الگوریتم‌های زمانبندی جدید برای بهینه‌سازی هزینه در گریدهای محاسباتی اقتصادی

یاسر مهدوی‌فر*، محمدرضا میبدی†

چکیده

منابع در گریدهای محاسباتی اقتصادی دارای قیمت هستند و کاربر باید هزینه اجرای کارهای خود را بپردازد. کاربر مهلت زمانی و بودجه مورد نظر خود را مشخص کرده و بهینه‌سازی هزینه یا زمان را درخواست می‌کند. یک الگوریتم زمانبندی که استراتژی بهینه‌سازی هزینه را اتخاذ می‌کند، باید منابع ناهمگون گرید را طوری به کارهای ناهمگون کاربر تخصیص دهد که اجرای آنها در مهلت تعیین شده و با کمترین هزینه ممکن انجام شود. در این مقاله، سه الگوریتم مکاشفه‌ای جدید برای این منظور پیشنهاد شده است. با استفاده از شبیه‌سازی نشان داده شده است که الگوریتم‌های پیشنهادی در مقایسه با تنها الگوریتم گزارش شده از کارایی بالاتری برخوردار بوده و درخواست‌های کاربر را با هزینه کمتری انجام می‌دهند.

واژه‌های کلیدی

گرید محاسباتی، زمانبندی اقتصادی، بهینه‌سازی هزینه.

New Scheduling Algorithms for Cost Optimization in Economic Computational Grids

Yasser Mahdavifar, Mohammad Reza Meybodi

Abstract

In economic computational grids, resources have prices and the users must pay for executing their applications. The user determines his deadline and budget and then requests cost or time optimization. A scheduling algorithm that adopts cost optimization strategy, should allocate heterogeneous grid resources to heterogeneous user jobs so that their execution finishes in the specified deadline with minimum cost. In this paper, three new algorithms for are proposed for this purpose. Using computer simulations, It is shown that the proposed algorithms have higher performance comparing to the existing algorithms.

Keywords

Computational Grid, Economic Scheduling, Cost Optimization.

* دانشجوی کارشناسی ارشد، دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، mahdavifar@ce.aut.ac.ir
† استاد و عضو هیأت علمی دانشگاه، دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، meybodi@ce.aut.ac.ir

۱- مقدمه

گریدهای محاسباتی[۱] یکی از رویکردهای جدید برای حل مسائل در زمینه‌های علمی، مهندسی و تجاری در مقیاس بزرگ می‌باشند [1-4]. آنها بستری را برای به اشتراک گذاشتن و یکپارچه‌سازی میلیونها منبع که از نظر جغرافیایی در سطح سازمان‌ها و حوزه‌های مدیریتی مختلف پراکنده‌اند بوجود آورده‌اند. گریدها از مجموعه‌ای از منابع ناهمگون[۲] (کامپیوترهای شخصی، ایستگاه‌های کاری[۳]، کلاسترها و ابرکامپیوترها)، سیستم‌های مدیریت زیربنایی (سیستم عامل واحد، سیستم صف و غیره)، سیاست‌ها و برنامه‌های کاربردی (علمی، مهندسی و تجاری) با نیازمندی‌های مختلف (پردازنده، ورودی و خروجی، حافظه و شبکه) تشکیل شده‌اند. مشتریان گرید با پرداخت هزینه برای درخواست خود، می‌توانند از این منابع استفاده کنند. صاحبان و استفاده‌کنندگان منابع دارای اهداف، استراتژی‌ها و الگوهای عرضه و تقاضای متفاوتی هستند. در چنین شرایطی نمی‌توان از راهکارهای متداول برای مدیریت منابع که سعی می‌کنند میزان کارایی کل سیستم را بهبود دهند، استفاده کرد. برای این منظور، در سال‌های اخیر از رویکردهای اقتصادی برای مدیریت تخصیص منابع در گرید استفاده شده است [5-8]. یکی از مدل‌های اقتصادی که بدین منظور استفاده شده است، مدل بازار کالا[۴] می‌باشد. در این مدل، هر منبع دارای قیمت مشخصی است که بر اساس عرضه، تقاضا و ارزش در سیستم اقتصادی تعیین شده است.

مسائلی مانند ناهمگونی[۵]، توزیع شدگی و اشتراکی بودن منابع در گرید و همچنین محدودیت‌هایی مانند مهلت[۶] (زمان اتمام اجرای برنامه) و بودجه (هزینه محاسبات) که توسط کاربران تعیین می‌شود، باعث پیچیدگی عمل زمانبندی برنامه کاربر می‌گردد. یکی از استراتژی‌هایی که یک الگوریتم زمانبندی می‌تواند اتخاذ کند، مینیمم کردن هزینه در محدوده مهلت تعیین شده (بهینه‌سازی هزینه) می‌باشد. تا آنجا که نگارندگان این مقاله مطلع می‌باشند، تاکنون تنها یک الگوریتم مکاشفه‌ای[۷] که آن را BCO[8] می‌نامیم، برای مینیمم کردن هزینه در محدوده مهلت در مدل اقتصادی بازار کالا توسط بویا در [9,10] گزارش شده و در [11] مورد ارزیابی قرار گرفته است. این الگوریتم برای زمانبندی برنامه‌های کاربردی پارامترروب[۹] که شامل تعداد زیادی کارهای همگون مستقل از هم می‌باشند، طراحی شده است و به همین دلیل برای کارهای ناهمگون، جوابهای قابل قبولی را به کاربر ارائه نمی‌کند.

در این مقاله، ۳ الگوریتم مکاشفه‌ای جدید که آنها را ABCO[10]، EBCO[11] و AEBCO[12] می‌نامیم و بر اساس الگوریتم BCO طراحی شده‌اند، برای زمانبندی کارهای مستقل از هم با هدف بهینه‌سازی هزینه در گریدهای محاسباتی اقتصادی با مدل بازار کالا پیشنهاد می‌شوند. الگوریتم‌های پیشنهادی با استفاده از جعبه‌ابزار GridSim [11] شبیه‌سازی شده و کارایی‌های آنها مورد بررسی قرار گرفته است. نتایج بدست آمده با نتایج الگوریتم بویا مقایسه شده است. ادامه مقاله بدین صورت سازماندهی شده است. در بخش ۲ زمانبندی اقتصادی در گرید شرح داده می‌شود. در بخش ۳ الگوریتم‌های زمانبندی پیشنهادی ارائه می‌گردد. بخش ۴ اختصاص به نتایج شبیه‌سازی‌ها دارد و بخش ۵ نتیجه‌گیری می‌باشد.

۲- زمانبندی اقتصادی در گرید

کاربر گرید پس از اینکه محدودیت‌های مهلت زمانی و بودجه خود را مشخص کرد، ممکن است بهینه‌سازی هزینه و یا زمان را درخواست کند. در صورتی که کاربر خواستار بهینه‌سازی هزینه باشد، الگوریتم زمانبندی باید در محدوده مهلتی که کاربر تعیین کرده است، کمترین هزینه را برای اجرای کارها بدست آورد. الگوریتم‌های زمانبندی با هدف بهینه‌سازی هزینه و یا زمان از دو نوع می‌باشند: زمانبندی مرحله‌ای و زمانبندی یکباره. الگوریتم‌هایی که از روش زمانبندی مرحله‌ای استفاده می‌کنند، در طول اجرای خود به تدریج کارها را به منابع موجود در گرید واگذار می‌کنند. در مقابل، الگوریتم‌هایی که رویکرد زمانبندی یکباره را اتخاذ می‌کنند، یک نگاشت از کارهای برنامه کاربر به منابع موجود را تولید می‌کنند که برای بررسی به بخش کنترل پذیرش داده می‌شود. پس از بررسی، اگر نگاشت بدست آمده، نیازمندی‌های کاربر را برآورده کند، کارها طبق این نگاشت به منابع برای اجرا واگذار می‌گردد.

الگوریتم‌های بهینه‌سازی هزینه و یا زمان، دارای چند مرحله مشترک ابتدایی می‌باشند. این مراحل عبارتند از: یافتن، تجارت و مرتب‌سازی منابع. الگوریتم‌هایی که از روش زمانبندی یکباره استفاده می‌کنند، در پایان نیز دارای دو مرحله مشترک کنترل پذیرش و توزیع هستند. به این ترتیب، ورودی همه الگوریتم‌های زمانبندی با هدف بهینه‌سازی هزینه و یا زمان، یک لیست مرتب شده از منابع موجود در گرید بر اساس قیمت مفید می‌باشد.

قیمت اعلام شده برای یک منبع، مقدار هزینه استفاده از آن در واحد زمان می‌باشد و قیمت مفید یک منبع به مقدار هزینه استفاده از آن برای اجرای هر MI گفته می‌شود. در این مقاله، واژه‌های قیمت، ارزانی و گرانی با توجه به قیمت مفید منبع به کار می‌روند.

طول یک کار، تعداد دستورالعمل‌های آن می‌باشد و بر حسب واحد MI[13] (میلیون دستورالعمل) اندازه‌گیری می‌شود. اگر کارهای یک برنامه کاربردی، کاملاً همگون[۱۴] باشند (طول یکسان داشته باشند)، می‌توان الگوریتم‌های زمانبندی با پیچیدگی چندجمله‌ای برای مینیمم کردن هزینه در محدوده مهلت، طراحی کرد. ولی هنگامی که کارها دارای ناهمگونی باشند (طول‌های متفاوت داشته باشند)، مسأله زمانبندی با هدف بهینه‌سازی هزینه به یک مسأله NP-Complete تبدیل می‌شود.

اجرای هر برنامه کاربردی بر روی گرید، در مهلت تعیین شده، دارای یک کران پایین یا مینیمم برای هزینه محاسبات می‌باشد. البته هیچ سیستم زمانبندی وجود ندارد که بتواند کارها را با مجموع

هزینه‌ای برابر با مینیمم اجرا کند. الگوریتم‌های زمانبندی نیز سعی می‌کنند تا حد ممکن هزینه محاسبات را به هزینه مینیمم نزدیک کنند. اگر مجموع طول همه کارها برابر با S میلیون دستورالعمل باشد، آنگاه برای بدست آوردن مینیمم هزینه، باید هزینه اجرای این مجموع را بدون در نظر گرفتن هیچ کدام از کارها، بر روی گرید محاسبه کرد. بنابراین محاسبه می‌شود که هر کدام از منابع (با شروع از ارزان‌ترین منبع) چه تعدادی از این S دستور را می‌توانند پیش از مهلت، اجرا کنند و هزینه اجرای آن چقدر است. مجموع این هزینه‌ها همان مینیمم هزینه محاسبات خواهد بود. در واقع، در این زمانبندی که برای دستورها (به عنوان کوچکترین واحد اجرایی) انجام می‌شود (و نه برای کارها)، زمان بیکاری منابعی که در زمانبندی شرکت می‌کنند (به جز منبع آخر) صفر خواهد بود و به این ترتیب، هزینه بدست آمده، کمترین هزینه ممکن می‌باشد.

۲-۱- زمانبندی مرحله‌ای و زمانبندی یکباره

با توجه به اینکه منابع در گرید، بصورت زمان‌مشترک[۱۵] و یا فضامشترک[۱۶] باشند، تأثیر زیادی بر زمانبندی کارها خواهد داشت. منابع زمان مشترک می‌توانند در هر لحظه در حال اجرای چندین کار باشند و به همین دلیل هر کاری که به یک منبع زمان‌مشترک واگذار شود، بلافاصله برای اجرا به یکی از پردازنده‌های آن منبع منتقل می‌گردد. در مقابل هر پردازنده فضامشترک در هر زمان فقط به اجرای یک کار مشغول می‌باشد. در منابع فضامشترک یک صف انتظار از کارهای پردازش نشده وجود دارد که به محض بیکار شدن یک پردازنده، اولین کار در این صف، برای اجرا به آن پردازنده منتقل می‌شود.

یکی از مشکلات منابع زمان‌مشترک عدم توانایی در پیش‌بینی دقیق زمان اتمام اجرای کارهای واگذار شده توسط این منابع می‌باشد. یک راه حل برای رفع این مشکل این است که به هر منبع فقط به تعداد پردازنده‌های موجود در آن، کار واگذار شود تا از این طریق از اجرای همزمان چند کار بر روی یک پردازنده جلوگیری شود. برای این منظور می‌توان عمل زمانبندی را به صورت یک عمل مرحله‌ای انجام داد که تا پایان واگذاری همه کارها ادامه می‌یابد. در این راهکار، به هر منبع فقط به تعداد پردازنده‌های موجود در آن، کار واگذار می‌شود تا از اجرای همزمان چند کار بر روی یک پردازنده جلوگیری شود. برای این کار برای هر کاربر، یک صف تشکیل و کارهای واگذار نشده برنامه او در این صف نگهداری می‌شود. در زمان تحویل یک برنامه کاربردی به گرید، کارهای این برنامه توسط الگوریتم زمانبندی به منابع در گرید نگاشت می‌شود و در هنگام واگذاری کارها به این منابع، به هر منبع فقط به تعداد پردازنده‌های بیکار آن، کار واگذار می‌شود. کارهای باقیمانده سپس در صف کاربر قرار داده می‌شود. در مرحله بعد، مجدداً الگوریتم زمانبندی اجرا و منابعی برای نگاشت انتخاب می‌شوند که باز به تعداد پردازنده‌های بیکار به آنها کار واگذار می‌شود. این فرایند ادامه می‌یابد تا همه کارهای کاربر به منابع واگذار شده باشد. فاصله زمانی بین هر دو مرحله از این فرایند بایستی آنقدر بزرگ در نظر گرفته شود که احتمال بیکار شدن لااقل یک پردازنده، بالا باشد. البته ممکن است در شروع یک مرحله، هیچ منبعی دارای پردازنده بیکار نباشد که در این صورت در این مرحله، کاری واگذار نمی‌شود.

بویا از روش زمانبندی مرحله‌ای برای زمانبندی با اهداف بهینه‌سازی هزینه و زمان استفاده کرده است. استفاده از زمانبندی مرحله‌ای برای حل مشکلات منابع زمان‌مشترکی که در گرید وجود دارند، مفید است؛ زیرا با واگذاری کارها به تعداد پردازنده‌های بیکار هر منبع، از اجرای همزمان بیش از یک کار بر روی یک پردازنده جلوگیری کرده و امکان پیش‌بینی زمان اتمام اجرای کارها را فراهم می‌کند. همچنین با کنترل تعداد کارهای واگذار شده به هر منبع، از بوجود آمدن وضعیت اضافه‌بار[۱۷] جلوگیری می‌شود.

برای گریدهای اقتصادی که در آنها تضمین کیفیت سرویس نیز مد نظر است، روش زمانبندی مرحله‌ای روش مناسبی نیست؛ زیرا با ورود یک کاربر جدید به گرید و ایجاد تداخل بین زمانبندی برنامه کاربر جدید و برنامه کاربر قبلی، برای اجرای برنامه کاربردی کاربر قبلی بایستی هزینه و زمان بیشتری را (نسبت به مقادیری که گرید قبل از ورود کاربر جدید، برای کاربر قبلی متعهد شده است) مصرف نمود. این مسأله منجر به نارضایتی کاربران که مشتریان گرید هستند خواهد شد و در نتیجه استفاده از منابع در گرید و سود صاحبان منابع کاهش می‌یابد. راهکاری که برای حل این مشکل در این مقاله پیشنهاد می‌شود این است که با همه منابع گرید به صورت فضامشترک رفتار شود و از اجرای بیش از یک کار بر روی هر پردازنده یک منبع جلوگیری شود یا به بیان دیگر از روش زمانبندی یکباره استفاده شود. برای انجام این کار بر خلاف زمانبندی مرحله‌ای، که برای هر کاربر یک صف در سطح گرید تشکیل می‌شود و همه کارهای واگذار نشده مربوط به او در آن قرار می‌گیرد، برای هر منبع یک صف تشکیل و کارهای واگذار شده به این منبع به ترتیب واگذاری، در آن نگهداری شود. از این طریق هر منبع بر تعداد کارهای در حال اجرا بر روی هر کدام از پردازنده‌های خود کنترل داشته و خود را فقط به صورت فضا مشترک در اختیار کارها قرار می‌دهد و با توجه به صف کارهای متعلق به خود، آنها را به ترتیب، به پردازنده‌هایی که بیکار می‌شوند منتقل می‌کند.

۲-۲- مراحل مشترک الگوریتم‌های زمانبندی

الگوریتم‌های بهینه‌سازی هزینه و یا زمان، دارای چند مرحله مشترک ابتدایی می‌باشند. این مراحل عبارتند از یافتن، تجارت و مرتب‌سازی منابع. الگوریتم‌هایی که از روش زمانبندی یکباره استفاده می‌کنند، در پایان نیز دارای دو مرحله مشترک کنترل پذیرش و توزیع هستند. در ادامه توضیحاتی درباره این مراحل داده می‌شود [10].

- **یافتن منابع**[۱۸]: شناسایی منابعی که می‌توانند در اجرای کارها مورد استفاده قرار بگیرند و همچنین بدست آوردن توانمندی‌ها و

ویژگی‌های آنها که از طریق سرویس اطلاعاتی گرید انجام می‌گیرد.

- **تجارت منابع**[19]: شناسایی هزینه هر یک از منابع در واحد زمان (G$/sec) ، شناسایی میزان توانمندی منبع در واحد زمان (نرخ اجرای میلیون دستورالعمل در ثانیه، MI/sec) و محاسبه قیمت واقعی و مفید منبع که همان هزینه اجرای یک میلیون دستورالعمل در ثانیه می‌باشد (G$/MI).
- **مرتب‌سازی منابع**: یک الگوریتم بهینه‌سازی، حتماً به یک ترتیب خاص از منابع بر حسب قیمت آنها نیاز دارد. در این مرحله، منابع به ترتیب صعودی از نظر قیمت مفید مرتب‌سازی می‌شوند. در واقع، اولویت واگذاری کار با منابع ارزان‌تر می‌باشد. اگر دو منبع دارای قیمت مفید یکسان باشند، منبعی که توانمندی بیشتری دارد در اولویت قرار می‌گیرد.
- **کنترل پذیرش**[20]: هزینه کل اجرای کارها بر روی منابع بر طبق زمانبندی مشخص و با بودجه تعیین شده توسط کاربر مقایسه می‌گردد. همچنین زمان تقریبی اجرای برنامه بر طبق زمانبندی محاسبه و سپس با مهلت تعیین شده توسط کاربر مقایسه می‌شود. در صورتی که هزینه‌ها در محدوده بودجه باشد و اجرای برنامه قبل از مهلت زمانی پایان می‌یابد، کاربر پذیرفته می‌شود.
- **توزیع**[21]: در این مرحله در صورت پذیرش کاربر، کارها بر طبق نگاشت تعیین شده، به منابع واگذار می‌شود.

۲-۳- الگوریتم BCO

الگوریتم BCO تنها الگوریتم زمانبندی گزارش شده با هدف مینیمم کردن هزینه در محدوده مهلت تعیین شده در گریدهای محاسباتی اقتصادی با مدل بازار کالا تا پیش از نگارش این مقاله می‌باشد. این الگوریتم توسط بویا در [9,10] گزارش شده و در [11] مورد ارزیابی قرار گرفته است. الگوریتم BCO کارها را در صف کارهای پردازش نشده از نظر طول به صورت صعودی مرتب می‌کند و سپس با در نظر گرفتن مهلت تعیین شده برای اتمام کارها، به هر منبع (به ترتیب صعودی بر حسب قیمت)، تا جایی که اجرای کارها با توجه به مهلت تعیین شده تمام می‌شود، کار منتسب می‌کند. به این ترتیب، مرحله زمانبندی انجام می‌شود. در مرحله توزیع، کارها به منابع منتسب شده، واگذار می‌شوند. البته به هر منبع، حداکثر به تعداد پردازنده‌های بیکاری که دارد، کار واگذار می‌شود. کارهای باقیمانده مجدداً به صف کارهای پردازش نشده، برگردانده می‌شوند. در الگوریتم BCO عمل زمانبندی به صورت مرحله‌ای انجام می‌گیرد. الگوریتم زمانبندی BCO به طور دقیق‌تر در شکل (۱) آورده شده است.

تا زمانی که کارهای پردازش نشده وجود دارد و از محدودیت‌های بودجه و مهلت تجاوز نکرده‌ایم، تکرار کن:

۱. **زمانبندی:**
   a. برای هر منبع انجام بده:
      i. کارهایی را که به منبع واگذار شده‌اند ولی اجرای آنها هنوز شروع نشده است، به صف کارهای پردازش نشده منتقل کن.
   b. صف کارهای پردازش نشده را با توجه به طول کارها به صورت صعودی مرتب کن.
   c. تا هنگامی که کار پردازش نشده وجود دارد، برای هر منبع به ترتیب (با شروع از ارزان‌ترین منبع) انجام بده:
      i. با توجه به کارهای واگذار شده، حداکثر تعداد کارهایی را که در ابتدای صف کارهای پردازش نشده قرار دارند و منبع می‌تواند اجرای آنها را قبل از مهلت تمام کند مشخص کن.
      ii. این کارها را از صف کارهای پردازش نشده حذف کرده و به منبع فعلی منتسب کن.
۲. **توزیع:** تعداد کارهایی را که می‌توان به یک منبع واگذار کرد، بدون اینکه با حالت اضافه‌بار مواجه شود، مشخص کرده و آنها را واگذار کن. سپس بقیه کارها را به صف کارهای پردازش نشده برگردان. سیاست پیش‌فرض این است که تعداد کارهای واگذار شده به یک منبع از تعداد پردازنده‌های آن بیشتر نباشد.
۳. برای مدت زمان مشخصی الگوریتم را متوقف کن. در این مدت زمان باید احتمال بیکار شدن حداقل یکی از پردازنده‌های منابع وجود داشته باشد.

شکل (۱): الگوریتم BCO

۳- الگوریتم‌های پیشنهادی

الگوریتم اول (ABCO): اگر کارهای کوچک را به یک منبع واگذار کنیم، فشردگی کارها باعث می‌شود تا زمان بیکاری منبع تا مهلت تعیین شده کم باشد. از آنجا که زمان در منابع گرانتر ارزش بیشتری دارد، بایستی کارهای کوچک را به آنها و کارهای بزرگتر را به منابع ارزانتر واگذار کنیم. الگوریتم ABCO برخلاف الگوریتم BCO، در ابتدای مرحله زمانبندی، صف کارهای پردازش نشده را بر حسب طول کارها به صورت نزولی مرتب می‌کند. این تغییر باعث می‌شود که زمان بیکاری منابع گرانتر کاهش پیدا کند. طبق نتایج شبیه‌سازی‌ها این تغییر باعث بهبود قابل ملاحظه‌ای در الگوریتم BCO می‌شود.

الگوریتم دوم (EBCO): الگوریتم EBCO از ایده الگوریتم BCO استفاده می‌کند با این تفاوت که برای زمانبندی کارهای کاربر روش زمانبندی یکباره را بکار می‌گیرد. به این ترتیب، امکان تضمین پارامترهای کیفیت سرویس را برای چندین کاربر به طور همزمان فراهم می‌سازد. الگوریتم EBCO از دو مرحله تشکیل شده است: در مرحله اول، کارها بر حسب طول به صورت نزولی مرتب می‌شوند و در مرحله دوم به هر منبع (به ترتیب صعودی قیمت) تا جایی که امکان

دارد کارها (به ترتیب نزولی طول) نگاشت می‌شوند. الگوریتم EBCO ابتدا ارزان‌ترین منبع را انتخاب می‌کند و کارها را تا جایی که مهلت زمانی نقض نشود، به آن نگاشت می‌کند. در صورت نقض مهلت، دومین منبع ارزان را انتخاب کرده و کارها را به آن نگاشت می‌کند. این فرایند تا تخصیص همه کارها ادامه پیدا می‌کند.

الگوریتم سوم (AEBCO): همانطور که گفته شد، در دو الگوریتم قبلی کارها به صورت نزولی مرتب می‌شوند و بهمین دلیل نگاشت کارهای کوچکتر بعد از نگاشت کارهای بزرگتر انجام می‌گیرد. بنابراین به تدریج در هنگام زمانبندی کارها ممکن است بتوان از منابعی که قبلاً قادر به اتمام کار در مهلت تعیین شده، نبودند، برای واگذاری کارهای کوچکتر استفاده کرد. الگوریتم AEBCO این مشکل را رفع می‌کند. بنابراین برای هر کار، همه منابع را به ترتیب از ابتدا بررسی کرده و به محض برخورد با منبعی که بتواند کار را در مهلت تعیین شده باتمام برساند، کار به آن واگذار می‌شود. الگوریتم AEBCO همانند الگوریتم EBCO از دو مرحله اصلی تشکیل شده است: در مرحله اول، کارها بر حسب طول به صورت نزولی مرتب می‌شوند و در مرحله دوم برای هر کار، ارزان‌ترین منبعی که بتواند کار را در مهلت تعیین شده تمام کند، انتخاب شده و کار به آن نگاشت می‌شود.

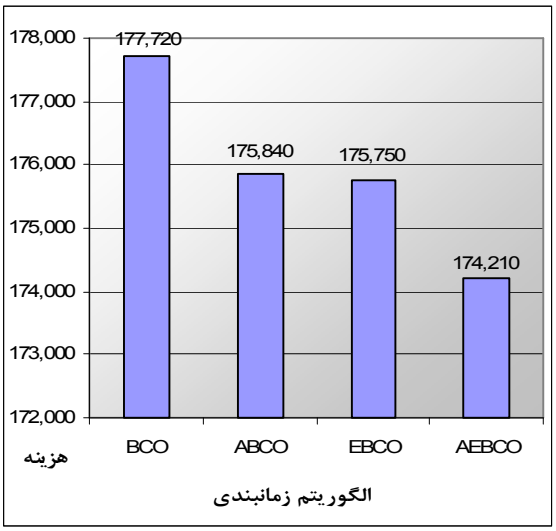
۴- نتایج شبیه‌سازی‌ها

الگوریتم‌های پیشنهادی با استفاده از جعبه‌ابزار GridSim [11] شبیه‌سازی شده و نتایج آنها با نتایج بدست آمده از الگوریتم بویا (BCO) مقایسه شده است. نتایج گزارش شده، میانگین ۲۰ بار شبیه‌سازی می‌باشد. محیط شبیه‌سازی شده برای گرید، شامل تعدادی منبع و یک کاربر است. کلیه منابع محاسباتی دارای یک پردازنده هستند که مشخصات آنها در جدول (۱) داده شده است. همانطور که مشاهده می‌شود، ناهمگونی بالایی برای منابع در نظر گرفته شده است. برنامه کاربر از ۲۰۰ کار مستقل از هم تشکیل شده است که طول هر کار به صورت تصادفی از محدوده (۲۰۰۰۰۰...۱۰۰۰۰) انتخاب می‌شود. گستردگی این محدوده، منجر به ناهمگونی کارها می‌گردد. کاربر مقدار ۱۲۰۰۰ را برای مهلت زمانی تعیین کرده و بهینه‌سازی هزینه را درخواست می‌کند.

جدول (۱): پیکربندی منابع

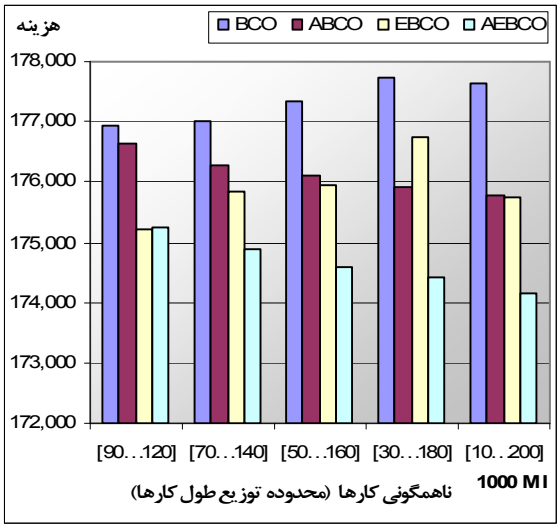
نام منبع	نرخ اجرا (MI/sec)	قیمت (G$/sec)	قیمت مفید (G$/1000MI)
R1	۱۰۰	۰,۵	۵
R2	۱۸۰	۱,۰	۵,۵۵
R3	۲۴۰	۱,۵	۶,۲۵
R4	۲۸۰	۲	۷,۱۴
R5	۳۰۰	۲,۵	۸,۳۳
R6	۴۰۰	۴	۱۰
R7	۵۰۰	۶	۱۲
R8	۶۰۰	۹	۱۵

در اولین آزمایش که نتایج آن در شکل (۲) نشان داده شده است، الگوریتم‌های پیشنهادی و الگوریتم BCO با یکدیگر مقایسه شده‌اند. همانطور که مشاهده می‌شود، الگوریتم AEBCO در مقایسه با سایر الگوریتم‌ها، از عملکرد بهتری برخوردار بوده و توانسته است هزینه را در مقایسه با الگوریتم BCO تا بیش از ۳۵۰۰ واحد (۲ درصد) کاهش دهد. گرچه الگوریتم‌های ABCO و EBCO کارایی تقریباً یکسانی دارند ولی در مقایسه با الگوریتم BCO از برتری قابل ملاحظه برخوردار هستند.



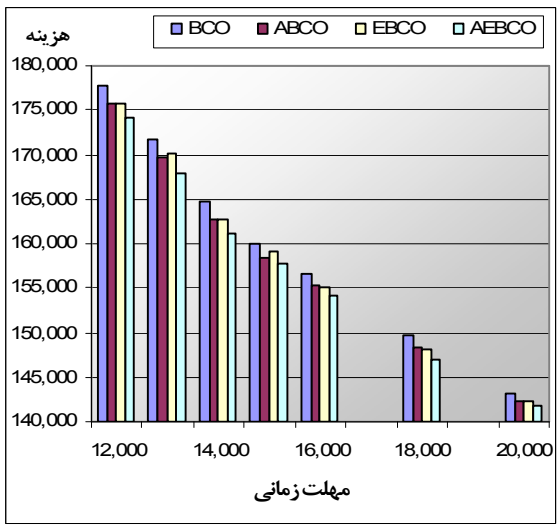
شکل (۲): مقایسه الگوریتم‌های بهینه‌سازی هزینه

در آزمایش بعدی، تأثیر ناهمگونی کارها را بر هزینه اجرای برنامه کاربر بررسی می‌کنیم. برای این منظور، محدوده توزیع طول کارها را تغییر می‌دهیم. محدوده‌های آزمایش شده و هزینه‌های بدست آمده در شکل (۳) نشان داده شده است. همانطور که مشاهده می‌شود با افزایش ناهمگونی درکارها، الگوریتم AEBCO منجر به کاهش هزینه اجرای برنامه کاربر گردیده است؛ در حالی که الگوریتم BCO با افزایش ناهمگونی درکارها دارای عملکرد مناسبی نمی‌باشد و برنامه‌های کاربران را با هزینه بالاتری اجرا می‌کند.



شکل (۳): مقایسه الگوریتم‌ها در ناهمگونی‌های مختلف

در آخرین آزمایش، تأثیر مهلت زمانی را بر روی هزینه اجرای برنامه کاربر بررسی می‌کنیم. نتایج این آزمایش در شکل (۴) نشان داده شده است. مشاهده می‌شود که با افزایش بودجه، زمان اجرای برنامه کاهش می‌یابد؛ زیرا الگوریتم‌های زمانبندی می‌توانند از منابع گران‌تر (که توانمندی بیشتری دارند) استفاده کنند. همچنین بهبود حاصل از الگوریتم‌های پیشنهادی در مقایسه با الگوریتم BCO در مهلت‌های بالا، کمتر شده است.



شکل (۴): نتایج الگوریتم‌ها در مهلت‌های مختلف

۵- نتیجه‌گیری

در این مقاله، سه الگوریتم مکاشفه‌ای جدید برای زمانبندی در گریدهای محاسباتی اقتصادی با مدل بازار کالا، به منظور مینیمم کردن هزینه در محدوده مهلت‌های تعیین شده پیشنهاد گردید. این الگوریتم‌ها با استفاده از جعبه‌ابزار GridSim شبیه‌سازی شده و کارایی آنها در شرایط مختلف، مورد بررسی قرار گرفت. نشان داده شد که الگوریتم‌های پیشنهادی در مقایسه با تنها الگوریتم گزارش شده، از کارایی بالاتری برخوردار بوده و برنامه کاربر را با هزینه کمتری اجرا می‌کنند. در بین الگوریتم‌های پیشنهادی، الگوریتم AEBCO در مقایسه با دو الگوریتم دیگر نتایج بهتری تولید می‌کند.

مراجع

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1999.

[2] I. Foster, C. Kesselman and S. Tuecke, *"The anatomy of the Grid: Enabling scalable virtual organizations"*, International Journal of Supercomputer Applications, 2001.

[3] M. Baker, R. Buyya and Domenico Laforenza, *"Grids and Grid technologies for wide-area distributed computing"*, The Journal of Concurrency and Computation: Practice and Experience, Vol 14, Issue 13-15, Nov. 2002.

[4] V. Berstis, *Fundamentals of Grid Computing*, IBM Redbooks, November 2002.

[5] R. Buyya, D. Abramson, and J. Giddy, *"A Case for Economy Grid Architecture for Service-Oriented Grid Computing"*, Proceedings of the 10th IEEE International Heterogeneous Computing Workshop, April 2001.

[6] R. Buyya, D. Abramson, and J. Giddy, *"An Economy Driven Resource Management Architecture for Global Computational Power Grids"*, Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, June 2000.

[7] R. Buyya, D. Abramson, and J. Giddy, *"Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid"*, The 4th International Conference on High Performance Computing in Asia-Pacific Region, May 2000.

[8] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, *"Economic Models for Resource Management and Scheduling in Grid Computing"*, The Journal of Concurrency and Computation: Practice and Experience, May 2002.

[9] R. Buyya, J. Giddy, D. Abramson, *"An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications"*, Proceedings of the 2nd International Workshop on Active Middleware Services, August 2000.

[10] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, Ph.D. Thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, April 2002.

[11] R. Buyya and M. Murshed, *"GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing"*, Journal of Concurrency and Computation: Practice and Experience, pp. 1-32, May 2002.

زیرنویس‌ها

[1] Computational Grids
[2] Heterogeneous
[3] Workstations
[4] Commodity Market Model

[5] Heterogeneous
[6] Deadline
[7] Heuristic
[8] Buyya Cost Optimization
[9] Parameter Sweep Applications
[10] Advanced Buyya Cost Optimization
[11] Extended Buyya Cost Optimization
[12] Advanced Extended Buyya Cost Optimization
[13] Million Instruction
[14] Homogeneous
[15] Time-shared
[16] Space-shared
[17] Overload
[18] Resource Discovery
[19] Resource Trading
[20] Admission Control
[21] Dispatching