. اگر درخواست چهارم VM_4 (۴ GB، ۵۰۰ MIPS) توسط کاربری جدید یا همان کاربران موجود اعلام شود، سرویس‌دهنده فیزیکی مزبور علی‌رغم دارابودن حافظه

1. Physical Resource Multiplexing

مکفی به دلیل منبع پردازشی ناکافی قادر به پذیرش درخواست جدید نیست. این به علت ناکارامدی طرح جایگذاری ماشین مجازی صورت می‌گیرد، جایی که تناسب منابع پردازشی و حافظه مصرفی در نظر گرفته نمی‌شود. با افزایش برنامه‌های کاربردی تحت وب مثل شبکه‌های اجتماعی، داده‌های حجیم، برنامه‌های هوشمند وسایل نقلیه، پردازش آنلاین تصاویر و ویدئو و غیره، مراکز ابری روزبه‌روز مدرن‌تر و پیچیده‌تر می‌شوند تا سرویس‌های درخواست‌شده را پوشش دهند [۴] تا [۶]. تجربه نشان می‌دهد کاربران در درازمدت، ارائه‌دهندگان ابری را که کیفیت خدمات پایینی ارائه می‌دهند رها می‌کنند. بنابراین ارائه‌دهندگان ابر باید در بازار باز رقابتی چندابری در کنار افزایش کیفیت خدمات، تعرفه سرویس‌ها را در حد قابل قبول حفظ کنند و به جای افزایش قیمت سرویس‌های پرکیفیت، در جهت کاهش هزینه‌های متغیر خود گام بردارند. هزینه‌های پشتیبانی مراکز داده‌ای به دو هزینه معمولاً ثابت سرمایه‌گذاری و متغیر عملیاتی تقسیم‌بندی می‌شوند که بخش بزرگی از هزینه متغیر عملیاتی مربوط به مصرف بالای برق مراکز داده‌ای ابر است [۷] و [۸]. به عنوان مثال، پیش‌بینی شده مصرف الکتریسته تجهیزات فناوری اطلاعات و ارتباطات مراکز ابری جهان برابر ۱۰ درصد مصرف الکتریسته کل دنیاست [۹] تا [۱۱]. به عنوان مثالی دیگر، مصرف برق مراکز داده‌ای ایالات متحده در سال‌های ۲۰۱۰ و ۲۰۱۴ به ترتیب ۱/۴ و ۱/۸ درصد مصرف کل برق این کشور بوده است [۹] تا [۱۱]. تحقیقات دیگری نشان می‌دهد که روند رو به رشد مصرف برق تا ۱۳ درصد تا سال ۲۰۳۰ افزایش خواهد یافت که هم هزینه‌های بالایی به ارائه‌دهندگان ابر تحمیل می‌کند و هم باعث تولید زیاد گازهای گلخانه‌ای و در نتیجه تغییرات اقلیمی می‌شود [۱۲]. برای حل این مشکل، جایگذاری بهینه ماشین‌های مجازی با یکپارچه‌سازی سرویس‌دهنده‌های فیزیکی ابر، یک روش امیدبخش جهت افزایش بهره‌وری منابع و کاهش تعداد سرویس‌دهنده‌های فعال است که فقط در بستر مجازی‌سازی محقق می‌شود. این روش در راستای کاهش هزینه سراسری و نیل به اهداف محاسبات سبز صورت می‌گیرد. مقالات مختلفی جهت جایگذاری بهینه ماشین‌های مجازی در محیط ابر با رویکرد کاهش مصرف انرژی ارائه شدند. اکثراً این مسئله را به مسئله انتزاعی بسته‌بندی اشیا[1] مدل کرده‌اند، به طوری که ظرفیت سرویس‌دهنده‌ها، حجم قابل تحمل بسته‌ها و ظرفیت پردازشی مورد نیاز ماشین‌های مجازی به حجم اشیا تناظر داده شده است. یکی از معروف‌ترین الگوریتم‌های ابتکاری حل این مسئله، الگوریتم اولین مناسب‌ترین مرتب‌شده کاهشی (FFD)[2] است [۱۳]. اولاً پژوهش‌های زیادی نشان می‌دهد این نوع الگوریتم‌ها حداکثر به ۱۱/۹ جواب بهینه دست پیدا خواهند کرد [۱۴]. ثانیاً نتایج حاصل از حل مسئله bin-packing تک‌بعدی و توجه صرف به منبع پردازشی و عدم توجه به منبع حافظه، تصمیمی پایدار نخواهد بود. یکی از روش‌های مدیریت مصرف انرژی مراکز داده‌ای ابر، اعمال تکنیک مهاجرت ماشین‌های مجازی است. با وجود این که طرح‌های مختلفی از مهاجرت ماشین‌های مجازی به طور گسترده در مراکز داده‌ای ابر به منظور مدیریت مصرف انرژی ارائه می‌شوند، اما این روش‌ها خود با چالش‌هایی همراه هستند، به طوری که اولاً منابع و زمان سیستم را به طور قابل ملاحظه‌ای مصرف می‌کنند و ثانیاً مصرف انرژی مهاجرت و بعضاً خطر تخطی سطح سرویس (SLAV)[3] را به سیستم تحمیل می‌کنند [۳]. مقاله حاضر، مسئله جایگذاری ماشین‌های مجازی ایستا را به طور خاص برای پروژه‌هایی مثل رندرکردن گرافیک، پردازش کلان داده‌های ایستا و غیره که الگوی مصرف تقریباً مشخصی دارند در نظر می‌گیرد. به همین منظور کاربر تقاضای خود را از لحاظ تعداد ماشین‌های مجازی درخواستی به صورت پیشگیرانه[4] مطرح می‌کند. در صورت نوسان بار سرویس‌دهنده‌ها، می‌توان با تنظیم متغیر آستانه حد بالا در سرویس‌دهنده‌ها از مهاجرت‌های غیر ضروری ماشین‌های مجازی جلوگیری کرد [۱۵]. روش‌های معمول دیگر، اعمال الگوریتم‌های فراابتکاری است. هرچند، الگوریتم‌های فراابتکاری مختلفی مثل الگوریتم ژنتیک، بهینه‌سازی ازدحام ذرات و کلونی مورچگان در این حوزه ارائه شده‌اند [۱۶] تا [۱۸]، اکثراً در بهینه محلی گرفتار شده و جواب‌های زیر بهینه ارائه می‌دهند. بنابراین نوآوری این مقاله شامل موارد زیر است:

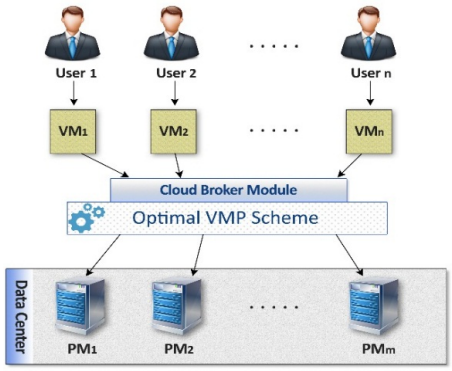
- ارائه دو مدل مصرف انرژی و هدررفت منابع مراکز داده
- مدل‌کردن مسئله جایگذاری ماشین‌های مجازی به یک مسئله بهینه‌سازی چندهدفه با رویکرد کمینه‌سازی مصرف انرژی و هدررفت منابع
- ارائه یک الگوریتم تبرید فلزات سفارشی‌سازی شده به همراه چند عملگر همسایگی جدید با توجه به فضای گسسته جستجو، جهت گریز از بهینه محلی و رسیدن به جواب بهینه سراسری

ادامه مقاله به صورت زیر سازمان‌دهی می‌شود: بخش ۲ به پیشینه پژوهش اختصاص داده می‌شود. بخش ۳ به چارچوب پیشنهادی و مدل‌های ارائه‌شده می‌پردازد. در بخش ۴ بیان مسئله انجام می‌شود. بخش ۵ به طور خلاصه الگوریتم فراابتکاری تبرید فلزات را معرفی می‌کند. بخش ۶ به ارائه الگوریتم پیشنهادی اختصاص دارد. ارزیابی کارایی الگوریتم پیشنهادی در بخش ۷ قرار می‌گیرد. بخش ۸ به جمع‌بندی و راهکارهای آینده خواهد پرداخت.

۲- پیشینه پژوهش

راه حل‌های کاندیدا برای حل مسئله جایگذاری ماشین‌های مجازی را می‌توان به سه دسته حریصانه، ابتکاری و فراابتکاری تقسیم‌بندی کرد. به عنوان نمونه، الگوریتم‌های اولین مناسب‌ترین مرتب‌شده کاهشی (FFD) [۱۳]، بهترین مناسب‌ترین مرتب‌شده کاهشی (BFD)[5] [۱۹] و بدترین مناسب‌ترین مرتب‌شده کاهشی (WFD)[6] [۲۰]، از معروف‌ترین الگوریتم‌های حریص در این حوزه هستند که سرویس‌دهنده‌ها را به بسته‌ها و توان پردازشی مورد نیاز ماشین‌های مجازی را به انواع اشیا متناظر می‌کنند. به عبارت دیگر، مسئله ذکرشده را به مسئله انتزاعی معروف ریاضی بسته‌بندی اشیا مدل می‌کنند. هدف همه آنها کاهش تعداد سرویس‌دهنده‌های فعال تا حد ممکن در راستای کاهش مصرف انرژی صورت می‌گیرد. پیچیدگی زمانی همه الگوریتم‌های یادشده برابر هزینه مرتب‌سازی $O(n \log n)$ است که اکثراً به راه حل‌های زیر بهینه منجر می‌شود [۱۴]. نکته مهم این است، به دلیل این که این الگوریتم‌ها بعد از اجرای عملگر بهترین انتخاب (طبق ملاک مشخص‌شده الگوریتم) قابلیت اصلاح اشتباه خود را ندارند، به جای الگوریتم‌های ابتکاری آنها را به الگوریتم‌های حریص نام‌گذاری می‌کنند. از طرف دیگر، روش ابتکاری

1. Bin-Packing
2. First Fit Decreasing Order
3. Service Level Agreement Violation
4. Proactively
5. Best Fit Decreasing Order
6. Worst Fit Decreasing Order



شکل ۲: چارچوب پیشنهادی.

معروف در این حوزه روش بازپرداخت[1] است [۲۱]. این الگوریتم، برخلاف روش حریص که برای تکمیل پاسخ نهایی طبق ملاک مشخصی، انتخابی از ورودی انجام می‌دهد قابلیت اصلاح اشتباه خود را دارد. همین موضوع باعث بالابودن نسبی زمان اجرای این نوع الگوریتم‌ها نسبت به الگوریتم‌های حریص مشابه است. یک الگوریتم انرژی-کارا برای زمان‌بندی ماشین‌های مجازی روی مراکز ابر توسط چاندیو و همکاران در سال ۲۰۱۹ ارائه شد [۲۱]. در این روش، در حین اجرای استراتژی اصلی الگوریتم به طور تناوبی به منظور بهبود جواب‌های زیر بهینه، هرچند یک بار ماشین‌های مجازی روی سرویس‌دهنده‌های مختلف جابه‌جا می‌شدند و این روش علی‌رغم کاهش مصرف انرژی، توجهی به بهبود هدررفت منابع نداشت. کار مشابهی توسط گیوری و همکاران انجام شد تا با یکپارچه‌سازی سرویس‌دهنده‌هایی که بهره‌وری پایین دارند در مصرف برق صرفه‌جویی صورت گیرد. علی‌رغم وجود الگوریتم‌های حریص و ابتکاری مفید در این حوزه، به دلیل افزایش فضای جستجو (مخصوصاً زمانی که تعداد ماشین‌های مجازی و سرویس‌دهنده‌های فیزیکی مقیاس بزرگ مراکز ابر بالاست)، این الگوریتم‌ها کارایی چندانی ندارند و باعث اتلاف منابع و مصرف بالای انرژی می‌شوند. بنابراین الگوریتم‌های فراابتکاری زیادی در این زمینه توسعه داده شده‌اند. یک روال خوشه‌بندی آگاه به شبکه جهت جایگذاری ماشین‌های مجازی در سطح زیرساخت توسط لینگنان و همکاران ارائه شد [۲۲]. هدف ارائه این روال استقرار ماشین‌های مجازی وابسته و پرترافیک در یک خوشه مشابه و نزدیک با توجه به توپولوژی شبکه بود، به طوری که میزان ترافیک ارسالی روی کانال‌های شبکه به حداقل ممکن برسد. برای ایجاد راه حل اولیه، از الگوریتم خوشه‌بندی K-means استفاده شد و سپس جهت بهبود کارایی آن از یک الگوریتم حریص پالایش‌شده استفاده گردید که توسط الگوریتم تبرید فلزات فراخوانی می‌شود. یونگ تیانگ و همکاران یک الگوریتم تبرید فلزات برای جایگذاری ماشین‌های مجازی به صورت انرژی کارا ارائه دادند. روش پیشنهادی آنها این مسئله را به مسئله انتزاعی دوبعدی بسته‌بندی اشیا مدل می‌کند [۲۳]. به طور مشابه، یک الگوریتم تبرید فلزات بهبودیافته برای حل مسئله جایگذاری ماشین‌های مجازی در محیط پویای ابر به منظور ایجاد توازن بار و افزایش بهره‌وری منابع ارائه شد [۲۴]. مطالعه در این حوزه نشان می‌دهد که در روش‌های بیان‌شده مبتنی بر تبرید فلزات به منظور کاوش فضای جستجو فقط از یک عملگر ساده همسایگی استفاده شده که باعث می‌شود جایگشت‌های متنوعی از فضای جستجو حاصل نشده و جواب‌های بالقوه کارا دیده نشوند. فرزای و همکاران یک الگوریتم فراابتکاری جایگذاری ماشین‌های مجازی چندهدفه را با رویکرد کاهش مصرف انرژی، هدررفت منابع و کاهش انتقال داده روی پهنای باند شبکه با الگوریتم ژنتیک ارائه دادند [۲۵]. این الگوریتم پیشنهادی، در حد امکان ماشین‌های مجازی وابسته به هم را از لحاظ فیزیکی در نزدیکی هم مستقر می‌کند تا از این طریق بار ارسالی به شبکه را کاهش دهد. ژائو و همکاران یک الگوریتم کلونی مورچگان جهت حل مسئله زمان‌بندی ماشین‌های مجازی جهت کاهش مصرف برق ارائه دادند [۱۸]. در این راستا، یک طرح جایگذاری مناسب ماشین مجازی جهت بهینه‌سازی بهره‌وری حافظه، پهنای باند و اندازه حافظه مصرفی ماشین‌های مجازی نیز با الگوریتم ازدحام ذرات ارائه شد [۱۷]. تانگ و پان یک مسئله بهینه‌سازی جایگذاری ماشین‌های مجازی جهت کاهش مصرف انرژی با در نظر گرفتن توان مصرفی سرویس‌دهنده‌ها و مؤلفه‌های شبکه ارائه دادند [۲۶]. به منظور اعتبارسنجی مدل پیشنهادی، یک الگوریتم ژنتیک ترکیبی جهت بهبود جواب‌ها با رویکرد محلی به کار گرفته شد.

اکثر کارهای مطالعه‌شده نشان می‌دهد که جایگذاری ماشین‌های مجازی بدون در نظر گرفتن توازن منابع مصرفی و در نتیجه هدررفت منابع که خود باعث به کارگیری سرویس‌دهنده‌های بیشتر می‌شود، باعث افزایش مصرف بالای انرژی در مراکز ابر خواهند شد. از طرفی اکثر الگوریتم‌های فراابتکاری در دام بهینه محلی گرفتار می‌شوند و از طرفی دیگر، توجهی به ذات گسسته مسئله جایگذاری ماشین مجازی ندارند. بنابراین پژوهش حاضر یک الگوریتم تبرید فلزات چندهدفه را به همراه چند عملگر جدید که آگاه به جایگشت‌های فضای حالت جستجو هستند در راستای پوشش ضعف‌های پژوهش این حوزه ارائه می‌دهد.

۳- چارچوب پیشنهادی و مدل‌ها

شکل ۲ چارچوب پیشنهادی پژوهش حاضر را به تصویر می‌کشد که در آن کاربران، n درخواست مختلف ماشین مجازی را که شامل پیکربندی خاصی از منابع پردازشی و حافظه‌ای است به کارگزار ابر ارائه می‌دهند. ماژول طرح بهینه‌سازی جایگذاری ماشین‌های مجازی که بخشی از کارگزار ابر است، قصد تخصیص منابع فیزیکی m سرویس‌دهنده‌های همگن موجود مرکز داده‌ای را به این درخواست‌ها دارد به طوری که اهداف سیستم مورد نظر محقق شوند. این مسئله از لحاظ انتزاعی به یک گراف دوبخشی تبدیل می‌شود و پیچیدگی زمانی الگوریتم ساده‌لوحانه[2] برای حل این مسئله زمان $O(m^n)$ صرف می‌کند که به وضوح از نوع NP-hard است [۳].

در این بخش دو مدل مصرف انرژی و هدررفت منابع مراکز داده‌ای نیز ارائه می‌شوند. جدول ۱، لیست نمادهای استفاده‌شده در مدل‌های ارائه‌گردیده این مقاله را به همراه توصیف آنها نمایش می‌دهد.

۳-۱ مدل مصرف انرژی مراکز داده‌ای

از آنجایی که بخش بزرگی از مصرف انرژی سرویس‌دهنده‌های مراکز داده‌ای ابر توسط اجرای دستورات پردازنده‌ها صرف می‌شود، بنابراین یک مدل مصرف انرژی متناسب با بهره‌وری پردازنده سرویس‌دهنده ارائه می‌گردد. مطالعات گسترده نشان می‌دهد مصرف انرژی پردازنده یک سرویس‌دهنده رابطه خطی با بهره‌وری پردازشی آن دارد. از طرفی در محیط مجازی ابر، بهره‌وری یک سرویس‌دهنده از مجموع بهره‌وری ماشین‌های مجازی که روی این پردازنده فیزیکی به طور هم‌زمان مستقر

1. Backfilling
2. Naïve

جدول ۱: نمادهای استفاده‌شده جهت مدل‌سازی مسئله.

نماد	توصیف
n	تعداد ماشین‌های مجازی درخواستی
m	تعداد ماشین‌های فیزیکی همگن ابر
VM_i	ماشین مجازی i ام
PM_j	ماشین فیزیکی j ام
PM_j^{CPU}	بهره‌وری پردازشی ماشین فیزیکی j ام
PM_j^{Mem}	بهره‌وری حافظه ماشین فیزیکی j ام
VM_i^{CPU}	پهنای باند پردازشی مورد نیاز درخواستی برای ماشین مجازی i ام
VM_i^{Mem}	پهنای باند حافظه مورد نیاز درخواستی برای ماشین مجازی i ام
PM_j^{Rcpu}	پهنای باند پردازشی مازاد ماشین فیزیکی j ام
PM_j^{Rmem}	پهنای باند حافظه مازاد ماشین فیزیکی j ام
PM_j^{D}	هدررفت منابع ماشین فیزیکی j ام
$PW(PM_j)$	توان مصرفی ماشین فیزیکی j ام
$PW(PM_j^{Full})$	توان مصرفی ماشین فیزیکی j ام در زمان بهره‌وری کامل
θ_j	درصدی از توان مصرفی ماشین فیزیکی j ام در زمان بهره‌وری کامل که همین ماشین در زمان بیکاری صرف می‌کند.
ω_i	ضریب اهمیت منبع i ام در سیستم
λ_j	ضریب اهمیت هدف i ام در سیستم
y_j	متغیر تصمیم جهت فعال‌بودن ماشین فیزیکی j ام
$Threshold$	آستانه تحمل بهره‌وری منابع

هستند محاسبه می‌شود. به عنوان مثال، بهره‌وری پردازشی سرویس‌دهنده نمایش داده شده در شکل ۱، برابر ۷۵ درصد است که از مجموع بهره‌وری ماشین‌های مجازی مستقر محاسبه شده است. معادله (۱) میزان مصرف انرژی پردازنده j ام را که متناسب با بهره‌وری پردازشی آن است محاسبه می‌کند

$$PW(PM_j) = \theta_j \times PW(PM_j^{Full}) + (1-\theta_j) \times PW(PM_j^{Full}) \times PM_j^{CPU} \quad (1)$$

بنابراین مصرف انرژی کل مرکز داده‌ای ابر از (۲) قابل محاسبه است

$$F_1 = \sum_{j=1}^{m} PW(PM_j).y_j \quad (2)$$

که در آن متغیر تصمیم دودویی y_j برای تعیین فعال‌بودن یا غیر فعال‌بودن پردازنده استفاده می‌شود. نکته مهم این است که پردازنده فیزیکی غیر فعال را در حالت خواب قرار می‌گیرد تا بلافاصله در صورت نیاز به حالت فعال تبدیل شده و در خدمت مرکز ابر باشد. از آنجایی که مصرف انرژی سرویس‌دهنده غیر فعال تقریباً ناچیز است، به همین دلیل متغیر تصمیم مربوط به صفر مقداردهی می‌شود و در غیر این صورت مقدار متغیر تصمیم یک در نظر گرفته می‌شود. تابع F_1 به عنوان هدف اول این مقاله است که باید مقدار آن کمینه‌سازی شود.

۳-۲ مدل هدررفت منابع مرکز داده‌ای

از آنجایی که تنوع زیادی در پیکربندی ماشین‌های مجازی از لحاظ منابع درخواستی مثل منبع پردازشی (با واحد MIPS) و منبع حافظه (با واحد MB) وجود دارد، جایگذاری غیر متوازن ماشین‌های مجازی روی ماشین‌های فیزیکی باعث بالارفتن میزان هدررفت منابع، کاهش بهره‌وری سیستم و استفاده بیش از حد در تعداد سرویس‌دهنده‌های فیزیکی فعال شده و در نتیجه سبب افزایش مصرف انرژی خواهد گردید. به همین منظور، در این بخش یک مدل هدررفت منبع در محیط ابر ارائه خواهد شد، به طوری که ماژول زمان‌بند ماشین‌های مجازی باید آگاه به کاهش هدررفت منابع نیز باشد. هدررفت منابع ماشین فیزیکی j ام از طریق (۳) قابل محاسبه است. به منظور بدون بعد کردن منابع، پارامترهای نرمال‌شده در معادله قرار می‌گیرند

$$PM_j^{D} = \frac{\left|\omega_1.PM_j^{Rcpu} - \omega_2.PM_j^{Rmem}\right| + \varepsilon}{\omega_1.PM_j^{CPU} + \omega_2.PM_j^{RAM}} \quad (3)$$

که در آن پارامترهای ω_1، ω_2 و ε به ترتیب به اهمیت منبع پردازشی، اهمیت منبع حافظه در سیستم و یک عدد بسیار کوچک اشاره دارند. در ضمن، پارامتر ε به مقدار کوچک ۰/۰۰۰۱ ارزش‌دهی می‌شود. بهره‌وری پردازشی و حافظه برای ماشین فیزیکی j ام به ترتیب از طریق (۴) و (۵) به دست می‌آیند

$$PM_j^{CPU} = \sum_{i=1}^{n} VM_i^{CPU}.x_{ij} \quad (4)$$

$$PM_j^{Mem} = \sum_{i=1}^{n} VM_i^{Mem}.x_{ij} \quad (5)$$

متغیر باینری تصمیم x_{ij} به این اشاره دارد که آیا ماشین مجازی VM_i روی ماشین فیزیکی PM_j جایگذاری شده است یا خیر. در این راستا، باید قیودی نیز تعریف و به مسئله اضافه شوند. به عنوان مثال، هر ماشین مجازی فقط روی یک ماشین فیزیکی جایگذاری می‌شود. در ضمن، هر سرویس‌دهنده فیزیکی نمی‌تواند بیش از آستانه بهره‌وری پردازشی و بهره‌وری حافظه تعیین‌شده پذیرای ماشین مجازی مازاد باشد. بنابراین، قیودی که در (۶) و نامعادلات (۷) و (۸) دیده می‌شوند، به این نکات اشاره دارند

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} = 1 \quad (6)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} VM_i^{CPU} \times x_{ij} \le Threshold \times y_j \quad (7)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} VM_i^{Mem} \times x_{ij} \le Threshold \times y_j \quad (8)$$

همچنین پهنای باند مازاد بهره‌وری پردازشی و حافظه برای ماشین فیزیکی j ام به ترتیب از طریق (۹) و (۱۰) به دست می‌آیند که در محاسبه هدررفت منابع استفاده می‌شوند

$$PM_j^{Rcpu} = Threshold - PM_j^{CPU} \quad (9)$$

$$PM_j^{Rmem} = Threshold - PM_j^{Mem} \quad (10)$$

بنابراین هدررفت کل منابع مرکز داده‌ای از (۱۱) محاسبه می‌شود. تابع F_2 یکی دیگر از اهدافی است که باید کمینه‌سازی شود

$$F_2 = \sum_{j=1}^{m} \frac{\left|\omega_1.PM_j^{Rcpu} - \omega_2.PM_j^{Rmem}\right| + \varepsilon}{\omega_1.PM_j^{CPU} + \omega_2.PM_j^{Mem}} \cdot y_j \quad (11)$$

از آنجایی که در این مدل منابع پردازشی و حافظه دارای اهمیت یکسانی هستند، ضرایب ω_1 و ω_2 در این مقاله برابر ۰/۵ در نظر گرفته می‌شوند، مگر این که در برنامه کاربردی خاصی ارزش هر منبع متفاوت باشد که در این صورت طراح، بسته به اهمیت منابع در آن برنامه کاربردی ضرایب مناسبی برای آنها در نظر می‌گیرد.

Algorithm 1. Virtual Machine Placement using Simulated Annealing (VMPSA)

Input: n , m , T_0 , $Freeze$, $MaxIteration$, ΔT

Output: An optimal VMP

1. $S[1..n]$ as a candidate solution
2. $S[1..n] \leftarrow \text{Random}[1..m]$
3. $S[1..n] \leftarrow \text{Check\&Correct}(S)$
4. $K \leftarrow 0, T_K \leftarrow T_0, Iteration \leftarrow 0$
5. while $T_K > Freeze$ do
6. $\sigma(S) \leftarrow \lambda_1.F_1(S) + \lambda_2.F_2(S)$ based on equation (12)
7. $Iteration \leftarrow Iteration + 1$
8. $R \leftarrow$ Draw an integer in [1..3]
9. if $R == 1$ then
10. $S'[1..n] \leftarrow \text{Shuffle}(S)$
11. else if $R == 2$ then
12. $S'[1..n] \leftarrow \text{Reverse}(S)$
13. else
14. $S'[1..n] \leftarrow \text{Mutate}(S)$
15. end if
16. $S'[1..n] \leftarrow \text{Check\&Correct}(S')$
17. $\sigma(S') \leftarrow \lambda_1.F_1(S') + \lambda_2.F_2(S')$ based on equation (12)
18. $\Delta F = \sigma(S') - \sigma(S)$
19. if $\Delta F < 0$ then
20. $S[1..n] \leftarrow S'[1..n]$
21. else
22. $\varepsilon \leftarrow$ Draw a real number in $\sim U(0,1)$
23. if $e^{-\frac{\Delta F}{T_K}} > \varepsilon$ then
24. $S[1..n] \leftarrow S'[1..n]$
25. end if
26. end if
27. if $Iteration > MaxIteration$
28. $T_K = T_K - \Delta T$
29. $K \leftarrow K + 1$
30. $Iteration \leftarrow 0$
31. end if
32. end while
33. return S as an optimal VMP

شکل ۳: الگوریتم پیشنهادی جایگذاری ماشین مجازی بر مبنای تبرید فلزات.

٤- بیان مسئله

در این مقاله، مسئله جایگذاری ماشین‌های مجازی به یک مسئله بهینه‌سازی چندهدفه با رویکرد کمینه‌سازی مصرف انرژی و هدررفت منابع فیزیکی مدل می‌شود. به دلیل متفاوت‌بودن نوع منابع، واحد آنها و اهداف، بعد از نرمال‌سازی پارامترها مسئله بهینه‌سازی مورد نظر توسط (۱۲) به یک مسئله بهینه‌سازی برداری تک‌هدفه تبدیل می‌شود، به طوری که اثر هر دو هدف در آن بهینه شوند. از آنجایی که اهمیت هر دو هدف در این مقاله یکسان در نظر گرفته می‌شوند، ضرایب λ_1 و λ_2 مساوی و برابر ۰/۵ فرض می‌شوند، مگر این که در هر سیستم با توجه به حساسیت اهداف، ضرایب جدید تعیین شوند

$$\min(F = \lambda_1 \cdot F_1 + \lambda_2 \cdot F_2) \qquad (۱۲)$$

مسئله اصلی، حل (۱۲) است که باید بهینه شود، منوط به این که قیود مطرح‌شده در (۶) و نامعادلات (۷) و (۸) رعایت شوند.

٥- الگوریتم فراابتکاری تبرید فلزات

الگوریتم تبرید فلزات الهام‌گرفته از عملیات تبرید در علم متالورژی است. به طوری که برای افزایش استحکام فلزات آنها را تا درجه ذوب

جدول ۲: کدگذاری راه حل کاندیدا.

شماره ماشین مجازی VM	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
شماره ماشین فیزیکی VM	۴	۱	۲	۱	۴	۴	۱	۱	۳	۳

حرارت می‌دهند و سپس تدریجاً درجه حرارت آن را کاهش می‌دهند تا به نقطه انجماد نزدیک شود. در این حالت، فلز در مستحکم‌ترین حالت ترمودینامیک خود قرار می‌گیرد [۲۷]. الگوریتم تبرید فلزات، تنها الگوریتمی است که برای حل مسایل مقیاس بزرگی که فضای جستجوی گسسته دارند مناسب و کارامد عمل می‌کند. در ضمن، این الگوریتم برخلاف سایر الگوریتم‌های فراابتکاری در بهینه محلی گیر نمی‌کند و حتی شانس پذیرش جواب‌های بد را حداقل برای مدتی امتحان می‌کند که آیا از آن طریق می‌تواند به مسیرهای بهتری دست یابد یا خیر. البته این الگوریتم در پایان اجرا، بیشتر به سمت جواب‌های خوب‌تر همگرا می‌شود. الگوریتم تبرید فلزات چهار عملگر اصلی دارد: عملگر اول وظیفه تولید جواب تصادفی اولیه از فضای جستجو را به عهده دارد. عملگر دوم که معروف به عملگر همسایگی است وظیفه تولید یک جواب تصادفی را در همسایگی راه حل جاری به عهده دارد که این جواب یکی از جایگشت‌های تصادفی جواب جاری است. عملگر سوم، معروف به تابع احتمال پذیرش همسایگی نام دارد. این عملگر طوری تنظیم می‌شود که اگر راه حل جدید که از عملگر همسایگی حاصل شد بهتر از راه حل جاری بود، ۱۰۰٪ پذیرش آن انجام شود، ولی در غیر این صورت پذیرش راه حل جدید بد، احتمالی انجام شود. این تابع احتمال، طوری سازمان‌دهی می‌گردد که شانس پذیرش جواب‌های بد در ابتدای الگوریتم خیلی محتمل‌تر از پایان اجرای الگوریتم باشد. به عبارت دیگر، هرچه الگوریتم به پایان نزدیک‌تر می‌شود، شانس پذیرش جواب‌های بد به صفر نزدیک‌تر شود و جواب‌ها به سمت بهینه سراسری همگرا شوند. برای این منظور سفارشی‌کردن تابع $y = e^{-1/x}$ در راستای حل مسئله خیلی مناسب است، چرا که برای ورودی‌های بزرگ (درجه حرارت بالا و نزدیک ذوب) مقدار این تابع نزدیک ۱ خواهد بود؛ همان جایی که پذیرش جواب بد محتمل‌تر است و برای ورودی‌های کوچک (درجه حرارت پایین و نزدیک انجماد) مقدار این تابع به صفر نزدیک خواهد شد، همان جایی که تقریباً پذیرش جواب بد نامحتمل‌تر است. بنابراین این تابع گزینه فوق‌العاده مناسبی برای عملگر سوم الگوریتم تبرید فلزات است. عملگر چهارم، به عملگر زمان‌بندی تبرید شهرت دارد. بعد از اجرای عملگرهای دوم و سوم، این عملگر به تدریج درجه حرارت فلز را کاهش می‌دهد. شرط پایان این الگوریتم رسیدن به درجه صفر فرضی یا انجماد است [۲۷].

٦- الگوریتم پیشنهادی حل مسئله جایگذاری ماشین مجازی بر مبنای تبرید فلزات

الگوریتم پیشنهادی جایگذاری ماشین مجازی با استفاده از الگوریتم تبرید فلزات در شکل ۳ به نمایش درآمده است. در این الگوریتم هر راه حل در قالب یک رشته n (تعداد ماشین‌های مجازی درخواستی) کاراکتری عین کروموزوم در الگوریتم ژنتیک کدگذاری می‌شود. هر کاراکتر (عین ژن در کروموزوم الگوریتم ژنتیک) می‌تواند مقدار عددی صحیح از ۱ تا m (تعداد سرویس‌دهنده‌های فیزیکی) بپذیرد [۲۸] و [۲۹]. به عنوان مثال، فرض کنید ۱۰ ماشین مجازی درخواست‌شده قرار است روی چهار سرویس‌دهنده فیزیکی مستقر شوند. رشته عددی جدول ۲،

جدول ۳: راه حل جاری S قبل از اجرای عملگر SHUFFLE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۲	۱	۴	۴	۱	۱	۳	۳

جدول ۴: راه حل جدید S' بعد از اجرای عملگر SHUFFLE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۴	۱	۲	۱	۴	۱	۳	۳

جدول ۵: راه حل جاری S قبل از اجرای عملگر REVERSE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۲	۱	۴	۴	۱	۱	۳	۳

جدول ۶: راه حل جدید S' بعد از اجرای عملگر REVERSE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۱	۴	۴	۱	۲	۱	۳	۳

نشان می‌دهد ماشین‌های مجازی VM_1، VM_5 و VM_6 روی سرویس‌دهنده فیزیکی PM_4 مستقر می‌شوند.

با توجه به جزئیات الگوریتم ۱ که در شکل ۳ قابل مشاهده است، این الگوریتم با مقداردهی اولیه و تولید جواب تصادفی آغاز می‌شود. پارامتر K مرحله اجرای اصلی برنامه، پارامتر T_k درجه حرارت در مرحله Kام و متغیر *Iteration* شماره جستجو در هر درجه حرارت خاص را نشان می‌دهد. به منظور افزایش کارایی در هر درجه حرارت ثابت، تعدادی جستجو (به تعداد *MaxIteration*) انجام می‌شود تا جواب‌های جاری را بهبود دهد و به نوعی این پارامتر وظیفه ایجاد توازن بین شناسایی[1] و بهره‌برداری[2] در فضای جستجو را به عهده دارد [۳۰]. در ضمن، پارامتر T_k به یک عدد بسیار بزرگ مثل T_0 (نقطه ذوب) مقداردهی می‌شود. جواب اولیه تصادفی در خط ۲ الگوریتم تولید می‌شود. نکته مهم این است که بعد از تولید جواب تصادفی و یا اعمال عملگرهای همسایگی روی جواب فعلی، ممکن است یک کروموزوم نامعتبر تولید شود. در این مسئله، زمانی که روی سرویس‌دهنده، بار کاری مازاد قرار گیرد که بهره‌وری آن از آستانه تعیین‌شده عبور کند سرریز اتفاق افتاده و کروموزوم مورد نظر (راه حل کدشده) باید بررسی و اصلاح شود. به همین منظور بعد از تولید جواب تصادفی اولیه و عملگر همسایگی، تابع Check&Correct فراخوانی می‌شود (مثل خطوط ۳ و ۱۶ الگوریتم). این تابع یک لیست پیوندی حلقوی از پردازنده‌ها ایجاد می‌کند و در صورت تشخیص‌دادن سرریز یک سرویس‌دهنده، بار مازاد را ترجیحاً روی یک سرویس‌دهنده فعال موجود که فضای مکفی هم از لحاظ پردازشی و هم از لحاظ حافظه دارد تخلیه می‌کند. در غیر این صورت یک سرویس‌دهنده غیر فعال را به حالت فعال تبدیل و بار مازاد را روی آن مستقر می‌کند و بدین ترتیب یک کروموزوم معتبر حاصل می‌شود. بعد از تولید جواب اولیه معتبر S حلقه اصلی برنامه آغاز به کار می‌کند. این اجرا از درجه حرارت خیلی بالا آغاز شده تا به درجه انجماد برسد که در شرط حلقه while در خط ۵ الگوریتم قابل ملاحظه است. در خط ۶ الگوریتم، مقدار تابع برازش راه حل S طبق (۱۲) محاسبه می‌شود. در این الگوریتم تابع $\sigma(.)$ در خط ۶ و ۱۷ به منظور تعیین برازش در نظر گرفته شده است. این تابع طوری تنظیم شده

1. Exploration
2. Exploitation

جدول ۷: راه حل جاری S قبل از اجرای عملگر MUTATE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۲	۱	۴	۴	۱	۱	۳	۳

جدول ۸: راه حل جدید S' بعد از اجرای عملگر MUTATE.

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۴	۱	۱	۱	۴	۴	۲	۱	۳	۳

که تأثیر هر دو تابع هدف مسئله به طور یکسان در آن لحاظ شده باشند. خطوط ۹ تا ۱۵ به عملگر همسایگی اختصاص داده شده‌اند. برای این که پیداکردن جواب همسایگی به طور جانبدارانه انجام نشود و حس تصادفی الگوریتم فراابتکاری حفظ شود، سه عملگر جدید متنوع برزدن[3]، معکوس‌کننده[4] و جهش[5] تعریف می‌شود تا فضای جستجو را به طور کارامد کاوش کنند و در هر مرحله یکی از آنها به طور تصادفی (نه جانبدارانه) انتخاب و اجرا شوند. در خط ۸ الگوریتم، یک عدد تصادفی بین ۱ تا ۳ ایجاد می‌شود که متناسب با آن یکی از عملگرهای ذکرشده اجرا می‌گردد تا همسایه جدیدی برای جواب جاری تولید کند. به عنوان مثال، اگر کدگذاری راه حل جاری شبیه جدول ۲ باشد. فرض کنید در خط ۸ الگوریتم، عدد ۱ تولید شود. عملگر Shuffle یک زیررشته تصادفی از S مثلاً زیررشته $S[۳..۷]$ را انتخاب می‌کند و سپس در این بازه عددی به شکل تصادفی مثل ۵ انتخاب می‌شود. این زیررشته به دو زیررشته $S[۳..۵]$ و $S[۶..۷]$ تجزیه می‌شود و بعد از آن این دو مقدار در رشته جدید S' با هم جابه‌جا خواهند شد (عمل برزدن انجام می‌شود). یعنی مقدار $S[۶..۷]$ در $S'[۳..۴]$ و مقدار $S[۳..۵]$ در $S'[۵..۷]$ قرار می‌گیرد و بقیه مقادیر متناظر از متغیر S به متغیر S' منتقل می‌شوند. جداول ۳ و ۴ به ترتیب راه حل جاری S و تولید همسایه جدید S' از روی S با اعمال عملگر Shuffle را نشان می‌دهند.

اگر خط ۸ الگوریتم ۱، مقدار ۲ را برگرداند، عملگر Reverse برای تولید همسایگی جدید فراخوانی می‌شود. کارایی این الگوریتم در جداول ۵ و ۶ قابل رؤیت است. این عملگر ابتدا یک زیررشته تصادفی از S، مثلاً $S[۳..۷]$ را انتخاب کرده و سپس آن را معکوس می‌کند. یعنی مقدار $\text{Reverse}(S[۳..۷])$ در رشته $S'[۳..۷]$ قرار می‌گیرد. در نهایت اگر عدد تولیدی در خط ۸ الگوریتم ۱، مقدار ۳ باشد، عملگر Mutate برای تولید همسایگی جدید فراخوانی می‌شود. در این فراخوانی، دو نقطه تصادفی از کروموزوم کدشده انتخاب گردیده و جابه‌جایی انجام می‌گیرد (به دلیل مشابهت این عملگر با عملگر ژنتیک، جهش نام‌گذاری شد). جداول ۷ و ۸، جزئیات این عملگر را به تصویر می‌کشند که دو نقطه تصادفی ژن سوم و ژن هفتم هستند.

یعنی مقدار $S[۷]$ در $S'[۳]$ و مقدار $S[۷]$ در $S'[۳]$ قرار می‌گیرند. بقیه ژن‌ها بدون تغییر باقی می‌مانند. بعد از تعیین همسایگی و یافتن راه حل جدید، در خط ۱۶ الگوریتم این راه حل ارزیابی و در صورت خرابی احتمالی، اصلاح روی آن صورت می‌گیرد. سپس تابع برازش این راه حل جدید در خط ۱۷ الگوریتم ۱، طبق (۱۲) محاسبه می‌شود. اگر مابه‌التفاوت برازش راه حل جدید و قدیم منفی بود (نشان می‌دهد جواب جدید در هزینه کمینه‌تر است) قطعاً راه حل جدید در خط ۲۰ الگوریتم

3. Shuffle
4. Reverse
5. Mutate

جدول ۹: معرفی سناریوها.

شماره سناریو	۱	۲	۳	۴	۵	۶
تعداد ماشین‌های مجازی	۲۰	۴۰	۶۰	۸۰	۱۰۰	۱۲۰
تعداد ماشین‌های فیزیکی	۱۶	۱۶	۳۲	۳۲	۶۴	۱۲۰

پذیرش می‌شود و در غیر این صورت عددی تصادفی از سیستم دریافت می‌شود و راه حل جدید بد، به احتمال خاص پذیرش می‌شود. مثلاً در شروع الگوریتم به دلیل بالابودن درجه حرارت، مقدار کسر $\Delta F/T_K$ عدد فوق‌العاده مثبت کوچکی مثل $-\varepsilon_1$ است و بنابراین احتمال محقق‌شدن شرط $e^{\varepsilon_1} > \varepsilon$ بالاست که به منزله احتمال بالایی برای پذیرش راه حل بد است. در هر درجه حرارت خاص T_k، این عملیات به اندازه *MaxIteration* بار انجام می‌شود تا این جواب مکان پایدار ترمودینامیک خود را در درجه حرارت خاص پیدا کند. سپس کاهش تدریجی درجه حرارت به اندازه ΔT در خط ۲۷ الگوریتم ۱ اعمال می‌شود. بعد از محقق‌شدن شرط پایان، کروموزوم S چیدمان و جایگذاری بهینه ماشین‌های مجازی را با توجه به اهداف مسئله به عنوان خروجی در خط ۳۳ الگوریتم برمی‌گرداند.

۷- ارزیابی کارایی

جهت ارزیابی کارایی الگوریتم پیشنهادی، سناریوهای مختلفی تعریف می‌شوند تا بتوان از طریق خروجی شبیه‌سازی الگوریتم پیشنهادی و مقایسه آن با الگوریتم‌های موجود به نتایج قابل اعتماد دست پیدا کرد. در این راستا، راه حل‌های موجود را می‌توان به سه دسته حریصانه، ابتکاری و فراابتکاری تقسیم‌بندی کرد. به همین منظور از هر دسته یک الگوریتم کارامد جهت مقایسه با الگوریتم پیشنهادی انتخاب می‌شود. از آنجایی که مسئله جایگذاری ماشین‌های مجازی به مسئله بسته‌بندی اشیا چکیده‌سازی می‌شود و روش FFD یک راه حل حریصانه پرکاربرد برای حل این نوع مسایل است، آن را جهت ارزیابی در نظر می‌گیریم [۱۳]. از طرف دیگر، EASY backfilling یک روش ابتکاری برای حل مسایل زمان‌بندی کارها در محیط صف‌بندی پویاست به طوری که با جابه‌جایی زیرمجموعه‌ای از کارها در لیست مرتبی از کارهای درخواست‌شده می‌توان کارایی بهتری به دست آورد، منوط به این که این جابه‌جایی کارها، زمان انتظار و تأخیر زیادی روی کارهای سر صف تحمیل نکند. به همین منظور روش backfilling از منبع شماره ۲۱ اخذ شده و آن را مطابق مسئله بیان‌شده در این مقاله سفارشی می‌کنیم [۲۱]. به عبارت دیگر، در الگوریتم سفارشی‌شده، زیرمجموعه‌ای از درخواست‌ها جهت جابه‌جایی انتخاب می‌شوند که نه تنها باعث تقلیل بهره‌وری منابع و افزایش هدررفت (متناسب با اهداف مسئله) نشوند و بالعکس با رویکرد کاهش هدررفت منابع در این الگوریتم سفارشی در تعداد سرویس‌دهنده‌های فعال صرفه‌جویی صورت می‌گیرد و بالطبع تأثیر مستقیم در کاهش مصرف برق خواهد داشت. از آنجایی که کاربران مختلف به طور بالقوه ماشین‌های مجازی مشابه ولی به تعداد زیاد جهت اجرای پروژه‌های خود درخواست می‌کنند، بنابراین در لیست درخواست‌ها نمونه‌های زیادی از ماشین‌های مجازی درخواستی مشابه وجود دارد که با اعمال الگوریتم ابتکاری backfilling سفارشی‌شده می‌توان کارایی راه حل FFD را به طور بالقوه بهبود داد. به نحوی که در ابتدا لیست درخواستی به صورت نزولی مرتب می‌شود و سپس یک زیرلیست از درخواست‌های مشابه انتخاب و به ابتدای لیست اصلی منتقل می‌گردد که ممکن است به طور بالقوه جواب بهتری حاصل کند. البته این تکنیک می‌تواند به صورت دوره‌ای و روی زیرلیست غیر مشابه نیز اعمال گردد که در صورت مؤثربودن در الگوریتم جدید در نظر گرفته شود. بنابراین روش backfilling به طور بالقوه کیفیت راه حل بهتری از روش حریصانه FFD ارائه می‌دهد که این کار با تحمیل هزینه اجرای بالاتر صورت می‌گیرد. به عنوان مثال، لیست نرمالی از درخواست‌ها به صورت ۰٫۳، ۰٫۳، ۰٫۳، ۰٫۳، ۰٫۳۵، ۰٫۴ را در نظر بگیرید که از چپ به راست به صورت نزولی مرتب شده‌اند. روش حریصانه برای جایگذاری ماشین‌های مجازی درخواستی به تعداد ۳ سرویس‌دهنده فعال نیاز دارد. حال آن که الگوریتم اکتشافی backfilling با انتخاب زیرلیست ۰٫۳ و ۰٫۳ در لیست اصلی ۰٫۳، ۰٫۳، ۰٫۳، ۰٫۳، ۰٫۳۵، ۰٫۴ و انتقال آن به سر لیست، یک صف جدید به صورت ۰٫۳، ۰٫۳، ۰٫۳۵، ۰٫۴، ۰٫۳، ۰٫۳ ایجاد می‌کند که این الگوریتم به تعداد ۲ سرویس‌دهنده فعال برای جایگذاری ماشین‌های مجازی درخواست‌شده نیاز دارد. نهایتاً یک الگوریتم ژنتیک ترکیبی کارامد به عنوان نماینده دسته سوم (الگوریتم‌های فراابتکاری) به منظور مقایسه و ارزیابی در برابر روش پیشنهادی انتخاب می‌شود [۲۶]. به دلایل قابلیت انطباق با فضای جستجوی گسسته مسئله و کارایی بالا در کاوش فضای جستجو، این الگوریتم انتخاب و متناسب با شرایط مسئله، بدون در نظر گرفتن توپولوژی شبکه سفارشی می‌شود. مقایسه خروجی الگوریتم‌ها بر اساس میزان توان کل مصرفی مرکز ابر، میزان هدررفت کل منابع مرکز ابر و تعداد سرویس‌دهنده‌های فعال ابر انجام می‌شود.

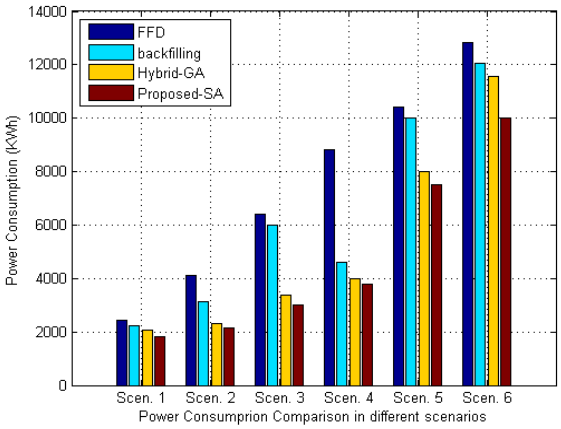
۷-۱ سناریوها، دادگان و تنظیمات

در این بخش، ۶ سناریوی مختلف برای حالات متفاوتی که تعداد ماشین‌های مجازی درخواستی و تعداد سرویس‌دهنده‌های موجود مرکز ابر متغیر هستند تعریف می‌شوند. در این سناریوها افزایش تدریجی مقادیر در نظر گرفته شده تا مقیاس‌پذیری الگوریتم مخصوصاً در شرایطی که با ورودی حجیمی از درخواست‌ها سروکار داریم در محک جدی قرار گیرد. جدول ۹، معرفی سناریوهای شبیه‌سازی را به تصویر می‌کشد. اگرچه مدیریت مصرف انرژی شامل سرویس‌دهنده‌ها و تجهیزات شبکه‌ای است، در این مقاله به طور خاص تمرکز روی شبیه‌سازی به منظور مدیریت مصرف انرژی و هدررفت منابع مراکز داده‌ای و سرویس‌دهنده‌های آن در معماری مقیاس متوسطی مثل معماری پورتلند که حداکثر شامل ۱۲۸ سرویس‌دهنده فیزیکی است در نظر گرفته شده است [۲۵].

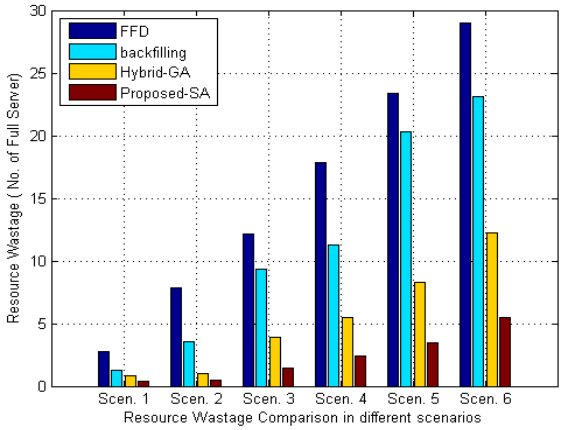
با توجه به جدول، سرویس‌دهنده‌های فیزیکی این شبیه‌سازی، همگن و به ترتیب دارای توان پردازشی و حافظه‌ای ۸۰۰۰ MIPS و ۱۶ GB هستند که در بهره‌وری کامل، ۳۰۰ وات توان مصرفی دارند. در ضمن، توان مصرفی آنها در حالت فعال بی‌کار ۷۰ درصد حالت بهره‌وری کامل در نظر گرفته شده و بنابراین در (۱)، ضریب θ_j برای هر پردازنده jام ۰٫۷ مقداردهی شده است. قدرت پردازشی و حافظه مورد نیاز ماشین‌های مجازی با یک توزیع نرمال به ترتیب از بازه‌های [۵۰۰~۴۰۰۰MIPS] و [۱~۸GB] انتخاب می‌شوند.

۷-۲ شبیه‌سازی و تحلیل داده‌ها

جهت اعتماد به نتایج شبیه‌سازی‌ها، اجرای کلیه سناریوها در شرایط یکسان برای تمام الگوریتم‌های مقایسه‌ای با ۲۰ اجرای مستقل انجام شده و در نهایت میانگین جواب‌های تولیدی آنها گزارش شده‌اند. جهت اجرای الگوریتم پیشنهادی تبرید فلزات، به ترتیب دمای ذوب اولیه 10^4، دمای انجماد ۱۰ و تغییر دما ۲۰۰ درجه سانتی‌گراد در نظر گرفته می‌شود. در ضمن برای رسیدن به حالت تعادل ترمودینامیک در هر دمای ثابت



شکل ۴: مقایسه کارایی الگوریتم‌ها در محاسبه مصرف انرژی برای تمام سناریوها.



شکل ۵: مقایسه کارایی الگوریتم‌ها در محاسبه میزان هدررفت منابع برای تمام سناریوها.

جدول ۱۰: معرفی پارمترهای الگوریتم تبرید فلزات.

پارامتر	مقدار
$T_{.}$	$۱۰^{۴}$
$Freeze$	۱۰
ΔT	۲۰۰
$MaxIteration$	۱۵
n	[۲۰ -۵۰۰]
m	[۲۰ -۳۰۰]

حداکثر ۱۵ تکرار انجام می‌شود که در جدول ۱۰ قابل رؤیت است. اعداد عنوان‌شده با تجربه به دست آمده از روش آزمون و خطا حاصل شده است، به طوری که مؤثرترین مقدار آنها در نتیجه نهایی در نظر گرفته شده‌اند.

همه آزمایش‌ها در شرایط یکسان روی کامپیوتری با پردازنده اینتل Corei۳ ۳۸۰M، با دو هسته فیزیکی، نرخ کلاک ۲٫۵۳ گیگاهرتز و چهار پردازنده منطقی و ۸ گیگابایت حافظه در محیط برنامه‌نویسی Matlab ۲۰۱۸ شبیه‌سازی شده‌اند. شکل ۴، مقایسه مصرف انرژی مرکز داده‌ای حاصل از خروجی الگوریتم‌های مختلف را در سناریوهای مختلف به تصویر می‌کشد.

همان طور که در شکل ۴ مشاهده می‌شود، الگوریتم پیشنهادی بر تمام الگوریتم‌ها در محاسبه مصرف انرژی برتری دارد. الگوریتم FFD و backfilling به دلیل این که توجهی به هدررفت منابع ندارند، نسبت به سایر الگوریتم‌ها کارایی پایین‌تری دارند. تنها الگوریتمی که با الگوریتم پیشنهادی رقابت جدی دارد الگوریتم ژنتیک ترکیبی است. البته در سناریوهای آخر که فضای جستجو خیلی بزرگ است، اختلاف کارایی از لحاظ محاسبه توان مصرفی قابل ملاحظه‌تر است. نکته جالب در سناریوی چهارم این است که دلیل کارایی خیلی پایین الگوریتم FFD استفاده از دادگانی است که ضریب همبستگی بین بردار منابع، عدم توازن بالایی نشان می‌دهد و این دلیل ناکارمدی زیاد این الگوریتم است. همچنین شکل ۵، مقایسه میزان هدررفت منابع محاسبه‌شده توسط الگوریتم‌های مختلف برای همه سناریوها را به تصویر می‌کشد.

همان طور که شکل ۵ نشان می‌دهد الگوریتم پیشنهادی منابع کمتری به هدر می‌دهد و تنها الگوریتمی که قابلیت رقابت دارد، الگوریتم ژنتیک ترکیبی در سناریوهای مربوط به مقیاس نسبتاً کوچک است. هرچه فضای جستجو بزرگ‌تر شود، برتری الگوریتم پیشنهادی محسوس‌تر خواهد بود.

شکل ۶ نیز رفتار مشابهی به مانند شکل‌های ۴ و ۵ از خود نشان می‌دهد. به عبارت دیگر، الگوریتم پیشنهادی تعداد سرویس‌دهنده‌های فعال کمتری را جهت پوشش سرویس دادن به کاربران به خدمت می‌گیرد.

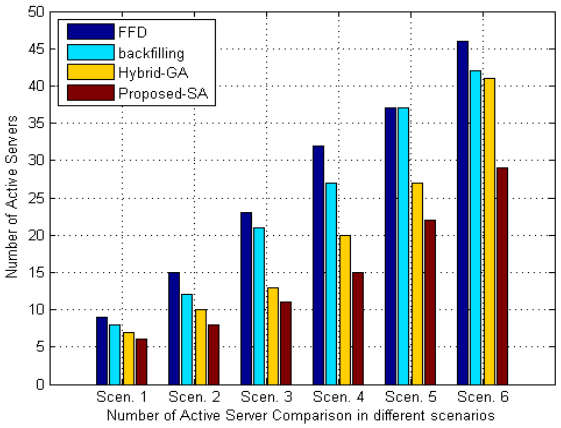
در مجموع با توجه به خروجی شکل‌های ۴ تا ۶، از لحاظ کارایی اهداف در نظر گرفته شده، به ترتیب الگوریتم پیشنهادی بر مبنای تبرید فلزات به همه الگوریتم‌ها برتری دارد و الگوریتم‌های ژنتیک ترکیبی، backfilling و FFD در رده‌های بعدی قرار می‌گیرند. فقط در سناریوی ۴، دو الگوریتم backfilling و FFD از لحاظ تعداد سرویس‌دهنده‌های فعال تقریباً رفتار مشابهی نشان می‌دهند، اما رجوع به مقایسه در هدررفت منابع، نشان می‌دهد که الگوریتم backfilling کاهش بهتری در هدررفت منابع دارد. همچنین جدول ۱۱، جزئیات شبیه‌سازی را به تفکیک سناریوها، الگوریتم‌ها و اهداف تعریف‌شده در مسئله به تصویر می‌کشد.

به منظور تحلیل دقیق عملکرد الگوریتم پیشنهادی در برابر سایر الگوریتم‌ها از لحاظ کمترین، بیشترین مقدار هر تابع هدف و انحراف معیار استاندارد آنها، جدول ۱۲ اختصاص داده می‌شود. از آنجایی که الگوریتم FFD برخلاف سایر الگوریتم‌های تصادفی به صورت قطعی عمل می‌کند، از درج مقدار ثابت آن در جدول ۱۲ صرف نظر شده است.

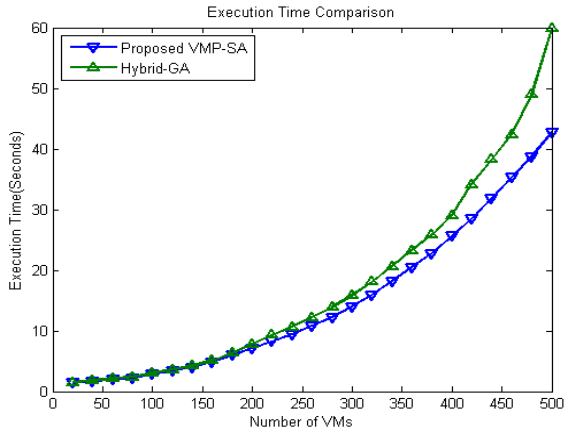
جدول ۱۲، برتری راه حل پیشنهادی را نسبت به سایر روش‌ها از لحاظ توابع هدف مسئله نشان می‌دهد. در ضمن، میزان انحراف معیار استاندارد الگوریتم پیشنهادی در همه سناریوها، عدم پراکندگی پاسخ‌ها و همگرایی بالای آن را نسبت به سایر روش‌ها اثبات می‌کند.

۳-۷ پیچیدگی زمانی الگوریتم

به منظور محاسبه پیچیدگی زمانی الگوریتم پیشنهادی، ابتدا پیچیدگی زمانی عملگرهای فرعی محاسبه می‌شوند و سپس به زمان اجرای الگوریتم اصلی پرداخته می‌گردد. در الگوریتم ۱، روال Check&Correct به دلیل پیمایش حداکثر به تعداد سرویس‌دهنده‌ها در لیست پیوندی، زمان $O(m)$ را صرف می‌کند و بدیهی است که هر یک از عملگرهای همسایگی زمان $O(۱)$ را صرف می‌کنند. از آنجایی که الگوریتم اصلی با درجه حرارت بسیار بالای $T_{.}$ شروع می‌شود و در هر مرحله به میزان ΔT از آن کاسته می‌شود تا در درجه حرارت پایین $Freeze$ به پایان برسد، بنابراین دستور اصلی الگوریتم ۱ که بین خطوط ۵ و ۳۲ قرار دارد به اندازه $K=(T_{.}-Freeze)/\Delta T$ مرتبه تکرار می‌شود. در ضمن، به منظور ایجاد توازن بین شناسایی و بهره‌برداری فضای جستجو، در هر درجه حرارت ثابت، به میزان حداکثر $O(MaxIteration)$ تکرار اجرا انجام می‌شود تا حالت پایدار ترمودینامیک در آن دما حاصل گردد. بنابراین پیچیدگی زمانی کل الگوریتم برابر $O(K.(m+MaxIteration))$ خواهد بود که نسبتاً پیچیدگی زمانی مناسبی است.



شکل ۶: مقایسه تعداد سرویس‌دهنده‌های فعال به عنوان خروجی الگوریتم‌ها برای تمام سناریوها.



شکل ۷: مقایسه زمان اجرای دو الگوریتم تبرید فلزات پیشنهادی در برابر الگوریتم ژنتیک ترکیبی به ازای ورودی‌های مختلف.

جدول ۱۱: مقایسه الگوریتم‌ها در پارامترهای مختلف.

شماره سناریو	تعداد ماشین مجازی	تعداد ماشین فیزیکی	الگوریتم	مصرف انرژی (کیلووات ساعت)	هدررفت منابع	تعداد سرویس‌دهنده‌های فعال
۱	۲۰	۱۶	FFD	۲٫۴۳۵۵	۲٫۷۳۶۲	۹
			Backfilling	۲٫۲۳۵۵	۱٫۲۴۸۹	۸
			HybridGA	۲٫۰۵۷۸	۰٫۸۶۰۱	۷
			VMP-SA	۱٫۸۳۵۵	۰٫۴۱۰۰	۶
۲	۴۰	۱۶	FFD	۴٫۱۳۱۶	۷٫۸۱۷۵	۱۵
			Backfilling	۳٫۱۳۱۶	۳٫۵۴۸۶	۱۲
			HybridGA	۲٫۳۳۱۶	۱٫۰۴۵۷	۱۰
			VMP-SA	۲٫۳۳۱۶	۰٫۴۸۷۴	۸
۳	۶۰	۳۲	FFD	۶٫۳۹۳۲	۱۲٫۱۲۳	۲۳
			Backfilling	۵٫۹۹۳۲	۹٫۳۳۵۷	۲۱
			HybridGA	۳٫۳۹۳۲	۳٫۸۷۸۸	۱۳
			VMP-SA	۲٫۹۹۳۲	۱٫۴۲۷۲	۱۱
۴	۸۰	۳۲	FFD	۸٫۸۰۴۶	۱۷٫۸۱۷	۳۲
			Backfilling	۴٫۶۰۴۶	۱۱٫۲۵۰	۲۷
			HybridGA	۴٫۰۰۴۶	۵٫۴۶۸۴	۲۰
			VMP-SA	۳٫۸۰۴۶	۲٫۳۸۹۵	۱۵
۵	۱۰۰	۶۴	FFD	۱۰٫۴۲۷۱	۲۳٫۳۳۹	۳۷
			Backfilling	۱۰٫۰۲۷	۲۰٫۳۱۱	۳۷
			HybridGA	۸٫۰۱۱۷	۸٫۲۹۶۲	۲۷
			VMP-SA	۷٫۵۰۴۶	۳٫۴۷۲۶	۲۲
۶	۱۲۰	۱۲۰	FFD	۱۲٫۸۴۲	۲۸٫۹۹۸	۴۶
			Backfilling	۱۲٫۰۴۲	۲۳٫۱۴۳	۴۲
			HybridGA	۱۱٫۵۴۶	۱۲٫۲۰۴	۴۱
			VMP-SA	۱۰٫۰۲۷	۵٫۵۱۱۷	۲۹

۷-٤ مقیاس‌پذیری الگوریتم

جهت بررسی و مقایسه الگوریتم‌ها از لحاظ مقیاس‌پذیری در زمان اجرا، میانگین زمان سپری‌شده اجرای هر یک از الگوریتم‌های فراابتکاری به ازای هر یک از ورودی‌های مختلف ثبت شده‌اند. شکل ۷، مقایسه زمان اجرای دو الگوریتم فراابتکاری رقیب یعنی الگوریتم تبرید فلزات پیشنهادی و الگوریتم ژنتیک ترکیبی را نشان می‌دهد. از آنجایی که زمان اجرای ظاهراً پایین دو الگوریتم حریص FFD و ابتکاری backfilling در برابر کیفیت خروجی اهداف مسئله، معنادار نیستند از نمایش آن در شکل ۷ خودداری می‌شود.

به منظور ارزیابی مقیاس‌پذیری الگوریتم پیشنهادی، تعداد ماشین‌های مجازی درخواستی و تعداد سرویس‌دهنده‌های فیزیکی موجود به ترتیب از ۲۰ به ۵۰۰ و از ۲۰ به ۳۰۰ افزایش داده می‌شوند. همان طور که شکل ۷ نشان می‌دهد، اگرچه زمان اجرای هر دو الگوریتم پیشنهادی و الگوریتم ژنتیک ترکیبی زیر ۱ دقیقه هستند ولی الگوریتم پیشنهادی سریع‌تر و با توجه به اهداف مسئله کارامدتر عمل می‌کند. در ضمن به ازای افزایش تدریجی ورودی‌ها، رفتار تقریباً نمایی از خود نشان می‌دهد در حالی که الگوریتم ژنتیک ترکیبی به دلیل استفاده از عملگرهای پیچیده و پرمحاسبه، رفتاری کاملاً نمایی از خود نشان می‌دهد.

جدول ۱۲: تحلیل و مقایسه آماری الگوریتم‌ها از لحاظ توابع هدف.

شماره سناریو	تعداد ماشین مجازی	تعداد ماشین فیزیکی	الگوریتم	مصرف انرژی (کیلووات ساعت)		هدررفت منابع	
				حداکثر– حداقل	انحراف معیار استاندارد	حداکثر– حداقل	انحراف معیار استاندارد
۱	۲۰	۱۶	Backfilling	۲٫۱۰–۲٫۴۳	۰٫۲۸	۱٫۱۶–۲٫۹۸	۰٫۳۳
			HybridGA	۲٫۰۱–۲٫۲۳	۰٫۱۶	۰٫۸۵–۰٫۹۴	۰٫۱۴
			VMP-SA	۰٫۱۰–۱٫۹۳	۰٫۱۰	۰٫۴۰–۰٫۵۰	۰٫۱۱
۲	۴۰	۱۶	Backfilling	۲٫۹۰–۴٫۱۳	۰٫۲۴	۳٫۱۸–۶٫۸۱	۰٫۲۷
			HybridGA	۲٫۲۰–۲٫۵۰	۰٫۲۰	۱٫۰۰–۱٫۲۸	۰٫۱۸
			VMP-SA	۲٫۰۰–۲٫۲۲	۰٫۱۱	۰٫۴۶–۰٫۵۰	۰٫۰۵
۳	۶۰	۳۲	Backfilling	۵٫۸۵–۶٫۶۷	۰٫۲۶	۸٫۵۰–۹٫۹۱	۰٫۱۷
			HybridGA	۳٫۱۰–۴٫۴۵	۰٫۱۷	۳٫۸۰–۴٫۱۰	۰٫۱۵
			VMP-SA	۲٫۹۵–۳٫۱۱	۰٫۱۰	۱٫۴۰–۱٫۵۳	۰٫۰۹
۴	۸۰	۳۲	Backfilling	۴٫۵۰–۴٫۸۲	۰٫۳۲	۱۰٫۲–۱۴٫۲۳	۰٫۲۸
			HybridGA	۴٫۰۰–۴٫۲۵	۰٫۱۶	۵٫۱۰–۶٫۴۸	۰٫۲۱
			VMP-SA	۳٫۷۵–۴٫۰۰	۰٫۱۲	۲٫۲۵–۲٫۶۰	۰٫۱۳
۵	۱۰۰	۶۴	Backfilling	۹٫۵۰–۱۳٫۶۸	۰٫۳۸	۱۸٫۸۵–۱۹٫۲	۰٫۴۳
			HybridGA	۷٫۵۰–۹٫۲۵	۰٫۲۶	۸٫۱۰–۱۰٫۶	۰٫۲۳
			VMP-SA	۷٫۲۰–۷٫۸۰	۰٫۱۰	۳٫۱۰–۳٫۹۵	۰٫۱۵
۶	۱۲۰	۱۲۰	Backfilling	۱۱٫۰۳–۱۲٫۸۴	۰٫۳۵	۲۳٫۰–۲۶٫۵۲	۰٫۴۵
			HybridGA	۱۰٫۹۰–۱۱٫۶۰	۰٫۲۸	۱۱٫۸۵–۱۲٫۶	۰٫۲۲
			VMP-SA	۱۰٫۰۰–۱۰٫۲۴	۰٫۱۰	۵٫۴۰–۵٫۶۳	۰٫۱۸

۸- نتیجه‌گیری و راهبرد آینده

در این مقاله مسئله جایگذاری ماشین‌های مجازی به یک مسئله بهینه‌سازی چندهدفه با رویکرد کمینه‌سازی مصرف انرژی و هدررفت منابع فرمول‌بندی شد. در ضمن، مسئله بهینه‌سازی چندهدفه، بعد از نرمال‌سازی و اعمال ضرایب اهمیت توابع هدف به یک مسئله بهینه‌سازی تک‌هدفه تبدیل شد. برای حل این مسئله پیچیده، یک الگوریتم مبتنی بر تبرید فلزات به همراه چند عملگر همسایگی جدید متناسب با فضای جستجوی گسسته ارائه شد. میانگین نتایج اجرای شبیه‌سازی‌ها در سناریوهای مختلف، برتری کامل الگوریتم پیشنهادی را از لحاظ کل مصرف انرژی، هدررفت منابع و تعداد سرویس‌دهنده‌های فعال در برابر الگوریتم‌های حریص، ابتکاری و فراابتکاری موجود در این حوزه نشان می‌دهد. در آینده یک مدل قابلیت اطمینان جهت محاسبه میزان قابلیت اطمینان مرکز ابر با توجه به کسب و کار برنامه کاربردی استفاده‌کننده ارائه می‌شود. در این راستا، یک روش جایگذاری ماشین‌های مجازی با رویکرد افزایش قابلیت اطمینان متناسب با کیفیت خدمات مورد نیاز برای کاربردهای حساس در آینده ارائه خواهد شد.

مراجع

[1] C. Wei, Z. H. Hu, and Y. G. Wang, "Exact algorithms for energy-efficient virtual machine placement in data centers," *Future Generation Computer Systems*, vol. 106, pp. 77-91, 2020.

[۲] س. اصغری و ن. جعفری نویمی‌پور، "یک روش آگاه از هزینه برای ترکیب خدمات ابری به کمک یک الگوریتم ترکیبی،" *مجله علمی رایانش نرم و فناوری اطلاعات*، جلد ۸، شماره ۲، صص. ۲۶-۱۷، تابستان ۱۳۹۸.

[3] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges," *J. of King Saud University-Computer and Information Sciences*, vol. 32, no. 3, pp. 267-286, Mar. 2020.

[4] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster Computing*, vol. 16, pp. 65-75, 2013.

[5] R. Brown, *et al.*, *Report to Congress on Server and Data Center Energy Efficiency: Public Law*, pp. 109-431, Lawrence Berkeley National Laboratory, Berkeley, 2008.

[6] S. U. Khan and A. Y. Zomaya, *Handbook on Datacenters*, Springer, New York, NY, 2015.

[7] M. Hosseini Shirvani, "To move or not to move: an iterative four-phase cloud adoption decision model for IT outsourcing based on TCO," *J. of Soft Computing and Information Technology*, vol. 9, no. 1, pp. 7-17, Spring 2020.

[8] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, "An iterative mathematical decision model for cloud migration: a cost and security risk approach," *Software: Practice and Experience*, vol. 48, no. 3, pp. 449-485, Mar. 2018.

[9] M. A. Reddy and R. Ravindranath, "Virtual machine placement using JAYA optimization algorithm," *Applied Artificial Intelligence*, vol. 34, no. 1, pp. 31-46, 2019.

[10] W. Van Heddeghem, *et al.*, "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64-76, 1 Sept. 2014.

[11] M. Mills, The Cloud Begins with Coal: An Overview of the Electricity Used by the Global Digital Ecosystem, Technical Report, Digital Power Group, Washington D.C, USA, 2013.

[12] V. D. Reddy, B. Setz, G. S. V. R. K. Rao, G. R. Gangadharan, and M. Aiello, "Best practices for sustainable datacenter," *IT Professional,* vol. 20, no. 5, pp. 57-67, Sept./Oct. 2018.

[13] B. S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," *J. of Algorithms*, vol. 6, no. 1, pp. 49-70, Mar. 1985.

[14] M. Yue, "A simple proof of the inequality FFD (L) ≤ 11/9 OPT (L) + 1, ∀ L for the FFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, no. 4, pp. 321-331, 1991.

[15] P. Saeedi and M. Hosseini Shirvani, "An improved thermodynamic simulated annealing-based approach for resource-skewness-aware and power-efficient virtual machine consolidation in cloud datacenters," *Soft Comput.*, vol. 25, pp. 5233-5260, 2021.

[16] P. Saeedi, "An energy-efficient genetic-based algorithm for virtual machine placement in cloud datacenter," *J. of Multidisciplinary Engineering Science and Studies*, vol. 5, no. 5, pp. 1-4, May 2019.

[17] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing," *J. of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 97-112, 2016.

[18] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud

computing," *J. of Computer and System Sciences*, vol. 79, no. 8, pp. 1230-1242, Dec. 2013.

[19] M. Y. Kao, (Ed.), *Encyclopedia of Algorithms*, Springer Science & Business Media, 2008. ISBN: 978-0-387-30162-4.

[20] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase, "Virtual machine hosting for networked clusters: building the foundations for autonomic orchestration," in *Proc. of First Int. Workshop on Virtualization Technology in Distributed Computing,* pp. 7-7, Tampa, FL, USA, 17-17 Nov. 2006.

[21] A. A. Chandio, N. Tziritas, M. S. Chandio, and C. Z. Xu, "Energy efficient VM scheduling strategies for HPC workloads in cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 24, Article No.: 100352, Dec.2019.

[22] L. Gao and G. N. Rouskas, "A spectral clustering approach to network-aware virtual request partitioning," *Computer Networks,* vol. 139, pp. 70-80, 5 Jul. 2018.

[23] Y. Wu, M. Tang, and M. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, SMC'12,* pp. 1245-1250, Seoul, South Korea, 14-17 Oct. 2012.

[24] N. Su, A. Shi, C. Chen, E. Chen, and Y. Wang, "Research on virtual machine placement in the cloud based on improved simulated annealing algorithm," *World Automation Congress, WAC'16,* 7 pp., Rio Grande, PR, USA, 31 Jul.-4 Aug. 2016.

[25] S. Farzai, M. Hosseini Shirvani, and M. Rabbani, "Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters," *Sustainable Computing: Informatics and Systems*, vol. 28, Article No.: 100374, Dec. 2020.

[26] M. Tang and S. A. Pan, "Hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," *Neural Process Lett*, vol. 41, no. 2, pp. 211-221, Apr. 2015.

[27] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 13 May 1983.

[28] M. Hosseini Shirvani, "Web service composition in multi-cloud environment: a bi-objective genetic optimization algorithm," in *Proc. Innovations in Intelligent Systems and Applications, INISTA'18*, 6 pp., Thessaloniki, Greece, 3-5 Jul. 2018.

[29] M. Hosseini Shirvani and A. Babazadeh Gorji, "Optimisation of automatic web services composition using genetic algorithm," *Int. J. Cloud Computing*, vol. 9, no. 4, pp. 397-411, 2020.

[۳۰] ع. محمدزاده، م. مصدری، ف. سلیمانیان قره چپق و ا. جعفریان، "ارائه یک الگوریتم بهبودیافته بهینه‌سازی گرگ‌های خاکستری برای زمان‌بندی جریان کار در محیط محاسبات ابری،" *مجله علمی رایانش نرم و فناوری اطلاعات*، جلد ۸، شماره ۴، صص. ۲۹-۱۷، زمستان ۱۳۹۸.

میرسعید حسینی شیروانی تحصیلات خود را در مقاطع کارشناسی (مهندسی کامپیوتر گرایش نرم‌افزار)، کارشناسی ارشد (مهندسی کامپیوتر گرایش نرم‌افزار) و دکتری (مهندسی کامپیوتر با گرایش سیستم‌های نرم‌افزاری) به‌ترتیب در سال‌های ۱۳۷۹، ۱۳۸۳ و ۱۳۹۶ از دانشگاه‌های تهران به پایان رسانده است و هم‌اکنون استادیار دانشکده مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد ساری می‌باشد. نام‌برده به عنوان عضو هیات علمی تمام وقت در دانشگاه آزاد واحد ساری مشغول به فعالیت های آموزشی و پژوهشی می باشد. در ضمن، ایشان دارای سابقه‌ی تدریس در سایر موسسات دولتی و آزاد نظیر دانشگاه صنعتی نوشیروانی بابل، دانشگاه مازندران بابلسر، دانشگاه‌های آزاد اسلامی واحدهای بابل، آمل و قائمشهر است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: رایانش ابری، رایانش مه، اینترنت اشیاء، شبکه‌های حسگر بی‌سیم، سیستم های توزیع شده، الگوریتم‌های موازی، محاسبات نرم هوش مصنوعی و یادگیری ماشین.