Tabriz Journal of Electrical Engineering (TJEE), vol. 55, no. 2, 2025 Serial no. 112
DOI: 10.22034/tjee.2024.59034.4755

Virtual Machine Placement in Cloud Data Centers using Energy Valley Optimizer Algorithm

Mohsen Kiani

Department of Computer Science, Khansar Campus, University of Isfahan, Isfahan, Iran.
E-mails: m.kiani@khc.ui.ac.ir

Abstract
In the present study, Energy Valley Optimizer (EVO), as an emerging meta-heuristic optimization algorithm, is employed to solve the problem of virtual to physical machine placement (VMP) in a cloud data center. Further, in order to accommodate the algorithm for the placement problem, some necessary changes were applied to the algorithm. EVO algorithm equipped with a set of properly-designed operators was then employed to solve the VMP problem. Minimizing the power consumption was considered as the optimization goal. The VMP problem was formulated as a constraint optimzation problem. In the evaluation phase, a data center with a given set of heterogeneous physical machines with different input workloads generated synthetically were modeled to evaluate the effectiveness of the EVO algorithm. The results obtained by EVO were compared with several heuristics. The evaluation results indicate that EVO algorithm is able to reduce power consumption from around 3 to around19 percent compared with FFD. Furhter, EVO outperformed all the evaluated heuristics in terms of power consumption. As a secondary parameter, resource wastage in the data center was also evaluated, which the obtained results show effectiveness of EVO when compared to other heuristics.

Keywords
Cloud datacenter, virtual machine placement, power consumption.

Introduction

As cloud data centers continue to grow in size, power consumption has became a primary concernt. At the infrastructure level, the placement of virtual machines (VMs) to physical machines, as a NP-hard problem, is approached by many researchers attempting to cosolidate VMs on PMs thus reducing data centers' power consumption. Various fast heuristic and slow yet efficient meta-heuristics were proposed in this regard. Based on the no free lunch theoreme, no meta-heuristic algorithm is able to outperform others in all optimization problem. This motivates the employment of newly emerged algorithms to solve VM to PM mapping (VMP) problem.

Proposed Work and Methodology

Energy vally optimizer is such a new algorithm that is employed in the present research to slove VMP promlem. The algorithm was slightly altered and proper operators were designed for it to becom applicable to VMP. A heterogeneous data center was modeled and a synthetic task generator was employed to generate VMs with a given correlation between its CPU and RAM demands. The EVO algorithm was used to optimize power consumption of the data center. Several heuristics were also evluated. The results show that EVO outperform FFD in terms of power consumption and resource wastage by a significant margin. Figure 1(a) shows a comparison of EVO with other algorithms (Genetic Algorithm (GA), Chemical Reaction Optimization (CRO), Best Fit (BF), Worst Fit (WF), and Random Fit (RF)) in terms of power comsumption reduction (in percent) with respect to FFD algorithm as the baseline. The results are given for VMs changing from 100 to 1500. As can be seen, EVO outperforms all other algorithms. Figure 1(b) shown the obtained resource wastage of the algorithms for VMs generated with correlation values (ρ) between CPU and RAM resources for 200 VMs. As it can be seen, while the performane of EVO is similar to GA and CRO, it performs well even comparing with BF which focuses on reducing resource wastage. One main benefit of EVO is its simple parameter setting. Further, a limitation of EVO is however its long running times that reaches several hundred seconds for large numbers of input VMs.

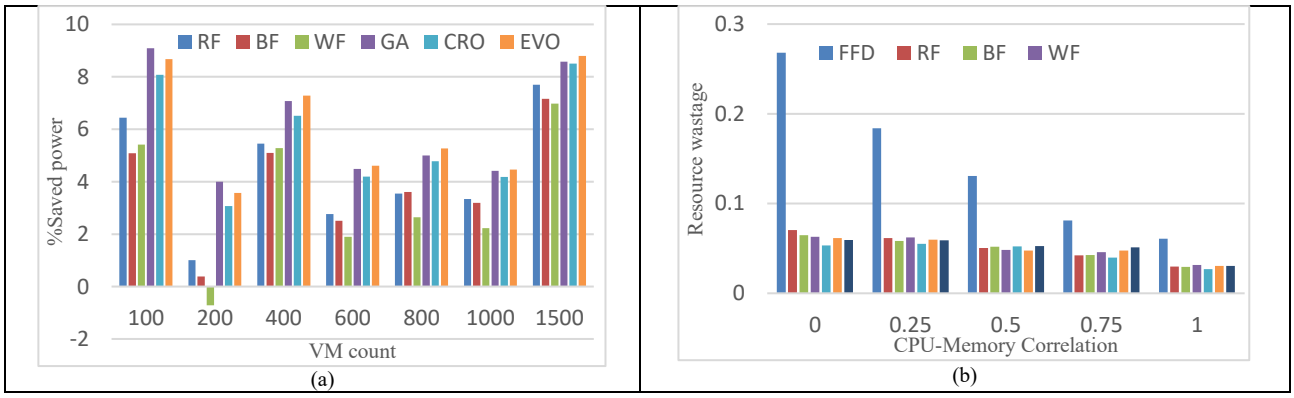


Figure 1: Power consumption reduction of different algorithms with respect to FFD

Conclusion

In this study, motivated by the well-known no free lunch theoreme, energy vally optimizer as a newly-introduced meta-heuristic algorithm was modified to be applicable to the promblem of mapping VMs to PMs. To do so, several appropriate oprators were designed and used by the algorithm. Then, the algorithm was evaluated for a heterogeneous cloud data center and compared with several heuristics. The obtained results indicate that EVO is efficient in solving the VMP problem. In the future, EVO should be compared with other similar meta-heuristic methods. Futher, since the simulated system in this study was static, in th future EVO should be employed in a dynamic simulation to consider other parameters such as service-level agreement violation.

جانمایی ماشین‌های مجازی در مرکز داده ابری با استفاده از الگوریتم بهینه‌سازی دره انرژی

محسن کیانی
استادیار، پردیس خوانسار، دانشگاه اصفهان، اصفهان، ایران

چکیده
در پژوهش حاضر الگوریتم بهینه‌سازی دره انرژی به عنوان یک الگوریتم نوظهور برای حل مسئله جانمایی ماشین‌های مجازی روی ماشین‌های فیزیکی در یک مرکزداده ابری اعمال شده است. همچنین، برای اعمال مناسب الگوریتم برای حل مسئله جانمایی، تغییراتی به الگوریتم پایه اعمال شد. الگوریتم بهینه‌سازی دره انرژی با معرفی چندین عملگر مناسب مسئله جانمایی ماشین‌های مجازی اصلاح شد. کاهش توان مصرفی مرکز داده به عنوان هدف بهینه‌سازی لحاظ شد. مسئله جانمایی به صورت یک مسئله بهینه‌سازی مقید فرموله شد. در مرحله ارزیابی، الگوریتم برای یک مرکز داده شامل ماشین‌های فیزیکی ناهمگن و به ازای بارهای کاری مختلف ارزیابی شد تا کارآمدی الگوریتم ارزیابی شود. نتایج به‌دست آمده توسط الگوریتم با چندین روش ابتکاری پایه و دو الگوریتم فراابتکاری مورد مقایسه قرار گرفت. نتایج شبیه‌سازی نشان می‌دهد الگوریتم بهینه‌سازی دره انرژی نسبت به الگوریتم پایه بین حدود ۳ تا ۱۹ درصد باعث کاهش توان مصرفی در مرکز داده شده است. همچنین، الگوریتم بهینه‌سازی دره انرژی در همه ارزیابی‌های صورت گرفته از نظر توان مصرفی از روش‌های ابتکاری بهتر و نسبت به دو روش فراابتکاری اندکی بهتر عمل کرده است. به‌عنوان یک پارامتر جانبی دیگر، هدررفت منابع نیز مورد ارزیابی قرار گرفت که نتایج نشانگر کارامدی الگوریتم نسبت به روش‌های ابتکاری است.

کلمات کلیدی
مرکز داده ابری، ماشین‌های مجازی، جانمایی، توان مصرفی.

نام نویسنده مسئول: محسن کیانی
ایمیل نویسنده مسئول: m.kiani@khc.ui.ac.ir

تاریخ ارسال مقاله: ۱۴۰۲/۰۸/۰۸
تاریخ(های) اصلاح مقاله: ۱۴۰۲/۱۱/۱۴
تاریخ پذیرش مقاله: ۱۴۰۳/۰۱/۱۹

۱- مقدمه

امروزه بسیاری از شرکت‌ها به جای خرید و نگهداری تجهیزات رایانه‌ای برای رفع نیازهایشان، از سامانه‌های ابری استفاده می‌کنند. یک سامانه ابری متشکل از مراکز داده است که در سه سطح نرم‌افزار به عنوان سرویس، سکو به عنوان سرویس و زیرساخت به عنوان سرویس، سرویس‌دهی می‌کنند. در سرویس‌دهی در سطح زیرساخت، یک مرکز داده ابری، برای رفع نیاز کاربران تعدادی ماشین مجازی (VM) ایجاد و در اختیار آن‌ها قرار می‌دهد. ماشین‌های مجازی برای اجرا بر روی ماشین‌های فیزیکی (PM) یا همان سرورها جانمایی می‌شوند. توان و انرژی مصرفی در مرکز داده، که از یک سو هزینه آن باید پرداخت شود و از سوی دیگر به دلیل تولید حرارت باید خنک‌سازی مناسب در مرکز داده انجام شود، یک پارامتر مورد توجه در مدیریت مرکز داده است. به همین دلیل، تحقیقات فراوانی برای مدیریت توان و انرژی در مراکز داده ابری انجام شده است. اکثر تحقیقات مبتنی بر جانمایی یا نگاشت مناسب VMها بر روی PMها است به طوری که تعداد PM فعال کمینه باشد تا با خاموش کردن PMهای غیرفعال، مصرف توان و انرژی را کاهش داد.

جانمایی VMها روی PMها یک مسئله ان‌پس سخت است [۱, ۲]. بنابراین این مسئله با روش‌های دقیق قابل حل نیست. روش‌های ابتکاری و فراابتکاری فراوانی برای حل این مسئله توسعه یافته است [۳-۵] که نتایج تحقیقات حاکی از این‌است که روش‌های فراابتکاری قادر هستند پاسخ‌های بهتری برای مسئله جانمایی تولید کنند، هرچند زمان اجرای آن‌ها بیشتر است. با توجه به رشد سریع مراکز داده و افزایش انرژی و توان مصرفی، توان مصرفی یکی از مهمترین پارامترهای مدنظر برای بهینه‌سازی در مرکز داده است [۶]. هرساله تعداد قابل توجهی روش ابتکاری برای استفاده در مسائل بهینه‌سازی توسعه می‌یابد [۷]. با توجه به نظریه هیچ ناهاری مجانی نیست (NFL) [۸]نمی‌توان هیچ الگوریتمی را برتر از سایر الگوریتم‌ها دانست بلکه اگر از دو الگوریتم برای حل همه مسائل بهره برد، به طور میانگین برابر هستند. بنابراین هر الگوریتم‌های فراابتکاری در دسته‌ای از کاربردها و مسائل می‌تواند کارامدتر از سایرین باشد. این نظریه باعث ایجاد انگیزه جهت استفاده از الگوریتم‌های مختلف برای حل مسائل گوناگون می‌شود تا کارامدی آن الگوریتم برای مسئله مورد نظر مشخص شود.

با توجه به اهمیت کاهش مصرف توان در مراکز داده، در مطالعه حاضر یک الگوریتم فراابتکاری جدید به نام روش بهینه‌ساز دره انرژی[۱] یا به اختصار EVO [۹] که در سال ۲۰۲۳ توسعه یافته است برای حل مسئله جانمایی VMها روی PMها انطباق داده می‌شود. تحقیق حاظر اولین تحقیقی است که الگوریتم EVO را برای این کاربرد بکار می‌گیرد.

الگوریتم‌های فراابتکاری فراوانی برای حل مسئله جانمایی ماشین‌های مجازی به کار گرفته شده است [۲, ۱۰]. از جمله این الگوریتم‌ها می‌توان به گونه‌های مختلف الگوریتم ژنتیک [۱۱, ۱۲]، الگوریتم بهینه‌سازی اجتماع ذرات [۱۳]، الگوریتم کلونی مورچگان [۱۴]، الگوریتم واکنش شیمیایی [۱۵] و الگوریتم

[۱] Energy Valley Optimer

کرم شب‌تاب [۱۶] اشاره کرد. به طور کلی دو رهیافت برای حل مسئله وجود دارد که شامل مدل ایستا و مدل پویا است [۱۷]. در دسته اول، جانمایی بدون در نظر گرفتن میزان استفاده از VMها و با مدل مسئله به صورت مسئله بسته‌بندی مدل می‌شود حال آنکه در دسته دوم، سیستم مدل شده واقع‌گرایانه‌تر بوده و میزان استفاده از منابع در VMها در نظر گرفته می‌شود. در پژوهش حاضر، مشابه کار انجام شده در [۱۸]، سیستم به صورت ایستا مدل شده است تا کارامدی یک الگوریتم نوظهور برای حل مسئله بررسی شود.

۲- کارهای پیشین

در سال‌های اخیر تحقیقات بسیاری برای حل مسئله جانمایی ماشین مجازی در بستر ابر انجام شده است. از نظر سیستم مدل‌شده، دو دسته روش ایستا و پویا وجود دارد. در دسته ایستا، ماشین‌های مجازی دارای اندازه و تعداد ثابت هستند و روش ماشین‌های فیزیکی نگاشت می‌یابند. هدف کاهش تعداد ماشین فیزیکی، کاهش توان و انرژی مصرفی، بهبود میزان استفاده و کاهش هدررفت منابع، و متعادل‌سازی بار است. در روش‌های پویا، میزان استفاده از منابع ماشین‌های مجازی نیز لحاظ شده و پارامترهای کنترل کیفی از قبیل نقض توافقنامه سطح خدمات نیز مورد بررسی قرار می‌گیرد. در این پژوهش، سیستم به صورت ایستا مدل شده است تا بتوان عملکرد الگوریتم EVO در حل مسئله جانمایی را به خوبی بررسی کرد. دو دسته الگوریتم برای حل مسئله جانمایی پیشنهاد شده است. دسته اول، روش‌های ابتکاری هستند که دارای سرعت بالایی بوده اما کیفیت پاسخ آنها چندان مناسب نیست. از جمله این روش‌ها، می‌توان به FFD، BF، WF و RF اشاره کرد [۶]. دسته دوم شامل روش‌های فراابتکاری هستند [۲-۴, ۷, ۱۴]. از جمله روش‌های فراابتکاری می‌توان به الگوریتم ژنتیک [۱۱, ۱۲]، اجتماع مورچگان [۱۹]، و رقابت استعماری [۲۰] و پرش ترکیبی قورباغه [۲۱] اشاره کرد [۲]. روش بهینه‌سازی واکنش شیمیایی[۲] نیز یک نمونه جدید است که با موفقیت در حل مسئله جانمایی مورد استفاده قرار گرفته است [۱۵, ۲۲]. در روش‌های فراابتکاری، در گام اول لازم است الگوریتم را برای حل مسئله جانمایی انطباق داد که مهمترین جنبه آن، توسعه عملگرهای مناسب برای اجرای مراحل مختلف الگوریتم است. در این پژوهش نیز الگوریتم EVO به عنوان یک الگوریتم جدید مد نظر قرار گرفته و عملگرهای مناسب برای مراحل مختلف الگوریتم طراحی شده است. علاوه بر الگوریتم‌های ابتکاری، دو الگوریتم فراابتکاری ژنتیک [۱۱, ۱۲] و CRO [۱۵, ۲۲] برای بررسی و مقایسه عملکرد EVO استفاده شده‌اند.

۳- مفاهیم پایه

در این بخش مقدماتی در مورد مسئله جانمایی و الگوریتم بهینه‌سازی دره انرژی ارائه شده است.

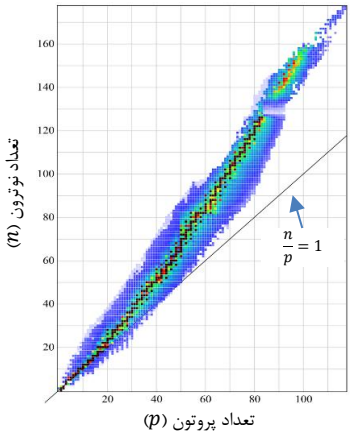
۳-۱- جانمایی ماشین‌های مجازی

در این بخش مسئله جانمایی به صورت مختصر معرفی و فرمول‌بندی می‌شود. فرض کنید تعداد VM ها برای جانمایی برابر m و تعداد PMها برابر n باشد. مجموعه VMها به‌صورت $VM = \{x \in i | 0 \le i \le m-1\}$ بیان می‌شود که هر عضو مجموعه اندیس یک ماشین مجازی است و m تعداد ماشین‌های مجازی را نشان می‌دهد. هر ماشین مجازی دارای میزانی از درخواست پردازش (CPU) و حافظه (RAM) است. همچنین فرض کنید مجموعه ماشین‌های فیزیکی به‌صورت $PM = \{x \in j | 0 \le j \le n-1\}$ باشد که هر عضو مجموعه اندیس یک ماشین مجازی است که دارای ظرفیت مشخصی از قدرت پردازش و فضای حافظه است. فرض شده است که کوچکترین ماشین فیزیکی ظرفیت پذیرش بزرگترین ماشین مجازی را داشته باشد. نگاشت VMها به PMها را می‌توان به صورت یک ماتریس m در n نمایش داد که در آن هر سطر نشان می‌دهد VM روی کدام PM نگاشت یافته است. هر درایه ماتریس، $x_{i,j}$ می‌تواند صفر یا یک باشد. اگر درایه i-ام در سطر j-ام یک باشد، VM شماره i روی PM شماره j نگاشت یافته است. در هر سطر ماتریس تنها یک درایه مقدار ۱ و سایر درایه‌ها مقدار صفر دارند چرا که هر VM تنها می‌تواند به یک PM اختصاص یابد.

۳-۲- بهینه‌سازی دره انرژی

در این بخش ابتدا مبانی و سپس الگوریتم EVO [۹] تشریح شده است.

هسته اتم‌های ناپایدار برای رسیدن به پایداری با تشعشعاتی به فرم ذرات آلفا، ذرات بتا و اشعه‌های نوترونی و گاما، انرژی از دست می‌دهند. هسته اتم حاوی پروتون (p) و نوترون (n) است. پروتون دارای بار مثبت و نوترون دارای بار خنثی است چرا که نوترون از یک الکترون و یک پروتون تشکیل شده است. عدد اتمی یک اتم برابر تعداد پروتون (p) و عدد جرمی آن جمع تعداد پروتون و نوترون ($p+q$) است. عدد اتمی تعیین می‌کند که عنصر در کجای جدول تناوبی ظاهر شود. اختلاف عدد اتمی و عدد جرمی برابر تعداد نوترون است. نسبت n/p یک مشخصه مهم در پایداری هسته اتم است. چنانچه عدد اتمی یک عنصر کوچک باشد (کوچکتر یا مساوی ۲۰)، نسبت n/p یک اتم پایدار نزدیک به ۱ است. اما چنانچه عدد اتمی عنصری بزرگتر باشد، به دلیل ازدیاد پروتون‌ها که دارای بار مثبت بوده و یکدیگر را دفع می‌کنند، تعداد نوترون بیشتری نیاز است تا هسته متعادل باشد و نسبت n/p کمی بالاتر است. همچنین، چنانچه عدد اتمی از ۸۳ بزرگتر باشد، هسته اتم بسیار ناپایدار بوده و سعی می‌کند با ازدست‌دادن پروتون و نوترون به سمت تعادل حرکت کند. یک عنصر به فرم ${}^{p+n}_{p}\mathrm{S}$ نمایش داده می‌شود که S یکی از عناصر جدول تناوبی است. واکنش هسته‌ای می‌تواند باعث تبدیل هسته اتم از یک عنصر به یک عنصر دیگر شود. لازم به ذکر است که عدد اتمی هر عنصر ثابت است اما تعداد نوترون می‌تواند متفاوت از تعداد پروتون باشد. برای مثال عنصر کربن دارای عدد اتمی ۶ است اما می‌تواند ۶، ۷ یا ۸ نوترون داشته باشد و بنابراین عدد جرمی آن می‌تواند ۱۲، ۱۳ یا ۱۴ باشد. به عنصر با هر عدد جرمی یک ایزوتوپ گفته می‌شود. بنابراین کربن می‌تواند به یکی از صورت‌های ${}^{12}_{6}\mathrm{C}$، ${}^{13}_{6}\mathrm{C}$، یا ${}^{14}_{6}\mathrm{C}$ باشد که ایزوتوپ ۱۲ آن پایدارترین حالت آن است.



شکل ۱- نوار پایداری: ایزوتوپ‌های پایدار با نقاط سیاه و ایزوتوپ‌های ناپایدار با نقاط رنگی نمایش داده شده‌اند [۲۳]

[۲] Chemical Reaction Optimization

طبق نظریه فیزیک هسته‌ای، هسته اتم میل به رسیدن به پایداری دارد. برای پایدارشدن، اتم واکنش‌های هسته‌ای از خود نشان می‌دهد. هر اتم با توجه به مشخصاتش (عدد اتمی و عدد جرمی) می‌تواند هر بار یکی از واکنش‌ها را به خود ببیند و در صفحه n-p به سمت نوار پایداری حرکت کند. برای رسیدن به تعادل گاهی نیاز به زنجیره‌ای از واکنش‌هاست. شکل ۱ نوار پایداری اتم عناصر مختلف را نمایش می‌دهد. محور افقی تعداد پروتون و محور عمودی تعداد نوترون را نمایش می‌دهد. باند رنگی در صحفه $n-p$ نوار پایداری است و اتم‌ها در مرکز باند، که با نقاط سیاه نمایش داده شده‌اند، پایدارترند. اتم‌هایی که با مرکز فاصله دارند، با انجام واکنش سعی بر رسیدن به مرکز و پایدار شدن دارند. در اعداد بزرگ اتمی، اتم پایدار وجود ندارد و آنها برای رسیدن به پایداری باید به سمت پایین حرکت کنند. همچنین، در شکل ۱ مشخص است که اتم‌ها با عدد اتمی کوچک در نسبت $n/p=1$ پایدارند اما در هرچه عدد اتمی بزرگتر باشد نوار پایداری دارای نسبت n/p بزرگتری است. با رخداد یک واکنش هسته‌ای، اتم در صفحه به سمت نوار پایداری حرکت می‌کند. واکنش‌های هسته‌ای در ادامه تشریح شده‌اند.

۳-۳- واکنش‌های هسته‌ای

واکنش انتشار آلفا[۳]: در واکنش انتشار آلفا یا تحلیل آلفا[۴]، هسته اتم دو نوترون و دو پروتون از دست می‌دهد. به عبارت دیگر، عدد اتمی ۲ تا و عدد جرمی ۴ تا کم می‌شود و به همین دلیل این واکنش به صورت $^{4}_{2}\alpha$ نمایش داده می‌شود. این واکنش برای هسته‌ها با عدد اتمی و جرمی بسیار بالا (بزرگتر یا برابر ۸۴) رخ می‌دهد تا عنصر به سمت تعادل پیش برود. به فراخور رخداد این واکنش، عنصر ماهیت خود را از دست داده و به عنصر دیگری تبدیل می‌شود. از آنجا که در جدول تناوبی، اتم هلیوم به فرم $^{4}_{2}\mathrm{H}$ است، به جای $^{4}_{2}\alpha$ از نمایش $^{4}_{2}\mathrm{H}$ نیز استفاده می‌شود (در واقع آنچه در نتیجه این واکنش از هسته جدا می‌شود یک اتم هلیوم با بار مثبت (بدون الکترون) است). برای مثال زنون با عدد اتمی ۵۴ ($^{118}_{54}\mathrm{Xe}$) و نسبت n/p برابر ۱/۱۸۵ در اثر یک واکنش از نوع انتشار آلفا به تلوریوم به شکل $^{116}_{52}\mathrm{Te}$ تبدیل می‌شود. این واکنش به صورت $^{118}_{54}\mathrm{Xe} \rightarrow {}^{4}_{2}\mathrm{H} + {}^{114}_{52}\mathrm{Te}$ نمایش داده می‌شود. به دلیل قانون بقای ماده و انرژی، جمع p و همچنین جمع $n+p$ ها در دو سمت فلش (معادله) باید یکسان باشد. شایان ذکر است که نسبت n/p به ۱/۲۳ افزایش می‌یابد. هرچند این عدد بزرگتر است، اما در واکنش‌های بعدی این عدد می‌تواند کوچکتر شود.

واکنش انتشار بتا[۵]: این واکنش خود دو نوع دارد و هریک باعث تغییر در عدد n/p هسته اتم می‌شوند.

نوع اول واکنش انتشار بتا، زمانی رخ می‌دهد که هسته در بالای نوار پایداری قرار دارد و باید برای پایدار شدن نسبت n/p اش کاهش یابد. در این واکنش یک نوترون با آزاد کردن الکترون (ذره بتا) خود به یک پروتون تبدیل می‌شود. الکترون آزاد شده همان ذره بتا است و به فرم β^- یا $^{0}_{-1}e$ نمایش داده می‌شود. برای مثال یک واکنش از نوع انتشار آلفا روی کربن ایزوتوپ ۱۴ به صورت $^{14}_{6}\mathrm{C} \rightarrow {}^{0}_{-1}e + {}^{14}_{7}\mathrm{N}$ نمایش داده می‌شود که نشان می‌دهد کربن با آزاد کردن یک الکترون به نیتروژن تبدیل می‌شود. نسبت n/p از ۱/۳۳ به ۱ کاهش می‌یابد که برای عنصری با عدد اتمی کمتر از ۲۰ به معنی قرارگرفتن در مرکز نوار پایداری است.

در نوع دوم، که انتشار پوزیترون نام گرفته است، یک پروتون به یک نوترون تبدیل می‌شود. پوزیترون ذره‌ای با جرم برابر با الکترون اما بار مثبت است. در اثر این واکنش عدد اتمی یکی کم و عدد جرمی ثابت می‌ماند. این واکنش بر روی هسته‌ای که پایینتر از نوار پایداری قرار گرفته رخ می‌دهد و در اثر آن اتم به سمت نوار پایداری (شکل ۱) حرکت می‌کند. در نتیجه این واکنش، نسبت n/p افزایش می‌یابد. الکترون جذب شده پوزیترون یا بتا مثبت نام دارد و به فرم β^+، $^{0}_{1}e$ یا e^+ نمایش داده می‌شود. برای مثال منیزیم در طی انتشار پوزیترون به سدیم تبدیل می‌شود که به شکل $^{23}_{12}\mathrm{Mg} \rightarrow {}^{0}_{1}e + {}^{23}_{11}\mathrm{Na}$ نمایش داده می‌شود. نسبت n/p منیزیم برابر ۰/۹۲ است که و در سدیم برابر ۱/۱ تبدیل می‌شود و بنابراین عنصر از حالت ناپایدار به سمت نوار پایداری حرکت می‌کند.

واکنش گیراندازی الکترون[۶]: این واکنش مشابه انتشار پوزیترون است. یک الکترون از لایه‌های پایین هسته جذب شده و یک پروتون را به نوترون تبدیل می‌کند. برای مثال در واکنش $^{106}_{47}\mathrm{Ag} + {}^{0}_{-1}e \rightarrow {}^{106}_{46}\mathrm{Pd}$ نقره با گیراندازی الکترون به پالادیوم تبدیل می‌شود. نتیجه این واکنش مشابه واکنش انتشار پوزیترون است. هرچند، واکنش انتشار پوزیترون نیازمند انرژی بیشتری نسبت به گیراندازی الکترون است.

واکنش انتشار گاما[۷]: این واکنش برای زمانی است که هسته تحریک شده است و با آزاد کردن انرژی خود به شکل فوتون، پایدارتر می‌شود. در اثر این واکنش، هسته عنصر تغییری نمی‌کند و نسبت n/p نیز ثابت می‌ماند. این واکنش به شکل $^{0}_{0}\gamma$ نمایش داده می‌شود.

است.

۴-۳- الگوریتم EVO

الگوریتم EVO [۹] با الهام از واکنش‌های هسته‌ای طراحی شده است. جمعیت شامل تعدادی اتم است و هر اتم حاوی ساختاری است که پاسخ به مسئله در حال بهینه‌سازی را نگه می‌دارد. فرض کنید تعداد اتم برابر l و تعداد ابعاد مسئله برابر d باشد. جمعیت به فرم نمایش‌داده‌شده در معادله (۱) بیان می‌شود.

$$X=\begin{bmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_n^1 & \cdots & x_n^d \end{bmatrix} \quad (۱)$$

که x_i^j نشان‌دهنده j–امین بعد در i–امین اتم است و داریم $i \in \{1,\dots,n\}$ و $j \in \{1,\dots,d\}$. در اولین گام الگوریتم، مقداردهی اولیه، جمعیت اولیه تولید می‌شود. برای این‌کار برای تولید هر اتم به هر بعد یک مقدار مناسب اختصاص می‌یابد.

در هر مرحله از الگوریتم، ابتدا پارامتر کران غنی‌سازی یا EB[۸] با استفاده از معادله (۲) محاسبه می‌شود.

$$EB=\frac{\sum_{i=1}^{n} f(X_i)}{n} \quad (۲)$$

که $f(X_i)$ تابع شایستگی است. در گام بعدی برای هر اتم پارامتر سطح پایداری یا SL[۹] محاسبه می‌شود. سطح پایداری اتم i–ام که با SL_i نمایش می‌یابد با استفاده از معادله (۳) محاسبه می‌شود.

$$SL_i=\frac{f(X_i)-f(X_b)}{f(X_w)-f(X_b)} \quad (۳)$$

---

[۳] Alpha radiation

[۴] Alpha decay

[۵] Beta radiation

[۶] Electron capture

[۷] Gamma radiation

[۸] Enrichment bound

[۹] Stability Level

که X_b و $X_{w_{ww}}$ به ترتیب بهترین و بدترین اتم در جمعیت از اولین تکرار تا تکرار جاری و $f(X_b)$ و $f(X_w)$ به ترتیب مقدار شایستگی آنها هستند. با داشتن EB و SL_i برای هر ذره تعیین می‌شود که کدام‌یک از واکنش‌های هسته‌ای رخ دهد.

اگر برای یک اتم (X_i) شرط $f(X_i) > EB$ برقرار باشد، فرض می‌شود اتم دارای عدد n/p بزرگی باشد و بنابراین می‌توان با اعمال واکنش‌هایی از قبیل انتشار آلفا، بتا و گاما اتم را به سمت پایداری برد. یک عدد تصادفی در محدوده [۰و۱] به عنوان مقدار SB تولید می‌شود. چنانچه برای اتم i-ام شرط $SL_i > SB$ برقرار باشد، واکنش‌های انتشار آلفا و گاما اعمال می‌شود چرا که این دو واکنش برای اتم‌های سنگین با سطح پایداری بالاتر محتمل هستند.

برای نمایش اثر یک واکنش انتشار آلفا، که باعث بهبود پایداری اتم می‌شود، ساختار اتم تغییر داده می‌شود تا یک اتم جدید ایجاد شود. دو عدد تصادفی $r1$ در بازه [d و۱] و $r2$ در بازه [r_1 و ۱] تولید می‌شود که از آن برای جایگزینی مقدار عنصر i-ام در اتم با مقدار متناظر در اتم دارای بهترین پاسخ استفاده می‌شود. در نتیجه این عمل یک اتم جدید ایجاد می‌شود.

برای واکنش انتشار گاما، فرایندی مشابه با واکنش انتشار آلفا انجام می‌شود اما جایگزینی یک عنصر در اتم اصلی با مقدار متناظر در یک اتم همسایه انجام می‌شود. برای انتخاب یک اتم همسایه، مفهوم فاصله تعریف شده که با معادله (۴) محاسبه می‌شود.

$$D_i^k = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (۴)$$

که D_i^k فاصله بین اتم i-ام و k-ام است. (x_1, y_1) و (x_2, y_2) موقعیت اتم‌ها i-ام و k-ام در فضای جستجو هستند. اتم دارای کمترین مقدار فاصله محاسبه‌شده به عنوان همسایه انتخاب می‌شود و در جایگزینی عنصر مورد استفاده قرار می‌گیرد.

برای واکنش از نوع بتا، مفهومی با نام مرکز اتم‌ها تعریف می‌شود. مرکز اتم‌ها برای کل جمعیت از معادله (۵) محاسبه می‌شود.

$$X_{CP} = \frac{\sum_{i=1}^{n} X_i}{n} \quad (۵)$$

معادله (۵) مقدار میانگین تمامی عناصر در هر بعد را محاسبه می‌کند. سپس، اتم جدید با استفاده از معادله (۶) تولید می‌شود.

$$X_i^{New1} = X_i + \frac{r_1 \times X_{BS} - r_2 \times X_{CP}}{SL_i} \quad (۶)$$

که X_{BS} ساختار مربوط به اتم با بهترین سطح پایداری (SL) است و X_{CP} مقدار مرکزی محاسبه شده با معادله (۵) است. همچنین، r_1 و r_2 دو عدد تصادفی بین ۰ و ۱ هستند که میزان جابجایی اتم را تعیین می‌کنند. X_i^{New1} اتم جدید است. برای بهبود الگوریتم، یک عملگر دیگر نیز برای واکنش از نوع بتا تعریف شده است که توسط معادله (۷) بیان می‌شود.

$$X_i^{New2} = X_i + r_3 \times X_{BS} - r_4 \times X_{NG} \quad (۷)$$

که X_{NG} اتم همسایه است. همچنین، r_3 و r_4 دو عدد تصادفی در محدوده [۰و۱] هستند. X_i^{New2} اتم جدید پس از اعمال عملگر است.

چنانچه شرط $f(X_i) \leq EB$ ارضا شود، مبین این‌است که اتم دارای یک نسبت n/p کوچک است و در این حالت دو واکنش پوزیترون و گیراندازی الکترون محتمل است. برای چنین حالتی عملگر نشان‌داده‌شده در معادله (۸) اعمال می‌شود تا یک اتم جدید که با X_i^{New} نمایش داده می‌شود، تولید شود.

$$X_i^{New} = X_i + r \quad (۸)$$

که X_i اتم کنونی، و r یک عدد تصادفی در محدوده [۰و۱] است.

در پایان هر تکرار الگوریتم، اتم یا اتم‌های جدید تولیدشده به جمعیت اضافه می‌شوند.

۴- الگوریتم EVO برای جانمایی ماشین‌های مجازی

برای حل مسئله جانمایی، d در الگوریتم (تعداد بعد) برابر همان تعداد VM یا m است. هر ساختار اتمی با یک توالی تصادفی از اندیس VM ها ایجاد می‌شود. در الگوریتم اصلی ارائه‌شده در [۹]، در هر تکرار به تعداد اتم‌ها واکنش رخ داده و برخی از واکنش‌ها باعث ایجاد دو اتم جدید در جمعیت می‌شوند و بنابراین تعداد اتم به صورت نمایی افزایش می‌یابد. در مسئله جانمایی ماشین، چنانچه در بخش ۲-۱ تشریح شد، هر اتم حاوی لیست تمامی ماشین‌های مجازی است و عملگرهای مورد نیاز برای واکنش‌ها پیچیده هستند. بنابراین اگر اندازه جمعیت به صورت نمایی افزایش یابد، زمان اجرای الگوریتم بسیار بالا خواهد رفت. برای حل این مشکل در هر تکرار به ازای هر اتم تنها یک اتم جدید تولید می‌شود. به ازای هر اتم یک واکنش و در صورتی که واکنش دو اتم جدید تولید کند، تنها یکی از آن‌ها به تصادف انتخاب می‌شود.

الگوریتم ۱ نمای کلی الگوریتم EVO را نشان می‌دهد. تابع `rand(1.0)` یک عدد تصادفی در محدوده [۰و۱] تولید می‌کند. در الگوریتم ابتدا یک جمعیت تصادفی تولید و ارزیابی می‌شود. سپس به تعداد تکرار مورد نظر الگوریتم اجرا می‌شود. در هر تکرار ابتدا کران غنی‌سازی (EB) محاسبه و اتم با بهترین میزان پایداری تعیین می‌شود. در گام بعدی با توجه به برازش اتم و مقدار EB تعیین می‌شود واکنش پوزیترون اعمال شود یا نه. در حالت دوم، کران پایداری جمعیت محاسبه و برای تعیین اعمال واکنش آلفا/گاما یا بتا استفاده می‌شود. اگر واکنش آلفا/گاما انتخاب شود، یکی از آنها به تصادف برای تولید اتم جدید به‌کار گرفته می‌شوند. در حالت واکنش بتا، دو اتم جدید، یکی بر پایه اتم مرکز و دیگری اتم همسایه تولید می‌شود که یکی از این دو حالت به تصادف انتخاب می‌شود. در هر مرحله اتم تولیدی به جمعیت جدید افزوده می‌شود و در انتهای هر تکرار جمعیت اتم‌های جدید جایگزین جمعیت اتم‌های قبلی می‌شود. در انتهای اجرای الگوریتم، بهترین دارای بهترین شایستگی، X_{BS} به عنوان پاسخ تولیدی الگوریتم بازگردانده می‌شود. در ادامه، پس از تشریح نحوه نمایش پاسخ جانمایی ماشین‌های مجازی، عملگرهای که منجر به تولید یک اتم جدید (با نام X^{New}) می‌شوند تشریح شده‌اند.

۴-۱- نمایش نگاشت ماشین مجازی

هر اتم در الگوریتم یک پاسخ برای نگاشت ماشین‌های مجازی روی ماشین‌های فیزیکی فراهم می‌کند. این پاسخ باید به صورت مناسبی در اتم نمایش یابد.

روش استفاده شده در این پژوهش به‌این صورت است که هر اتم توالی‌ای از اندیس‌های ماشین‌های مجازی را نگه می‌دارد و در زمان محاسبه مقدار تابع شایستگی، ابتدا با الگوریتم اولین برازش یا FF توالی VMها روی ماشین‌های فیزیکی نگاشت می‌یابد و سپس پارامترهای هدف (توان مصرفی، هدررفت منابع) محاسبه می‌شوند. برای مثال، شکل ۲ را در نظر بگیرید. در بالای شکل مجموعه ماشین‌های مجازی (با مقدار مشخصی از درخواست پردازش و حافظه) و در پایین شکل یک نگاشت نمونه بر روی دو PM با ظرفیت پردازش و حافظه مشخص نمایش داده شده است.

۴-۲- عملگر انتشار آلفا

عملگر انتشار آلفا به صورت تصادفی یک اندیس بین ۰ تا $m-1$ به نام i' و سپس یک اندیس بین ۰ تا i' به نام i''، تولید می‌کند. پس از آن، از ابتدا تا اندیس i'' در اتم منتخب با مقادیر متناظر در اتم X_{BS} پر می‌شوند. از آنجا که

هر اندیس تنها یک‌بار امکان ظهور در ساختار اتم را دارد (چرا که اندیس ماشین‌های مجازی یکتا هستند)، و همه اندیس‌های مربوط به VMها باید در پاسخ وجود داشته باشند تا پاسخ معتبر باشد، پس از مرحله اول، اندیس‌های تکراری حذف و اندیس‌های غایب به صورت تصادفی و در محل‌های i'' به بعد قرار می‌گیرند.

الگوریتم EVO

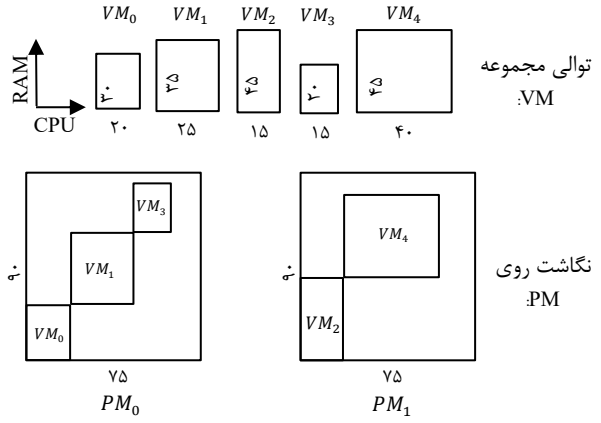
```
begin Algorithm
Pop ← initialize();
Pop.fit ← calculateFitness(Pop);
while(termination criterion not met)
   EB ← calculateEnrichmentBound(Pop);
   X_BS ← findAtomWithBestStabilityLevel(Pop);
   for all atom X_i in Pop
      if(f(X_i) > EB) then
         SB ← calculateStabilityBound(Pop);
         if(SL_i > SB) then
                if(rand(1.0)<0.5) then
                 X_i^New ←applyAlphaReaction(X_i);
                else
                 X_NG ←findNeighborAtom(X_i);
                 X_i^New ←applyGammaReaction(X_i);
                end if
         else if SL_i <= SB
                if(rand(1.0)<0.5) then
                 X_CP ←calculateCenterAtom(Pop);
                 X_i^New ←applyBetaReaction1(X_i,X_CP);
                else
                 X_NG ←findNeighborAtom(X_i);
                 X_i^New ←applyBetaReaction2(X_i,X_NG);
                end if
         end if
      else //f(X_i) ≤ EB
         X_i^New ←PositronReaction(X_i);
      end if
        PopNew.add(X_i^New);
   end for
   Pop ← PopNew;
end while
return(findAtomWithBestStabilityLevel(Pop));
end Algorithm
```

الگوریتم ۱: الگوریتم بهینه‌سازی دره انرژی [۹]

به‌عنوان مثال فرض کنید $VM=\{0,1,2,3,4,5,6,7\}$ باشد (m=۸). همچنین، فرض کنید پاسخ یک اتم منتخب برای اعمال عملگر دارای ساختار $X=\{2,0,7,4,3,1,5,6\}$ باشد. بعلاوه، فرض کنید اتم دارای بهترین پایداری دارای ساختار $X_{BS}=\{7,5,0,1,3,4,2,6\}$ باشد. اگر i'' برابر ۳ باشد، ابتدا اتم جدید به صورت $X^{New}=\{\mathbf{7},\mathbf{5},\mathbf{0},\mathbf{1},3,1,5,6\}$ (قسمت پررنگ همان تکه‌ای است که از X_{BS} برداشته شده است) تولید و سپس اندیس‌های ۱ و ۵ که در اندیس i'' به بعد در اتم تکراری هستند حذف می‌شوند بنابراین داریم $X^{New}=\{\mathbf{7},\mathbf{5},\mathbf{0},\mathbf{1},3,-,-,6\}$ که محل‌های خالی پس از حذف اندیس‌های تکراری با – مشخص شده‌اند. در نهایت اندیس‌های ۲ و ۴ که غایب هستند، برای ایجاد یک ساختار معتبر به صورت تصادفی در اتم قرار می‌گیرند: $X^{New}=\{7,5,0,1,3,4,2,6\}$.

٤-٣- عملگر انتشار گاما

عملگر انتشار گاما مشابه انتشار آلفا عمل می‌کند با این تفاوت که به جای استفاده از X_{BS} از X_{NG} یا اتم همسایه استفاده می‌کند. برای انتخاب اتم همسایه، بدین صورت عمل می‌شود که برای هر VM در هر محل، میزان درخواست پردازش و حافظه آن با سایر VMها به عنوان دو پارامتر x و y در معادله (۴) قرار می‌گیرد. سپس اتم با کوچکترین مجموع فاصله به عنوان اتم همسایه در نظر گرفته می‌شود و عملگر با استفاده از آن تکمیل می‌شود تا X^{New} حاصل شود.



شکل ۲-:مثالی از یک توالی ماشین مجازی و نگاشت آن بر روی دو ماشین فیزیکی با FF

٤-٤- عملگر انتشار بتا

عملگر پیشنهادی در الگوریتم EVO برای مسئله جانمایی ماشین مجازی نیازمند بازطراحی است. در عملگر مورد استفاده در این پژوهش، دو اتم جدید از اتم دارای بهترین شایستگی (X_{BS}) و اتم مرکز (X_{CP}) تولید می‌شود. دو عملگر به کار گرفته می‌شوند: عملگر چرخش و عملگر شیفت. یک عدد تصادفی در محدوده [۱و۰] تولید و اگر کوچکتر از ۰/۵ باشد عملگر شیفت و اگر بزرگتر از ۰/۵ باشد عملگر چرخش اعمال می‌شود. برای تولید دو اتم جدید، این فرایند یکبار به ازای X_{BS} و یکبار به ازای X_{CP} انجام می‌شود.

عملگر چرخش: در عملگر چرخش ساختار اتم به راست چرخش داده می‌شود. تعداد چرخش به صورت تصادفی بین ۱ تا $\frac{m}{2}$ تعیین می‌شود و هر اندیس VM که از انتهای ساختار خارج می‌شود به ابتدای آن منتقل می‌شود. برای مثال فرض کنید $X_{BS}=\{4,2,0,1,5,7,6,3\}$ و تعداد چرخش ۲ باشد بنابراین داریم $X^{New}=\{\mathbf{6},\mathbf{3},4,2,0,1,5,7\}$

عملگر برش: ساختار اتم از انتها به تعداد عددی تصادفی بین ۱ تا $\frac{m}{2}$ برش داده می‌شود و اندیس‌هایی که خارج می‌شوند، به صورت تصادفی به اتم الحاق می‌شوند. برای مثال فرض کنید $X_{CP}=\{5,4,2,0,6,7,3,1\}$ و اندازه برش ۳ باشد. ابتدا سه اندیس انتهایی بریده می‌شود و داریم $X^{New}=\{5,4,2,0,6\}$ و مجموعه $\{7,3,1\}$ اندیس‌های بریده‌شده هستند که به صورت تصادفی در ساختار اتم قرار می‌گیرند: $X^{New}=\{5,4,2,\mathbf{3},0,\mathbf{7},6,\mathbf{1}\}$.

برای تعیین اتم X_{CP}، بدین صورت عمل می‌شود که برای هر اندیس در تمامی جمعیت مقدار میانگین پردازش و میانگین حافظه به‌صورت جداگانه محاسبه می‌شود. سپس، اتم با کمترین میزان فاصله با مقدار میانگین به عنوان X_{CP} انتخاب می‌شود. محاسبه فاصله برای هر اندیس، i، با استفاده از معادله (۴) انجام می‌شود که در آن (x_1,y_1) مقادیر میانگین درخواست پردازش و حافظه در اندیس i-ام و (x_2,y_2) مقادیر درخواست پردازش و حافظه اندیس i-ام اتم k-ام، اتمی که فاصله آن در حال محاسبه است، می‌باشد.

۴-۵- عملگر انتشار پوزیترون/گیراندازی الکترون

این عملگر با تغییر X_{BS} یک اتم جدید ایجاد می‌کند. ابتدا یک عدد تصادفی بین ۱ و $\frac{m}{2}$ تولید می‌شود و اتم X_{BS} به تعداد آن به راست شیفت داده می‌شود. سپس، اندیس‌های خارج‌شده به صورت تصادفی در انتهای ساختار اتم الحاق می‌شوند. برای مثال فرض کنید دو اتم که $X_{BS} = \{5,4,2,0,6,7,3,1\}$ و باشد و عدد تصادفی شیفت برابر ۳ باشد. در ابتدا داریم: $X^{New} = \{5,4,2,0,6\}$ سپس با الحاق مجموعه اندیس‌های خارج‌شده، {7,3,1} به انتهای اتم، داریم: $X^{New} = \{5,4,2,0,6, \mathbf{1,7,3}\}$.

۴-۶- فرمول‌بندی مسئله بهینه‌سازی

مسئله نگاشت ماشین‌های مجازی بر روی ماشین‌های فیزیکی را می‌توان به صورت یک مسئله بهینه‌سازی مقید فرمول‌بندی کرد. هدف الگوریتم، کاهش مصرف توان، کاهش هدررفت منابع و بهبود میزان استفاده از منابع است.

محاسبه توان مصرفی و هدرفت منابع: برای محاسبه توان مصرفی از معادله (۹) استفاده می‌شود که مبین ارتباط بین توان مصرفی با میزان استفاده از منبع پردازشی است [۲۴].

$$P_j = P_{idle} + (U_j^{CPU} \times (P_{max} - P_{idle})) \quad (۹)$$

که P_j توان مصرفی ماشین فیزیکی j-ام است، P_{idle} و P_{max} توان مصرفی در حالت معلق و حداکثر توان مصرفی ماشین فیزیکی هستند، و U_j^{CPU} میزان استفاده منبع CPU در ماشین فیزیکی است. توان مصرفی کل از جمع توان مصرفی همه ماشین‌های فیزیکی فعال بدست می‌آید. هدف بهینه‌سازی کمینه‌کردن توان مصرفی در مرکز داده، مطابق با معادله (۱۰)، است:

$$\begin{aligned} &minimize \ \textstyle\sum_{j=1}^{n} P_j \\ &\text{s.t} \\ &\textstyle\sum_{i=1}^{m} x_{i,j} = 1 \qquad \forall j \in \{1,\dots,n\} \\ &\textstyle\sum_{j=1}^{n} R_i^l \times x_{i,j} \le C_j^l \quad \forall i \in \{1,\dots,m\}, l \in \{CPU, RAM\} \end{aligned} \quad (۱۰)$$

که قید اول مشخص می‌کند هر ماشین مجازی به یک ماشین فیزیکی اختصاص می‌یابد که $x_{i,j}$ یک متغیر بولی است که اگر ماشین مجازی i-ام به ماشین فیزیکی j-ام اختصاص یافته باشد یک و در غیر این‌صورت صفر است. قید دوم مشخص می‌کند که جمع منابع درخواستی همه ماشین‌های مجازی (شامل پردازش و حافظه) اختصاص‌یافته به یک ماشین فیزیکی نباید از ظرفیت ماشین فیزیکی تجاوز کند که R_i^l نشان‌دهنده میزان منبع (پردازش/حافظه) درخواستی توسط ماشین مجازی i-ام است.

میزان استفاده[۱۰] از یک منبع (شامل پردازش و حافظه) در هر ماشین فیزیکی، از ماتریس نگاشت همه ماشین‌های مجازی نگاشت یافته روی ماشین فیزیکی تعیین و میزان منابع تجمیعی آن‌ها به صورت جداگانه محاسبه و بر ظرفیت PM تقسیم می‌شود. برای مثال اگر مجموعه ماشین‌های مجازی {2,5,6} که دارای درخواست پردازش برابر {100,200,50} هستند روی یک PM با ظرفیت پردازشی ۴۰۰ جانمایی شوند، میزان استفاده منبع پردازش برابر ۰/۸۷۵ خواهد بود. میزان استفاده کل از میانگین میزان استفاده در همه PMها محاسبه می‌شود.

علاوه بر توان مصرفی و میزان استفاده، هدررفت منابع در ماشین فیزیکی j-ام که با W_j نشان داده می‌شود، به عنوان یک پارامتر اضافه مطابق با معادله (۱۱) محاسبه می‌شود [۱۹, ۲۵].

$$W_j = \frac{|L_j^p - L_j^m| + \epsilon}{U_j^p - U_j^m} \quad (۱۱)$$

که L_j^p و L_j^m به ترتیب میزان فضای خالی باقیمانده منبع پردازش و حافظه نرمال‌شده در ماشین فیزیکی j-ام و ϵ یک عدد کوچک است. همچنین U_j^p و U_j^m میزان استفاده منابع پردازش و حافظه در ماشین مجازی j-ام هستند. هدررفت منابع کل از جمع هدررفت منابع همه PMهای فعال محاسبه می‌شود.

۵- ارزیابی و نتایج

سیستم مورد استفاده: الگوریتم EVO جهت ارزیابی با استفاده از زبان برنامه‌نویسی C++ پیاده‌سازی شد. برای اجرا از یک کامپیوتر مجهز به پردازنده Intel® Core i5 دارای ۸ گیگابایت حافظه رم استفاده شد.

پارامترهای مورد بررسی: توان مصرفی به عنوان پارامتر مورد سنجش اصلی در نظر گرفته شد و در الگوریتم EVO نیز توان مصرفی (معادله (۹)) به عنوان تابع هدف لحاظ شد. در کنار آن تعداد PM فعال، میزان استفاده از منابع، هدررفت منابع و زمان اجرا اندازه‌گیری شد. نتایج ارائه شده میانگین ۴ بار اجرای هر الگوریتم هستند.

مدل ماشین‌های فیزیکی و مجازی: برای بررسی هرچه بهتر الگوریتم‌ها، از مدل ارائه‌شده در [۲۶] که توسط بسیاری از محققان مورد استفاده قرار گرفته، برای تولید ماشین‌های مجازی (با میزان درخواست CPU و RAM با همبستگی‌های مختلف) استفاده شد. الگوریتم تولید ماشین مجازی در الگوریتم ۲ آورده شده است. پارامتر ρ میزان همبستگی بین مقادیر تولیدی برای پردازش و حافظه را تعیین می‌کند و این پارامتر برابر یکی از مقادیر $\rho = \{0.0,0.25,0.50,0.75,1.0\}$ انتخاب می‌شود. همچنین، دو پارامتر ورودی $\overline{R^{CPU}}$ و $\overline{R^{\mathrm{RAM}}}$ در همه شبیه‌سازی‌ها برابر ۰/۴۵ تنظیم شدند. الگوریتم به VM با اندیس i مقدار پردازش برابر R_i^{CPU} و مقدار حافظه برابر R_i^{CPU} اختصاص می‌دهد. در الگوریتم، $rand(a)$ یک عدد تصادفی بین ۰ تا a و با توزیع نرمال تولید می‌کند. از آنجا که الگوریتم میزان یک عدد بین ۰ تا ۱ تولید می‌کند، این عدد در ۴۰ ضرب شد و با عدد ۱۰ جمع شد تا عدد منبع پردازشی و حافظه حاصل شود. این اعداد با توجه به مقادیر ظرفیت اختصاصی به PMها (جدول ۲) انتخاب شد. پارامترها مطابق جدول ۱ تنظیم شد.

Input: $m, \rho, \overline{R^{CPU}}, \overline{R^{\mathrm{RAM}}}$
Output: $\{VM = \{x \in i | \ 0 \le \ i \le m-1\}$
for i **in** 0 **to** m − 1 **do**
  $\mathrm{R_i^{CPU}} \leftarrow$ rand$(2 \times \overline{\mathrm{R^{CPU}}})$
  $R_i^{\mathrm{RAM}} \leftarrow$ rand$(\overline{R^{\mathrm{RAM}}})$
  $r \leftarrow$ rand(1.0)
  if $(r < \rho$ **and** $R_i^{\mathrm{CPU}} \ge R_i^{\mathrm{RAM}})$ **or** $(r \ge \rho$ **and** $R_i^{\mathrm{CPU}} < R_i^{\mathrm{RAM}})$ **then**
    $R_i^{\mathrm{RAM}} \leftarrow R_i^{\mathrm{RAM}} + \overline{R^{RAM}}$
  end if
 $VM_i \leftarrow$ createVM$(R_i^{\mathrm{CPU}}, R_i^{\mathrm{RAM}})$
end for

الگوریتم ۲: مولد ماشین مجازی

جدول ۱: پارامترهای شبیه‌سازی

نماد	توضیح	مقدار
$\overline{R^{RAM}}$ و $\overline{R^{CPU}}$	پارامترهای مولد وظیفه	۰/۴۵
$\boldsymbol{\rho}$	همبستگی منبع پردازش و حافظه در مولد VM	{0,0.25,0.5,0.75,1}
ϵ	مقدار کوچک ثابت در معادله (۱۱)	۰/۰۰۰۰۱
$Pop.size$	اندازه جمعیت در EVO (تعداد اتم)	۱۰۰
termination criterion	معیار اتمام الگوریتم (تکرار)	۱۰۰

[۹] Utilization

برای ماشین‌های فیزیکی، چهار نوع ماشین در نظر گرفته شد که مشخصات آنها در جدول ۲ لیست شده است. در زمان تولید مجموعه ماشین‌های فیزیکی هربار یک نوع ماشین به صورت تصادفی انتخاب شد.

جدول ۲: مشخصه‌های PM

نوع PM	ظرفیت پردازش	ظرفیت حافظه	P_{idle}	P_{max}
۱	۱۰۰	۱۰۰	۱۰۰	۱۴۵
۲	۲۰۰	۲۰۰	۱۵۰	۲۱۴
۳	۴۰۰	۴۰۰	۱۷۵	۲۵۰
۴	۶۰۰	۶۰۰	۲۰۰	۲۸۵

۵-۱- نتایج

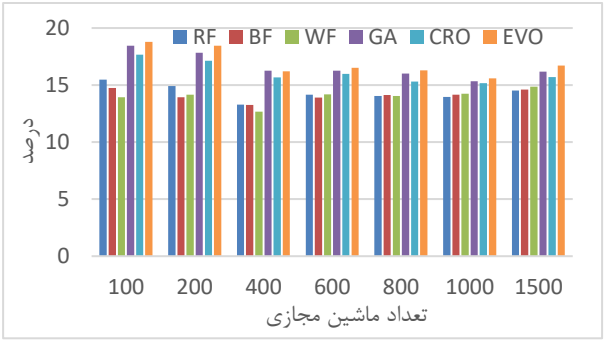
در این بخش، نتایج تولیدشده توسط الگوریتم‌ها به ازای سناریوهای مختلف ارائه شده است.

الگوریتم‌های مورد استفاده در ارزیابی در زیر معرفی شده‌اند.

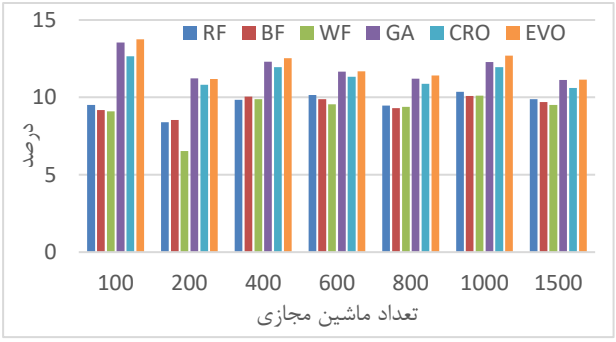
- ✓ الگوریتم FFD: قبل از جانمایی، ماشین‌های مجازی با توجه به حجم درخواست پردازشی به صورت کاهشی مرتب می‌شوند. سپس از ابتدای PMها، هر VM از ابتدای لیست به اولین PM که فضای کافی برای قبول آن را داشته اختصاص می‌یابد. این الگوریتم به عنوان الگوریتم پایه در نظر گرفته شده است و مقادیر آورده شده در نتایج با بر اساس نتیجه تولیدی توسط این الگوریتم محاسبه شده‌اند.
- ✓ الگوریتم BF: برای هر VM فضای پردازشی آزاد در همه PM ها محاسبه و VM به PM دارای کمترین اختلاف بین فضای آزاد با مقدار درخواست پردازش در VM را داشته باشد.
- ✓ الگوریتم WF: مشابه الگوریتم BF است اما PM دارای بیشترین اختلاف فضای آزاد با اندازه پردازش درخواستی VM انتخاب می‌شود.
- ✓ الگوریتم RF: برای هر VM یک PM به صورت تصادفی انتخاب و VM به آن اختصاص می‌یابد.
- ✓ الگوریتم ژنتیک (GA) [[۱۵]]: با نرخ برش ۰/۸، نرخ جهش ۰/۱، جمعیت ۱۰۰ و تکرار ۱۰۰.
- ✓ الگوریتم بهینه‌سازی واکنش شیمیایی (CRO[۱۱]) [[۲۲]]: با نرخ برخورد ۰/۶ و نرخ افت ۰/۲، جمعیت ۱۰۰ و تکرار ۱۰۰.
- ✓ الگوریتم EVO: الگوریتم پیشنهادی در مقاله.

شکل ۳ توان مصرفی چهار الگوریتم را نسبت به الگوریتم پایه، FFD، نمایش می‌دهد. توان مصرفی به ازای VM های تولیدشده به ازای مقادیر مختلف ρ (ضریب همبستگی)، شامل $\rho = \{0.0,0.25,0.50,0.75,1.0\}$ آورده شده است. محور افقی در شکل ۳ تعداد VM است که از ۱۰۰ تا ۱۵۰۰ متغیر است و محور عمودی بهبود توان مصرفی نسبت به FFD را نمایش می‌دهد. همانطور که نتایج نشان می‌دهند، الگوریتم EVO در تمامی ارزیابی‌ها نتایج بهتری را تولید کرده است. در حالی که میانگین بهبود EVO نسبت به FFD در $\rho = 0.0$ و $\rho = 0.25$ بین حدود ۱۳ تا ۱۵ درصد بوده، برای $\rho = \{0.5,0.75,1\}$ که میزان همبستگی بین پردازش و حافظه بیشتر است و مقادیر آنها اختلاف کمتری دارد، بهبود کاهش یافته و از حدود ۱۰ درصد به حدود ۶ درصد و سپس به حدود ۳ درصد رسیده است. برای سایر الگوریتم‌ها نیز همین روند مشاهده می‌شود. در مقایسه با CRO، EVO دارای نتایجی حدود نیم تا ۱ درصد بهتر است و در مقایسه با GA، بهبود کمتر از ۱ درصدی مشاهده شد. در اعداد همبستگی پایین الگوریتم‌های BF، RF و WF حدود ۱۰ درصد بهتر از FFD عمل کرده‌اند در حالی که برای عدد همبستگی یک در برخی موارد بدتر از FFD عمل کرده‌اند. این مسئله نشانگر این واقعیت است که الگوریتم FFD برای اعداد همبستگی پایین خوب عمل نمی‌کند اما برای اعداد همبستگی بالا خوب عمل می‌کند چرا که تنها منبع پردازشی را در نظر می‌گیرد و زمانی که عدد همبستگی بالاست منبع پردازش نماینده خوبی از کل منابع (پردازش و حافظه) است. همچنین، در اعداد کوچک‌ترین و بزرگترین ضریب همبستگی، هرچه تعداد VM افزایش می‌یابد، بهبود الگوریتم‌های مختلف اندکی بالا می‌رود اما در سایر مقادیر روند بهبود تقریبا ثابت است.

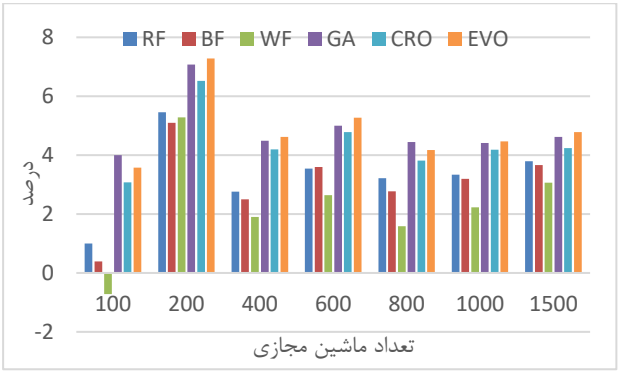
شکل ۴-الف و ۴-ب به ترتیب میزان استفاده از منابع پردازشی و حافظه را در الگوریتم‌های مختلف و به ازای ضرایب همبستگی مختلف و تعداد VM برابر ۱۵۰۰ نشان می‌دهد. همانطور که در شکل ۴ مشخص است، با افزایش همبستگی اختلاف بین FFD و سایر الگوریتم‌ها کاهش می‌یابد. EVO (با میانگین ۹۲/۸ و ۹۴/۴ درصدی برای پردازش و حافظه) و GA (با ۹۳/۴ و ۹۴ درصدی برای پردازش و حافظه) دارای بهترین میزان استفاده بوده‌اند و پس از آن CRO (میانگین ۹۲/۹ و ۹۳/۶) و RF (میانگین ۹۲/۶ و ۹۴/۶ درصد) و BF (۹۲/۴ و ۹۲/۲ درصد) بوده‌اند که دلیل آن این‌است که RF با توزیع یکنواخت و مناسبی پاسخ را تولید می‌کند و BF نیز سعی بر یافتن بهترین PM که دارای حداقل هدررفت منابع است می‌کند. الگوریتم WF نیز دارای میانگین استفاده ۹۱/۱ و ۹۱ درصدی برای پردازش و حافظه بوده است.



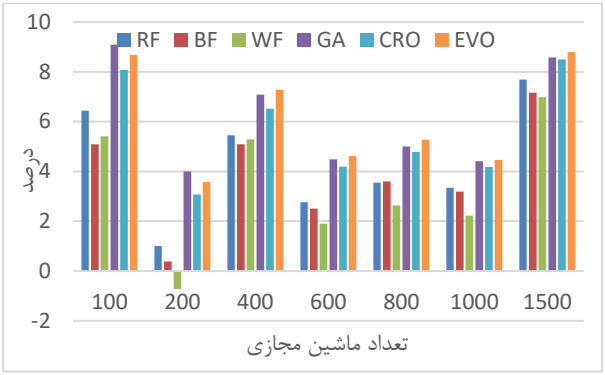
(الف)



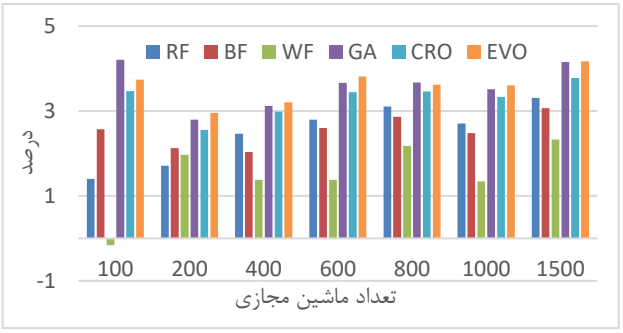
(ب)



(د)

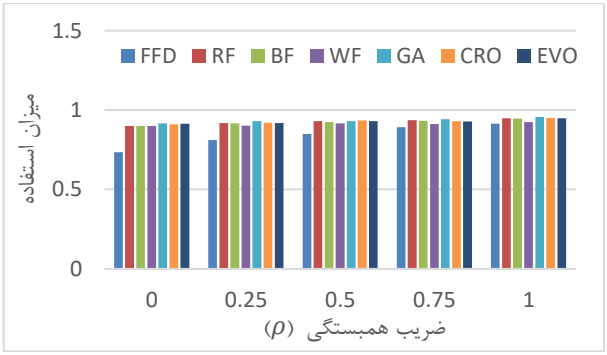


(ج)

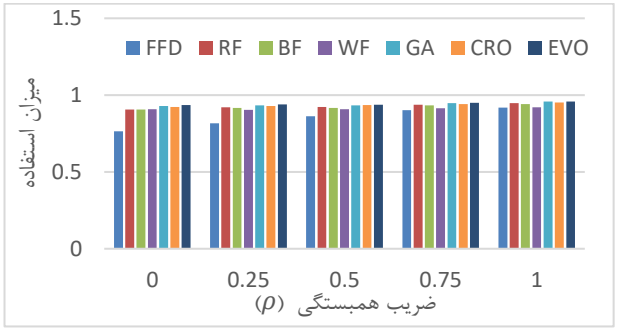


(ه)

شکل ۳- درصد بهبود توان مصرفی در الگوریتم‌های مختلف نسبت به FFD به ازای تعداد مختلف ماشین مجازی ۱۵۰۰: (الف) $\rho = 0.0$، (ب) $\rho = 0.25$، (ج) $\rho = 0.5$، (د) $\rho = 0.75$ و (ه) $\rho = 1.0$.



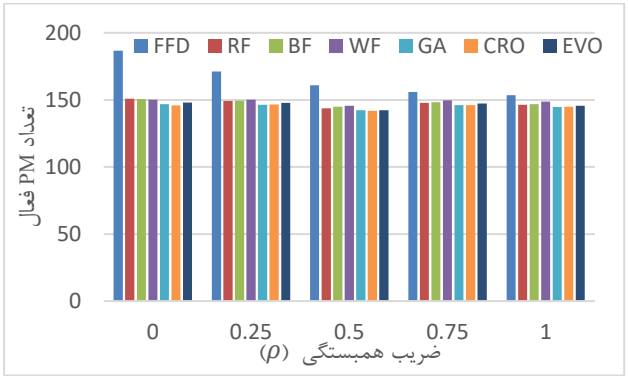
(الف)



(ب)

شکل ۴- درصد میزان استفاده از منابع به ازای تعداد VM برابر ۱۵۰۰ و مقادیر مختلف *P*: (الف) پردازش (CPU)، (ب) حافظه (RAM).

یک روند مشاهده شده نیز این‌است که با افزایش ضریب همبستگی میزان استفاده کمی بهبود می‌یابد. شکل ۵ تعداد PM فعال به ازای تعداد VM برابر ۱۵۰۰ و ضرایب همبستگی مختلف را نشان می‌دهد که EVO دارای کمترین (میانگین ۱۴۵/۱۲) و FFD دارای بیشترین (میانگین ۱۶۵/۶) تعداد PM فعال بوده است. این مقادیر برای GA، CRO، RF، BF، و WF به ترتیب ۱۴۵/۳، ۱۴۶/۲، ۱۴۷/۶، ۱۴۸، و ۱۴۸/۹ بدست آمد.

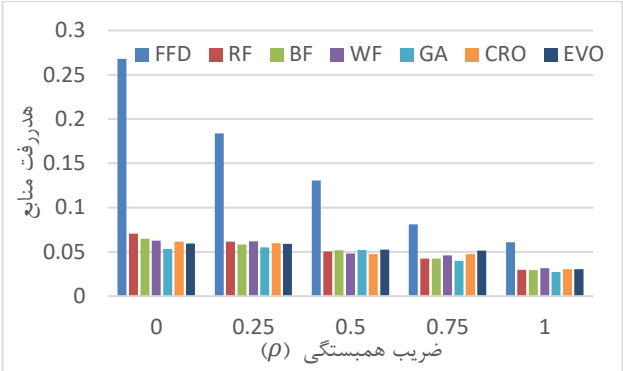


شکل ۵- تعداد PM فعال به ازای مقادیر مختلف *P*

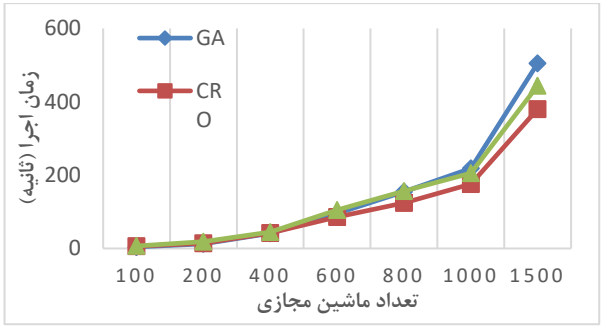
شکل ۶ هدررفت منابع را (معادله (۱۱)) به ازای ۱۵۰۰ عدد VM و ضرایب همبستگی مختلف نمایش می‌دهد. به طور کلی GA از سایر الگوریتم‌ها بهتر عمل کرده است هرچند اختلاف زیادی بین الگوریتم‌ها، بجز FFD مشاهده نشد.

شکل ۷ زمان اجرای سه الگوریتم فرابتکاری به ازای تعداد مختلف VM و ضریب همبستگی ۰/۵ را نشان می‌دهد. زمان اجرای مربوط به سایر ضرایب همبستگی اختلاف چندانی با ضریب همبستگی ۰/۵ نداشته و تعداد VM اثر اصلی را دارد. زمان اجرای سایر الگوریتم‌ها به ازای تعداد VM برابر ۲۰۰ به چند ده یا چندصد میلی ثانیه محدود بوده و الگوریتم‌های BF و WF که نیازمند بررسی همه PM ها برای یافتن بهترین و بدترین هستند زمان اجرای بیشتری داشته‌اند.

الگوریتم‌های فراابتکاری قادرند کاهش مناسبی در توان مصرفی به همراه بیاورند. به نظر می‌رسد نیاز به تغییراتی در الگوریتم EVO وجود دارد، از جمله اینکه می‌توان پارامتر مهم n/p که یک معیار مهم در رسیدن به پایداری (و مقدار بهینه) است را به صورت بهتری در الگوریتم مدل کرد. زمان اجرای الگوریتم‌های فراابتکاری نسبتا بالا هستند و دلیل آن انجام تکراری عملگرهای مربوطه بر روی اعضای جمعیت است. به طور کلی، در مقایسه با الگوریتم GA، EVO نتایج کمی بهتر تولید می‌کند و زمان اجرای آن بهتر است. در مقایسه با CRO به عنوان یک الگوریتم جدید و مشابه EVO، نتایج تولیدی توسط EVO بهتر است اما زمان اجرای آن کمی بالاتر است. از سوی دیگر، تنظیم پارامترهای الگوریتم CRO دشوار بوده و برای رسیدن به تعادل مناسب در تعداد واکنش‌ها دشوار و بسیار وابسته به مسئله است اما تنظیما پارامترهای EVO بسیار ساده‌تر است.



شکل ۶- هدررفت منابع (معادله (۱۱)) به ازای مقادیر مختلف P



شکل ۷- زمان اجرای الگوریتم EVO به ازای تعداد مختلف VM

۶- نتیجه‌گیری

پژوهش حاضر، در راستای نظریه هیچ *ناهاری مجانی نیست*، الگوریتم نوظهور EVO برای حل مسئله جانمایی ماشین‌های مجازی روی فیزیکی به عنوان یک مسئله ان‌پی-سخت به کار گرفته شد. یک مزیت این الگوریتم عدم وجود پارامترهای ورودی مختلف و فراوان در الگوریتم است. چندین عملگر مناسب طراحی و در الگوریتم، که نیاز به اصلاحاتی برای اجرای بهتر داشت، استفاده شدند. نتایج ارزیابی مبین این واقعیت است که الگوریتم قادر است به طور مناسبی بهتر از الگوریتم‌های مختلف ابتکاری عمل کرده و توان مصرفی را کاهش دهد. در آینده می‌توان در جهت اعمال اصلاحاتی جهت بهبود الگوریتم گام برداشت. به‌کارگیری الگوریتم در مدل مرکزداده پویا و در نظر گرفتن میزان استفاده در VM و استفاده از روش اختصاص فضای بیش از ظرفیت (oversubscription) و محاسبه کیفیت سطح سرویس (Service-Level Agreement) در آینده مد نظر مولف است. همچنین، استفاده از الگوریتم برای لحاظ پارامترهای مهم دیگر در مراکز داده ابری از جمله در نظر گرفتن انتشار کربن و محاسبات سبز از جمله کارهای مدنظر برای انجام در آینده است.

مراجع

[1] J. Bekesi, G. Galambos, and H. Kellerer, "A 5/4 linear time bin packing algorithm", Journal of Computer and System Sciences, vol. 60, no. 1, pp. 145-160, 2000.

[2] H. Talebian *et al.*, "Optimizing virtual machine placement in iaas data centers: taxonomy, review and open issues", Cluster Computing, vol. 23, pp. 837-878, 2020.

[3] D. Alsadie, "Virtual Machine Placement Methods using Metaheuristic Algorithms in a Cloud Environment-A Comprehensive Review", International Journal of Computer Science & Network Security, vol. 22, no. 4, pp. 147-158, 2022.

[4] J. P. Gabhane, S. Pathak, and N. M. Thakare, "Metaheuristics algorithms for virtual machine placement in cloud computing environments—a review", Computer Networks, Big Data and IoT: Proceedings of ICCBI 2020, pp. 329-349, 2021.

[5] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various VM placement strategies in CloudSim", Journal of Cloud Computing, vol. 4, no. 1, pp. 1-21, 2015.

[6] N. Khattar, J. Sidhu, and J. Singh, "Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques", The Journal of Supercomputing, vol. 75, pp. 4750-4810, 2019.

[7] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges", Artificial Intelligence Review, pp. 1-71, 2023.

[8] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", IEEE transactions on evolutionary computation, vol. 1, no. 1, pp. 67-82, 1997.

[9] M. Azizi, U. Aickelin, H. A. Khorshidi, and M. Baghalzadeh Shishehgarkhaneh, "Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization", Scientific Reports, vol. 13, no. 1, p. 226, 2023.

[10] B. Pourghebleh, A. Aghaei Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments", Cluster Computing, vol. 24, no. 3, pp. 2673-2696, 2021/09/01 2021, doi: 10.1007/s10586-021-03294-4.

[11] G. Wu, M. Tang, Y.-C. Tian, and W. Li, "Energy-efficient virtual machine placement in data centers by genetic algorithm", in Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part III 19, 2012: Springer, pp. 315-323.

[12] B. Zhang, X. Wang, and H. Wang, "Virtual machine placement strategy using cluster-based genetic algorithm", Neurocomputing, vol. 428, pp. 310-316, 2021.

[13] K. Braiki and H. Youssef, "Multi-objective virtual machine placement algorithm based on particle swarm optimization", in 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018: IEEE, pp. 279-284.

[14] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers", Expert Systems with Applications, vol. 120, pp. 228-238, 2019.

[15] M. Kiani and M. R. Khayyambashi, "A network-aware and power-efficient virtual machine placement scheme in cloud datacenters based on chemical reaction optimization", Computer Networks, vol. 196, p. 108270, 2021.

[16] K. Balaji, P. Sai Kiran, and M. Sunil Kumar, "Power aware virtual machine placement in IaaS cloud using discrete firefly algorithm", Applied Nanoscience, vol. 13, no. 3, pp. 2003-2011, 2023.

[17] N. Donyagard Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review", International Journal of Communication Systems, vol. 32, no. 14, p. e4068, 2019.

[18] M. HS, P. Gupta, and G. McArdle, "A Harris Hawk Optimisation system for energy and resource efficient virtual machine placement in cloud data centers", Plos one, vol. 18, no. 8, p. e0289156, 2023.

[19] 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", Journal of computer and system sciences, vol. 79, no. 8, pp. 1230-1242, 2013.

[20] Jamali, Malektaji, Analoui, "Virtual Machine Location Using the Imperialist Competitive Algorithm", Journal of Electrical Engineering, University of Tabriz, Volume 46, Number 1, Spring 2016.

[21] Naeini, Salem, Rashedi, "Using the Hybrid Frog Leaping Algorithm to Reduce Energy Consumption of Cloud Data Centers through Optimization of Task Scheduling Management and Effective Composition of Virtual Machines", Journal of Electrical Engineering, University of Tabriz, vol. 48, no. 2, Summer 2018.

[22] Z. Li, Y. Li, T. Yuan, S. Chen, and S. Jiang, "Chemical reaction optimization for virtual machine placement in cloud computing", Applied Intelligence, vol. 49, pp. 220-232, 2019.

[23] Wikipedia. "https://en.wikipedia.org/wiki/Table_of_nuclides#Isotopes_for_elements_75-89." (accessed 29 October, 2023).

[24] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer", ACM SIGARCH computer architecture news, vol. 35, no. 2, pp. 13-23, 2007.

[25] A. S. Abohamama and E. Hamouda, "A hybrid energy–aware virtual machine placement algorithm for cloud environments", Expert Systems with Applications, vol. 150, p. 113306, 2020.

[26] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation", in int. CMG Conference, 2007, vol. 253, pp. 399-406.