بهبود زمانبندی وظایف مستقل در رایانش ابری با استفاده از الگوریتم رقابت استعماری

مینا نیکی‌بخت، دانشجوی کارشناسی ارشد، گروه مهندسی کامپیوتر، واحد تهران مرکزی، دانشگاه آزاد اسلامی، تهران، ایران
minanikibakht@gmail.com
زهره باطنی، عضو هیئت علمی، گروه مهندسی کامپیوتر، واحد تهران مرکزی، دانشگاه آزاد اسلامی، تهران، ایران
zbateni@hotmail.com

چکیده

اخیراً رایانش ابری یک حوزه تحقیقاتی پر رونق است که به عنوان یک واقعیت تجاری در حوزه فناوری اطلاعات مطرح شده است. رایانش ابری می تواند یک مدل کامل، مصرفی و تحویلی ارائه دهد که برای پرداخت به ازای مصرف، مبتنی بر اینترنت است. برای افزایش سرعت پاسخگویی و تضمین رضایت کاربران ابر، وظایف باید به شکل بهینه به ماشین‌های مجازی زمانبندی شوند. هدف از زمانبندی ابری ایجاد توان عملیات و محاسباتی بالا برای سیستم و تخصیص منابع مختلف به برنامه‌های مختلف کاربران می‌باشد. در این مقاله با استفاده از الگوریتم رقابت استعماری زمانبندی وظایف در محیط ابر در جهت کمینه‌سازی زمان تکمیل وظایف انجام شده است. در جهت بهبود عملکرد الگوریتم رقابت استعماری از عملگرهای تعویض و وارون‌سازی استفاده شده است تا تنوعی بیشتری در فضای پاسخ ایجاد شود و در صورت افتادن در نقاط بهینه محلی باعث خروج و یافتن پاسخ بهینه سراسری شوند. پس از مدل‌سازی مسئله با استفاده از الگوریتم رقابت استعماری و بهبود آن، زمانبندی وظایف به ازای 25 و30 وظیفه بر روی 4 پردازنده انجام می‌شود و نتایج آن با الگوریتم بهینه‌سازی ازدحام ذرات مقایسه می‌شود. نتایج حاکی از عملکرد بهینه الگوریتم رقابت استعماری با زمان تکمیل وظایف 196 و 245 به ترتیب براساس 25 و 30 وظیفه در برابر الگوریتم بهینه‌سازی ازدحام ذرات است.

کلیدواژه‌ها: رایانش ابری، زمانبندی وظایف، الگوریتم رقابت استعماری، الگوریتم بهینه‌سازی ازدحام ذرات.

1- مقدمه

سیستم‌های توزیع‌شده و تکنیک‌های پردازش موازی از جمله راه‌حل‌های استفاده‌ی بهتر و سریع‌تر از دنیای حجیم و پیچیده اطلاعات عصر حاضر می‌باشند. امروزه صدها رایانه و ابررایانه با ظرفیت‌ها و معماری‌های گوناگون در سراسر دنیا وجود دارند که در کاربردهای گوناگون علمی، نظامی، تجاری و نظایر آن از آن‌ها استفاده می‌شود و اکثراً لزوم به اشتراک‌گذاری اطلاعات در میان آنها امری ضروری است. یک سیستم توزیع-شده مجموعه‌ای از کامپیوترهای مستقل است که در نظر کاربران به‌صورت یک سیستم منسجم واحد به‌نظر می‌آید. محاسبات ابری الگویی از محاسبات توزیع‌شده، مرکب از تعداد زیاد منابع و درخواست‌ها با هدف به اشتراک‌گذاری منابع به‌صورت سرویس بر روی بستر اینترنت است(Wu, 2010). زمانبندی وظایف در کنار امنیت، قابلیت اطمینان و حفظ اعتماد مشترکین از چالش‌های مهم در زمینه رایانش ابری است که می‌تواند روی سایر مسائل نیز تأثیرگذار باشد. واضح است که تعداد وظایف در محیط ابر می‌تواند بسیار گسترده باشد و به همین علت ترتیب اجرای وظایف تأثیر مهمی بر کارایی سرویس‌دهنده ابر دارد. با توجه به زمانبربودن اجرای مسائل بهینه‌سازی در مقیاس بالا، یافتن راه‌حلی کارا در راستای رسیدن به یک زمانبندی بهینه، بسیار ضروری است. الگوریتم‌های زمانبندی نقش بسیار مهمی در محاسبات ابری دارند چرا که هدف زمانبندی این است تا زمان پاسخ را کاهش داده و بهره‌برداری از منبع را بهبود بخشد، برای این منظور الگوریتم‌های زمابندی مختلفی وجود دارد(Kumar, 2012). در این مقاله با استفاده از الگوریتم رقابت استعماری زمانبندی وظایف در محیط ابر انجام می‌شود. ادامه ساختار این مقاله به این صورت است که در بخش دوم مروری بر پیشینه تحقیقات انجام شده خواهیم داشت و در بخش سوم الگوریتم‌های تکاملی بیان می‌شوند. در بخش چهارم مدل پیشنهادی و ارزیابی نتایج استفاده از الگوریتم‌های تکاملی در جهت حل مسأله ارائه می‌شود و در نهایت در بخش آخر نتیجه‌گیری انجام می‌شود.

2- پیشینه تحقیق

یو و همکاران در سال 2006 به زمانبندی با محدودیت منابع برنامه‌های برروی ابر با استفاده از الگوریتم ژنتیک پرداختند. در محاسبات ابری کاربران از خدماتی که در بستر ابر فراهم شده است، استفاده می‌کنند درنتیجه کاربران خدمات را از ابر دریافت کرده و متعاقب آن هزینه پرداخت خواهند کرد. نحوه زمانبندی اجرای برنامه‌ها در جهت افزایش کیفیت سرویس انجام شده است. لذا در این پژوهش روشی برای زمانبندی در جهت محدودیت بودجه ارائه شده و زمان اجرای برنامه‌ها را در حالی‌که نیازمند یک بودجه معین برای کارهای محول‌شده است کمینه می‌کند. نسخه جدیدی از الگوریتم ژنتیک ارائه شد. نتایج شبیه‌سازی حاکی از بهبود در زمانبندی بر بستر آزمایش است(Yu, 2006). رحمان و همکاران در سال 2007 در مقاله خود تحت عنوان الگوریتم مسیر بحرانی پویا برای زمانبندی جریان کاری وظایف در ابر، یک روش DCP براساس الگوریتم زمانبندی جریان کاری که نگاشتی کارا از وظایف را توسط محاسبه کردن مسیر بحرانی در گراف وظایف جریان کاری تعیین می‌کند. عملکرد روش پیشنهادی با دیگر روش‌های ابتکاری و فراابتکاری براساس استراتژی‌های زمانبندی برای انواع مختلف و اندازه‌های مختلف جریان کاری حاکی از آن دارد که روش پیشنهادی در مقایسه با روش‌های دیگر از عملکرد بهتری برخوردار است(Rahman, 2007). چن و همکاران در سال 2009 با الگوریتم مورچگان به مساله زمانبندی وظایف در ابر با توجه به نیازهای افزایش کیفیت

سرویس پرداختند. باتوجه به اینکه در زمینه ابر الگوریتم‌های زمانبندی زیادی معرفی شده است و هرکدام از آنها تنها بخشی از چالش‌های به وجود آمده در این حوزه را برطرف می‌کنند و یا وظایف را می‌توانند مدیریت کنند. در این تحقیق یک روش زمانبندی براساس الگوریتم مورچگان برای جریان‌کاری با اندازه بزرگ و با پارامترهای زیاد در جهت افزایش کیفیت سرویس ارائه شده است. الگوریتم پیشنهادی کاربران را قادر می‌سازد تا اولویت‌های مختلف افزایش کیفیت سرویس را برآورده کرده و همین‌طور آستانه کمینه کیفیت سرویس را برای یافتن راه‌حلی که تمامی محدودیت‌های افزایش کیفیت سرویس را رفع کند و پارامترهای آن را که برای کاربران حائز اهمیت است را بهینه کند. در این مقاله 7 روش ابتکاری برای الگوریتم مورچگان طراحی شده است و یک برنامه تطبیق‌پذیر که به مورچگان اجازه می‌دهد روش‌های هوشمند را براساس مقادیر فرومون انتخاب کنند(Chen, 2009). پاندی و همکاران در سال 2010 به زمانبندی جریان کارها براساس الگوریتم بهینه‌سازی ازدحام ذرات در محاسبات ابر پرداختند. در این مقاله روشی برای بهینه‌کردن زمان اجرا و هزینه ناشی از انتقال داده بین منابع و هزینه اجرا ارائه شد. روش پیشنهادی برنامه جریان کاری توسط تغییر محاسبات و هزینه‌های برقراری ارتباط آزمایش شد و مقایسه‌ای براساس صرفه‌جویی در هزینه با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات انجام شد. نتایج حاکی از کاهش هزینه با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات تا 3 برابر نسبت به الگوریتم BRS دارد(Pandey, 2010). مائو و همکاران در سال 2011 به خود مقایسه سنجی برای کمینه‌سازی هزینه و برطرف کردن موعد تحویل وظایف در جریان‌های کاری در ابر پرداختند. در این پژوهش روشی براساس عناصر محاسباتی بنیادی در ماشین‌های مجازی با هزینه‌ها و اندازه-های متفاوت ارائه شد. کاربران با استفاده از این روش می‌توانند نیازمندهای عملیاتی خود را توسط تعیین موعد تحویل برای کارها برآورده کنند. هدف در اینجا اطمینان از این است که تمامی کارها در زمان مقرر و با کمترین هزینه مالی انجام شوند. برای رسیدن به این اهداف تخصیص وظایف برروی ماشین‌های مجازی و زمانبندی کارها به‌طور پویا انجام شد. این روش با 4 روش دیگر در زمینه محاسبات ابری مورد مقایسه قرار گرفته که نشان از صرفه‌جویی در هزینه از 9.8% تا 40.4% در مقایسه با دیگر روش‌ها دارد(Mao, 2011). مالوسکی و همکاران در سال 2012 به محدودیت موعد تحویل و هزینه ایجادشده برروی زیرساخت به عنوان سرویس پرداختند. در این مقاله الگوریتم‌های موردنظر به صورت پویا و ایستا برای زمانبندی وظایف و تخصیص منابع مورد ارزیابی واقع شدند. ارزیابی‌های انجام‌شده براساس شبیه‌سازی مجموعه‌ای از جریان‌های کاری با انواع مختلفی از بودجه و موعد تحویل است. عدم قطعیت در زمان اجرا و تأخیر و نقص در انجام کارها را در نظر می‌گیرد. نتایج حاکی از آن است که عملکرد یک الگوریتم توسط توانایی آن برای تصمیم‌گیری در جریان‌های کاری برای اجرا یا رد درخواست برای اجراشدن دارد. نتایج نشان می‌دهد که رویه‌های پذیرفته‌شده براساس ساختار جریان کار و تخمین زمان اجرای وظایف می‌تواند به طور موثری کیفیت راه‌حل‌ها را بهبود بخشد(Malawski, 2012). ابریشمی و همکاران در سال 2013 به الگوریتم‌های زمانبندی جریان کاری با محدودیت موعد تحویل به عنوان زیرساخت به عنوان سرویس در ابر پرداختند. در این مقاله الگوریتم PCP در محیط ابر مورد آزمایش قرار گرفت. همچنین دو الگوریتم زمانبندی جریان‌کاری ارائه شد. الگوریتم اول با نام IC-PCP یک مرحله‌ای است و الگوریتم دوم IC-PCPD2 نام دارد. هردوی این الگوریتم‌ها دارای پیچیدگی زمانی درجه 2 هستند که آنها را گزینه‌ای مناسب برای زمانبندی جریان‌های کاری می‌کند. نتایج نشان می‌دهد هردوی الگوریتم‌ها عملکرد قابل قبولی دارند اما الگوریتم IC-PCP در اکثر موارد عملکرد

بهتری از خود نشان می‌دهد(Abrishami, 2013). رودریگز و همکاران در سال 2014 به ایجاد منابع براساس موعد تحویل و الگوریتم زمانبندی برروی جریان‌های کاری در ابر پرداختند. روش پیشنهادی در پی فراهم‌کردن منابع موردنیاز در بستر ابر و ایجاد استراتژی زمانبندی برای جریان‌های کاری در زیرساخت به عنوان سرویس است. الگوریتم پیشنهادی PSO بود و هدف آن کمینه‌سازی هزینه اجرای کلی جریان کاری با محدودیت زمان تحویل است. شبیه‌سازی‌های انجام شده با جریان‌های کاری با اندازه‌های مختلف نشان از بهبود این روش نسبت به روش‌های دیگر را نشان می‌دهد(Rodriguez, 2014).

3- الگوریتم‌های تکاملی

در این بخش الگوریتم‌های تکاملی مرور می‌شوند. در این بخش الگوریتم‌های رقابت استعماری و بهینه‌سازی ازدحام ذرات بررسی می‌شوند.

3-1- الگوریتم رقابت استعماری

الگوریتم رقابت استعماری برای اولین در سال 2007 معرفی شد. این الگوریتم مبتنی بر هوش‌جمعی بوده و شامل مراحل زیر است(Atashpaz-Gargari, 2007).

مراحل الگوریتم رقابت استعماری:

مرحله آماده‌سازی اولیه: جمعیتی از کشورهای اولیه ایجاد و در فضای جستجو به‌صورت تصادفی پخش می شوند و مورد ارزیابی قرار می‌گیرند. کشورها براساس مقدار هدف مرتب شده و بهترین آن‌ها به عنوان امپراطور در نظر گرفته‌شده و بقیه به عنوان مستعمرات در نظر گرفته می‌شوند. امپراطورها براساس شایستگی که دارند می‌توانند مستعمرات را به عنوان کلونی به امپراطوری خود ملحق کنند(Atashpaz-Gargari, 2007).

مرحله تکرار: در هر امپراطوری سیاست جذب (همگون‌سازی یا شبیه‌سازی) اعمال می‌شود.کشور مستعمره، به اندازه x واحد در جهت خط واصل مستعمره به استعمارگر، حرکت کرده و به موقعیت جدید، کشانده می‌شود. در رابطه (1) فاصله میان استعمارگر و مستعمره با d نشان داده شده است. x نیز عددی تصادفی با توزیع یکنواخت (و یا هر توزیع مناسب دیگر) است. یعنی برای x داریم:

$$x \epsilon U(0, \beta \times d) \quad (1)$$

در هر امپراطوری انقلاب بر روی امپراطور و مستعمرات آن با احتمال انقلاب انجام می‌شود. در هر امپراطوری رقابت درون امپراطوری صورت گرفته و در صورتی که مستعمره‌ای یافت شود که از امپراطورش بهتر باشد جایگزین آن می‌شود. برای هر امپراطوری شاخص عملکرد براساس امپراطور و ضریبی از میانگین مستعمراتش محاسبه می‌شود. بدین ترتیب هزینه کل یک امپراطوری براساس رابطه (2) محاسبه می‌شود:

$$T.C_n = Cost(imperialist_n) + \xi mean\{Cost(colonies\ of\ empire_n)\} \quad (2)$$

رقابت بین امپراطورها صورت می‌گیرد و از امپراطور ضعیف مستعمره‌ای حذف و به امپراطوری قوی‌تر تخصیص می‌یابد (البته امپراطور قوی‌تر شانس بیش‌تری برای دریافت مستعمره دارد). اگر امپراطور ضعیف فاقد مستعمره باشد خود به عنوان مستعمره به امپراطور قوی‌تر تخصیص یافته و امپراطوریش سقوط می‌کند. به مرحله تکرار رفته و تا برآورده شدن شرایط خاتمه فرآیند تکرار می‌شود(Atashpaz-Gargari, 2007).

2-3- الگوریتم بهینه‌سازی ازدحام ذرات

این الگوریتم از حرکت دسته جمعی پرندگانی که به دنبال غذا می‌باشند الهام گرفته شده است. گروهی از پرندگان در فضایی به صورت تصادفی به دنبال غذا می‌گردند تنها یک تکه غذا در فضای مورد بحث وجود دارد. هیچ‌یک از پرندگان محل غذا را نمی‌دانند. یکی از بهترین استراتژی‌ها می‌تواند دنبال‌کردن پرنده‌ای باشد که کم‌ترین فاصله را تا غذا داشته باشد. این استراتژی در واقع جان‌مایه الگوریتم است. فضای مسأله مورد جستجو در الگوریتم PSO معادل فضای مورد جستجو در الگوی حرکت پرندگان است. هر راه‌حل که به آن یک ذره گفته می‌شود در الگوریتم PSO معادل یک پرنده است و تعداد ذرات(راه‌حل‌ها) معادل تعداد پرندگان است. هر ذره یک مقدار شایستگی دارد که توسط یک تابع شایستگی محاسبه می‌شود و هر چه ذره در فضای جستجو به هدف یعنی غذا در مدل حرکت پرندگان نزدیک‌تر باشد، شایستگی بیش‌تری دارد. هم‌چنین هر ذره دارای یک جابه‌جایی است که هدایت حرکت ذره را بر عهده دارد و به کمک آن مکان بعدی ذره مشخص می‌شود. هر ذره با دنبال‌کردن ذرات بهینه در حالت فعلی، به حرکت خود در فضای جستجو ادامه می‌دهد تا این‌که نهایتاً به جواب بهینه دست یابد(Shi, 2010).

مراحل اصلی الگوریتم PSO به شرح زیر است:

- مرحله آماده‌سازی اولیه
- مرحله تکرار

مرحله آماده‌سازی اولیه

در این مرحله جمعیتی از ذرات ایجاد می‌شوند و برای هر ذره یک بردار مکان و یک بردار جابه‌جایی در نظر گرفته می‌شود. در ادامه با استفاده از تابع هدف مقدار هدف هر ذره محاسبه و بهترین ذره از لحاظ شایستگی به عنوان بهترین پاسخ سراسری نگهداری می‌شود(Shi, 2010).

مرحله تکرار

در این مرحله عملیات زیر تا حصول یکی از شرایط رسیدن به جواب بهینه یا میزان خطا به حدنصاب برسد و یا این‌که تعداد تکرار مشخص به اتمام برسد، تکرار می‌شود. عملیاتی که در این مرحله انجام می‌شوند عبارتند از:

در هر مرحله از تکرار الگوریتم بردار جابجایی V_i و مکان X_i در تکرار $(t+1)$ ام از روابط (3) و (4) تعیین می‌شوند:

$$V_i(t+1) = \omega V_i(t) + C_1 \times rand_1(pbest_i(t) - X_i(t)) + C_2 \times rand_2(gbest_i(t) - X_i(t)) \quad (3)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (4)$$

- $X_i(t)$: مکان فعلی ذره iام در تکرار tام
- $V_i(t)$: بردار جابه‌جایی فعلی ذره iام در تکرار tام
- $pbest_i(t)$: بهترین مکان تجربه شده توسط ذره iام تا تکرار tام
- $gbest_i(t)$: بهترین مکان به‌دست آمده میان همسایگان ذره iام تا تکرار tام
- $rand_1$ و $rand_2$: عدد تصادفی بین 0 و 1 تولید می‌کنند(Shi, 2010).

4- روش پیشنهادی

برای شبیه‌سازی مسأله زمانبندی وظایف در محیط ابر مفروضات زیر در فرآیند شبیه‌سازی در نظر گرفته شده است:

I: تعداد وظایف در محیط ابر

J: تعداد پردازنده‌ها برای پردازش وظایف

P_{ij}: زمان پردازش وظیفه iام بر روی پردازنده jام

$S_{i1,i2,j}$: زمان آماده‌سازی وظیفه i_2 بعداز اتمام اجرای وظیفه i_1 بر روی پردازنده jام

4-1- قیدها در فرایند شبیه‌سازی

1) همه وظایف بایستی اجرا شوند.
2) هر وظیفه فقط و فقط توسط یک پردازنده اجرا می‌شود.
3) هر وظیفه فقط یک‌بار اجرا می‌شود.
4) زمان شروع وظیفه بعدی توسط یک پردازنده برابر است با زمان اتمام وظیفه قبلی به علاوه زمان آماده‌سازی وظیفه بعدی(طبق رابطه (5))

Start Time(Next Task)=Finish Time(Previous Task) + Setup Time(Next Task) (5)

5) زمان اتمام وظیفه فعلی برابر است با زمان شروع وظیفه فعلی به علاوه زمان اجرای وظیفه فعلی(طبق رابطه (6))

Finish Time(Current Task)=Start Time(Current Task)+Process Time(Current Task) (6)

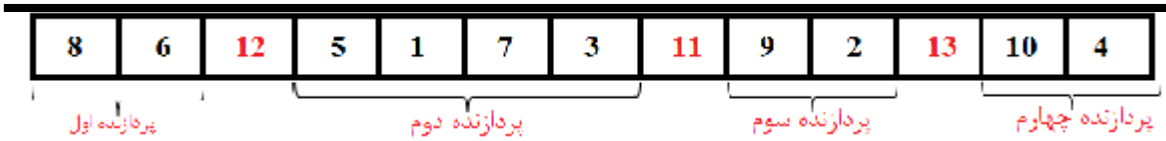
4-2- تعریف متغیر تصمیم در شبیه‌سازی

برای شبیه‌سازی متغیرهای تصمیم در مساله از کدگذاری گسسته استفاده شده است و ساختار متغیر تصمیم شامل I+J-1 بعد است. با فرض اینکه تعداد وظایف برابر 10 و تعداد پردازنده‌ها برابر 4 باشد شکل (1) نحوه کدگذاری متغیر تصمیم را نشان می‌دهد.

8	6	12	5	1	7	3	11	9	2	13	10	4

شکل 1: شکل متغیر تصمیم

در شکل (1) اعداد سیاه رنگ نشان‌دهنده شماره وظایف و اعداد قرمز رنگ نشان‌دهنده جداکننده‌ها است یعنی وظایفی که بایستی توسط هر پردازنده اجرا شوند را نشان می‌دهد. در این روش از ابتدای لیست شروع کرده و تا رسیدن به هر جداکننده وظایفی که مشاهده شوند به ترتیب به پردازنده‌ها تخصیص می‌یابند. شکل (2) نحوه تخصیص وظایف به پردازنده‌ها را نشان می‌دهد.



شکل 2: نحوه تخصیص وظایف به پردازنده‌ها

4-3- تعریف تابع هدف

در مسأله زمان‌بندی وظایف در محیط ابر هدف زمان‌بندی و تخصیص مناسب وظایف به پردازنده‌ها در جهت کمینه‌سازی زمان تکمیل اجرای وظایف است. بنابراین تابع هدف به صورت رابطه (7) و به صورت کمینه‌سازی زمان تکمیل وظایف تعریف می‌شود.

$$f(x) = Min\ (CpuCompletionTime_i) \qquad \forall\ i \qquad (7)$$

به عبارت دیگر در این شبیه‌سازی برای هر راه‌حل، زمان تکمیل وظایف هر پردازنده مشخص می‌شود سپس پردازنده‌ای که دارای بیشترین زمان تکمیل وظایف باشد به عنوان مقدار تابع هدف در نظر گرفته می‌شود.

4-4- نتایج استفاده از الگوریتم‌های تکاملی در زمانبندی

در این بخش قصد داریم به بررسی نتایج شبیه‌سازی زمان‌بندی وظایف در محیط ابر با استفاده از الگوریتم-های تکاملی بپردازیم. الگوریتم‌های رقابت استعماری و بهینه‌سازی ازدحام ذرات از جمله الگوریتم‌های هستند که در جهت زمانبندی وظایف در محیط ابر استفاده شده‌اند. در کلیه مراحل شبیه‌سازی تعداد پردازنده‌ها برابر 4 و تعداد وظایف 25 و 30 در نظر گرفته شده است.

4-4-1- بهبود الگوریتم‌های تکاملی با استفاده از عملگرها

به دلیل اینکه مسأله زمانبندی وظایف به صورت یک مسأله جایگشتی تعریف شده است بنابراین در این مقاله قصد داریم با اعمال عملگرهای تعویض و وارون‌سازی باعث تنوع در فضای پاسخ الگوریتم‌ها شویم. اعمال عملگرها باعث اجتناب از همگرایی به بهینه محلی و ایجاد تنوع و گوناگونی در فضای پاسخ می‌شود.

4-4-1-1- روش تعویض

در این روش دو بعد یا دو بلوک از پاسخ به‌طور تصادفی انتخاب شده و موقعیت آن‌ها تعویض می‌شود. شکل (3) مثالی از روش تعویض را نمایش می‌دهد.

شکل 3: روش تعویض

4-4-1-2- روش وارون‌سازی

دو بعد از پاسخ به‌طور تصادفی انتخاب شده و ترتیب آنها در بین این دو بعد معکوس می‌شود. شکل (4) مثالی از روش وارون‌سازی را نمایش می‌دهد.

شکل 4: روش وارون‌سازی

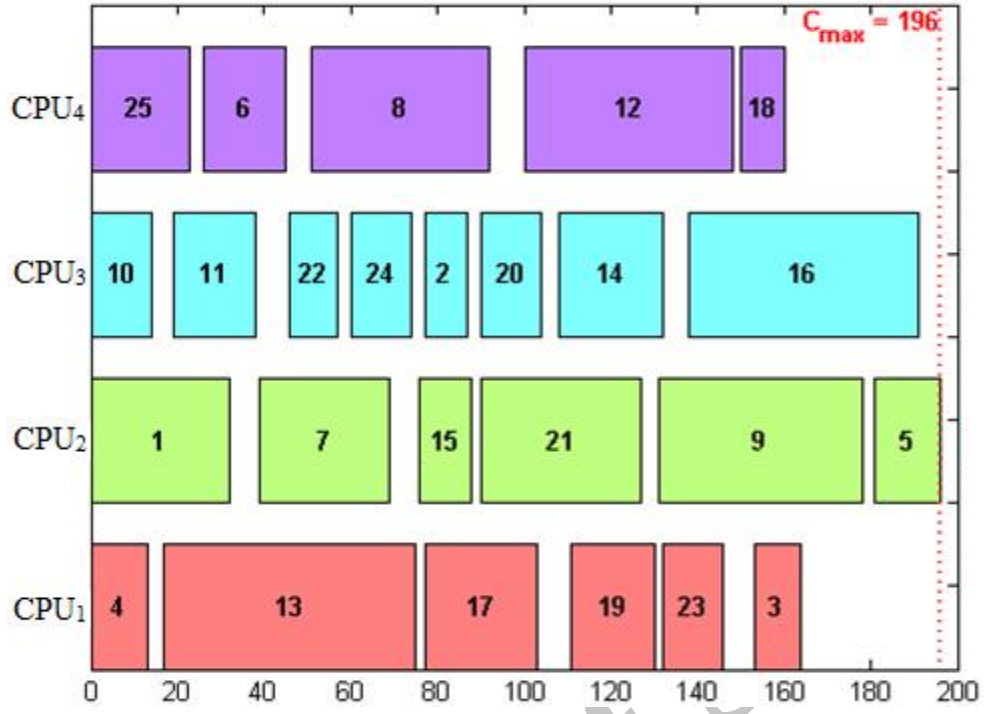
4-5- نتایج استفاده از الگوریتم رقابت استعماری

در الگوریتم رقابت استعماری اندازه جمعیت برابر 50، حداکثر تعداد نسل‌ها برابر 50، ضریب جذب برابر 0/4 و تعداد امپراطورها برابر 10 در نظر گرفته شده است. جدول (1) موارد مربوط به تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل، اتلاف وقت و درصد کارایی را در هر پردازنده در حالت 25 وظیفه را نشان می‌دهد.

جدول 1: نتایج زمان‌بندی وظایف در الگوریتم رقابت استعماری در حالت 25 وظیفه

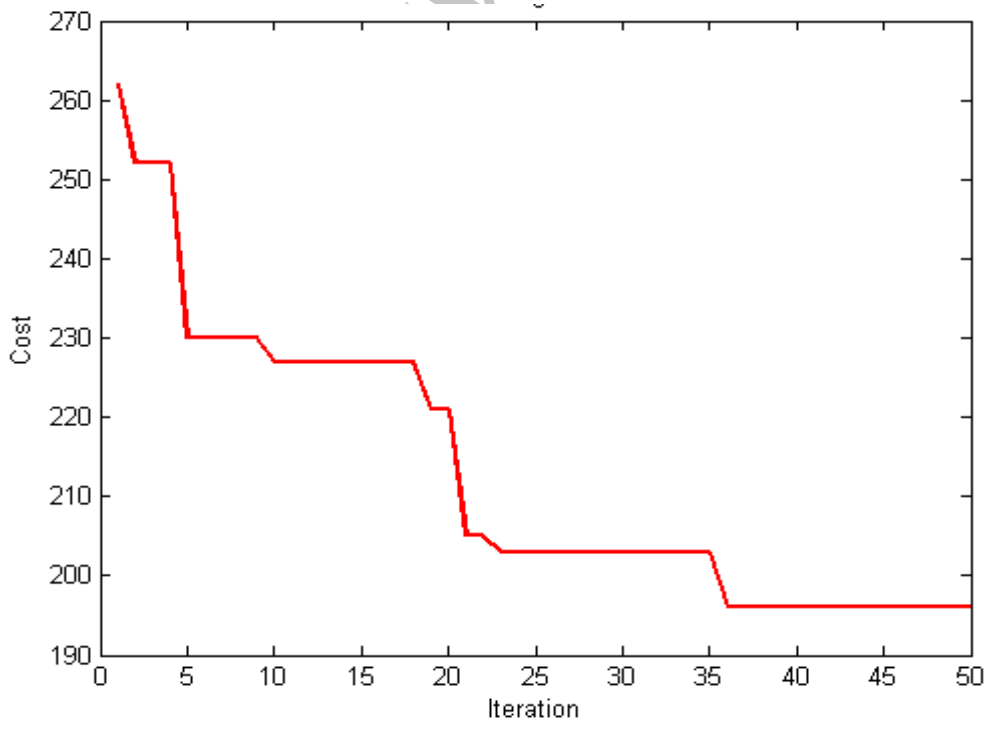
	پردازنده اول	پردازنده دوم	پردازنده سوم	پردازنده چهارم
زمان تکمیل وظایف(میلی‌ثانیه)	6	6	8	5
تعداد وظایف	105/16	115	87/125	93/6
متوسط تعداد وظایف	81/66	86/16	67/25	65/4
متوسط زمان پاسخ(میلی‌ثانیه)	164	196	191	160
متوسط زمان انتظار(میلی‌ثانیه)	32	0	5	36
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	83/67	100	97/44	81/63

شکل (5) زمانبندی و تخصیص وظایف به پردازنده‌ها در حالت 25 وظیفه با استفاده از الگوریتم رقابت استعماری را نشان می‌دهد. محور افقی زمان و محور عمودی تعداد وظایف تخصیص‌یافته به هر پردازنده را نشان می‌دهد.



شکل 5: زمانبندی وظایف با استفاده از الگوریتم رقابت استعماری در حالت 25 وظیفه

شکل (6) نمودار همگرایی تابع هدف الگوریتم رقابت استعماری در نسل‌های مختلف در حالتی که تعداد وظایف برابر 25 است را نشان می‌دهد.



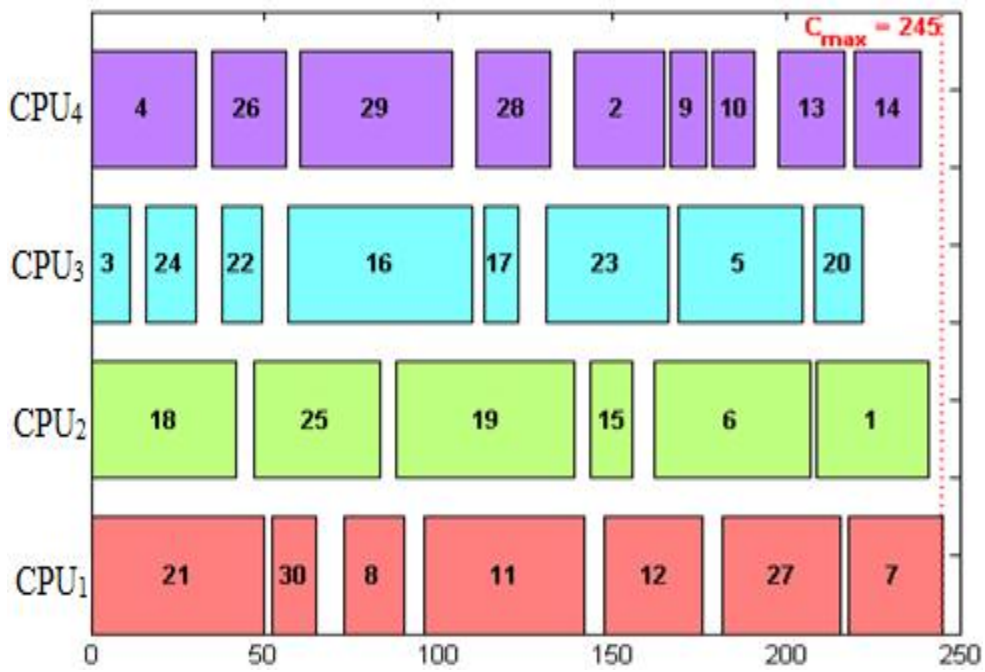
شکل 6: نمودار همگرایی تابع هدف در الگوریتم رقابت استعماری در حالت 25 وظیفه

جدول (2) موارد مربوط به تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل، اتلاف وقت و درصد کارایی را در هر پردازنده در حالت 30 وظیفه را نشان می‌دهد.

جدول 2: نتایج زمان‌بندی وظایف در الگوریتم رقابت استعماری در حالت 30 وظیفه

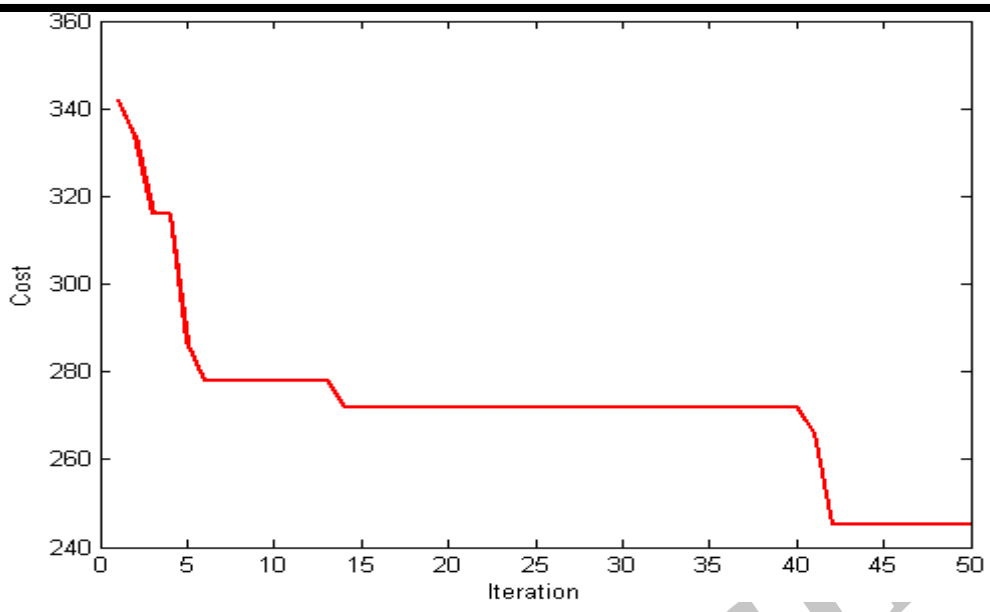
	پردازنده اول	پردازنده دوم	پردازنده سوم	پردازنده چهارم
زمان تکمیل وظایف(میلی‌ثانیه)	7	6	8	9
تعداد وظایف	140/57	144/66	114/5	145/66
متوسط تعداد وظایف	109/85	108/33	91/5	123/22
متوسط زمان پاسخ(میلی‌ثانیه)	245	241	222	239
متوسط زمان انتظار(میلی‌ثانیه)	0	4	23	6
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	100	98/36	90/61	97/55

شکل (7) زمانبندی و تخصیص وظایف به پردازنده‌ها در حالت 30 وظیفه با استفاده از الگوریتم رقابت استعماری را نشان می‌دهد. محور افقی زمان و محور عمودی تعداد وظایف تخصیص‌یافته به هر پردازنده را نشان می‌دهد.



شکل 7: زمانبندی وظایف با استفاده از الگوریتم رقابت استعماری در حالت 30 وظیفه

شکل (8) نمودار همگرایی تابع هدف الگوریتم رقابت استعماری در نسل‌های مختلف در حالتی که تعداد وظایف برابر 30 است را نشان می‌دهد.



شکل 8: نمودار همگرایی تابع هدف در الگوریتم رقابت استعماری در حالت 30 وظیفه

جدول (3) زمان تکمیل وظایف، تعداد وظایف، متوسط تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل وظایف، متوسط زمان اتلاف وقت و متوسط کارایی را به‌ازای 25 و 30 وظیفه برای 4 پردازنده در الگوریتم رقابت استعماری نشان می‌دهد.

جدول 3: نتایج استفاده از الگوریتم رقابت استعماری با تعداد 4 پردازنده

زمان تکمیل وظایف(میلی‌ثانیه)	245	196
تعداد وظایف	30	25
متوسط تعداد وظایف	7/5	6/25
متوسط زمان پاسخ(میلی‌ثانیه)	108/22	100/22
متوسط زمان انتظار(میلی‌ثانیه)	92/61	75/12
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	236/75	177/75
متوسط زمان اتلاف(میلی‌ثانیه)	8/25	5/25
متوسط کارایی(%)	96/63	97/4

6-4- نتایج استفاده از الگوریتم بهینه‌سازی ازدحام ذرات

در الگوریتم بهینه‌سازی ازدحام ذرات اندازه جمعیت برابر 50، حداکثر تعداد نسل‌ها برابر 50، وزن اینرسی برابر 0/7، ضریب یادگیری شخصی برابر 1/4962 و ضریب یادگیری سراسری برابر 1/2 در نظر گرفته شده است. جدول (4) موارد مربوط به تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل، اتلاف وقت و درصد کارایی را در هر پردازنده در حالت 25 وظیفه را نشان می‌دهد.

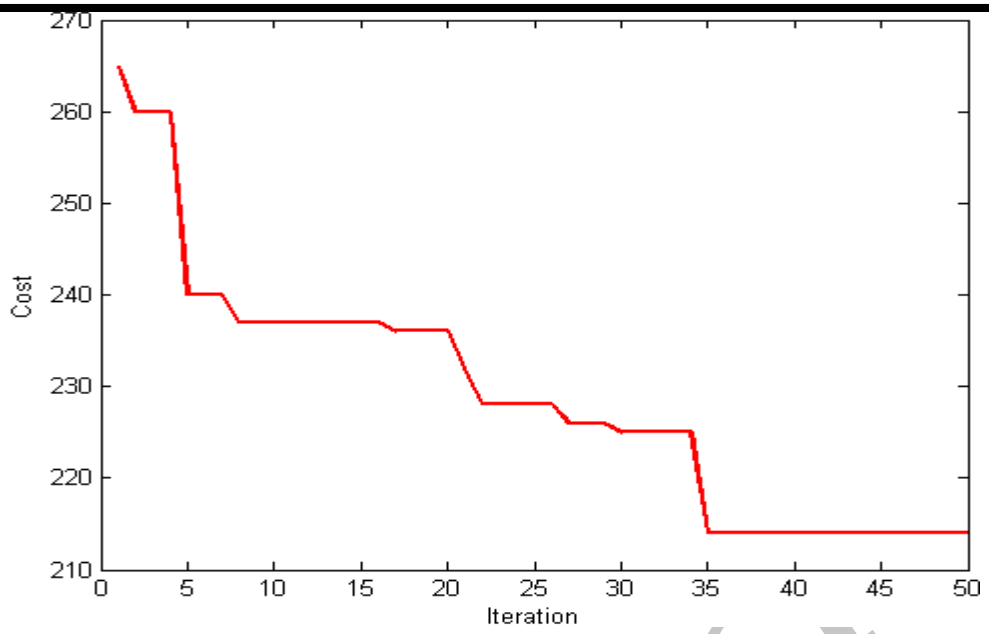
جدول 4: نتایج زمان‌بندی وظایف در الگوریتم بهینه‌سازی ازدحام ذرات در حالت 25 وظیفه

	پردازنده اول	پردازنده دوم	پردازنده سوم	پردازنده چهارم
زمان تکمیل وظایف(میلی‌ثانیه)	5	6	8	6
تعداد وظایف	102/8	115/83	127/75	86/83
متوسط تعداد وظایف	64	85/5	104/87	58/66
متوسط زمان پاسخ(میلی‌ثانیه)	213	212	214	193
متوسط زمان انتظار(میلی‌ثانیه)	1	2	0	21
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	99/53	99/06	100	90/18

شکل (9) زمانبندی و تخصیص وظایف به پردازنده‌ها در حالت 25 وظیفه با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات را نشان می‌دهد. محور افقی زمان و محور عمودی تعداد وظایف تخصیص‌یافته به هر پردازنده را نشان می‌دهد.

شکل 9: زمانبندی وظایف با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات در حالت 25 وظیفه

شکل (10) نمودار همگرایی تابع هدف الگوریتم بهینه‌سازی ازدحام ذرات در نسل‌های مختلف در حالتی که تعداد وظایف برابر 25 است را نشان می‌دهد.



شکل 10: نمودار همگرایی تابع هدف در الگوریتم بهینه‌سازی ازدحام ذرات در حالت 25 وظیفه

جدول (5) موارد مربوط به تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل، اتلاف وقت و درصد کارایی را در هر پردازنده در حالت 30 وظیفه را نشان می‌دهد.

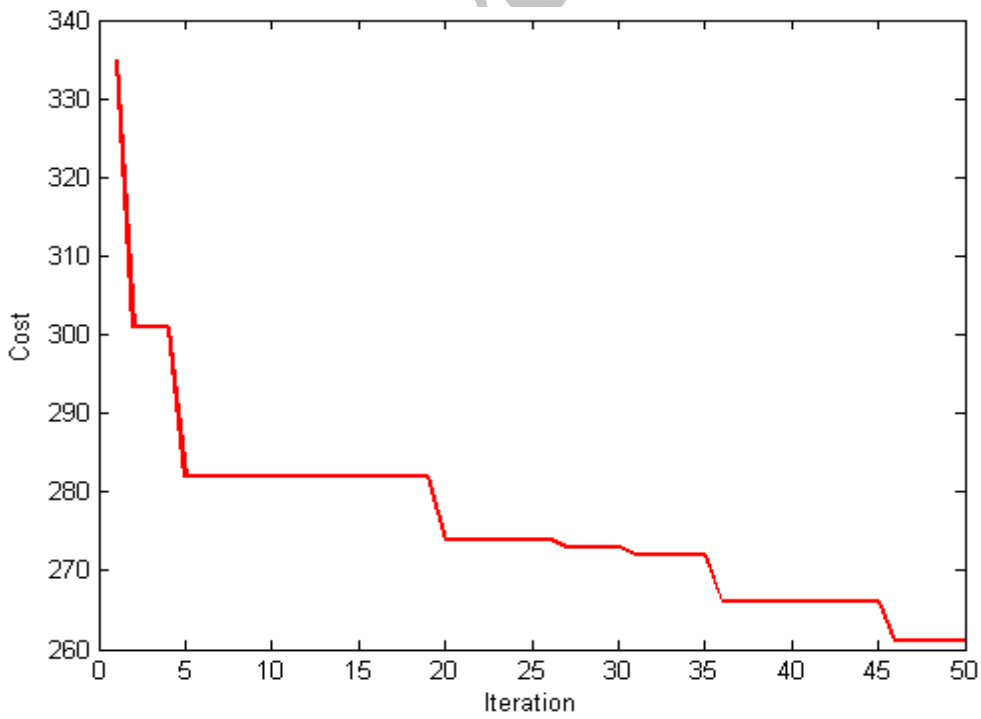
جدول 5: نتایج زمان‌بندی وظایف در الگوریتم بهینه‌سازی ازدحام ذرات در حالت 30 وظیفه

	پردازنده اول	پردازنده دوم	پردازنده سوم	پردازنده چهارم
زمان تکمیل وظایف(میلی‌ثانیه)	7	7	6	10
تعداد وظایف	128/14	156/85	121/33	135
متوسط تعداد وظایف	98/42	124/74	87/33	112/3
متوسط زمان پاسخ(میلی‌ثانیه)	226	249	219	261
متوسط زمان انتظار(میلی‌ثانیه)	35	12	42	0
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	86/59	95/4	83/9	100

شکل (11) زمانبندی و تخصیص وظایف به پردازنده‌ها در حالت 30 وظیفه با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات را نشان می‌دهد. محور افقی زمان و محور عمودی تعداد وظایف تخصیص‌یافته به هر پردازنده را نشان می‌دهد.

شکل 11: زمانبندی وظایف با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات در حالت 30 وظیفه

شکل (12) نمودار همگرایی تابع هدف الگوریتم بهینه‌سازی ازدحام ذرات در نسل‌های مختلف در حالتی که تعداد وظایف برابر 30 است را نشان می‌دهد.



شکل 12: نمودار همگرایی تابع هدف در الگوریتم بهینه‌سازی ازدحام ذرات در حالت 30 وظیفه

جدول (6) زمان تکمیل وظایف، تعداد وظایف، متوسط تعداد وظایف، متوسط زمان پاسخ، متوسط زمان انتظار، متوسط زمان تکمیل وظایف، متوسط زمان اتلاف وقت و متوسط کارایی را به‌ازای 25 و 30 وظیفه برای 4 پردازنده در الگوریتم بهینه‌سازی ازدحام ذرات نشان می‌دهد.

جدول 6: نتایج استفاده از الگوریتم بهینه‌سازی ازدحام ذرات با تعداد 4 پردازنده

زمان تکمیل وظایف(میلی‌ثانیه)	261	214
تعداد وظایف	30	25
متوسط تعداد وظایف	7/5	6/25
متوسط زمان پاسخ(میلی‌ثانیه)	135/33	108/3
متوسط زمان انتظار(میلی‌ثانیه)	105/69	78/26
متوسط زمان تکمیل وظایف(میلی‌ثانیه)	241/25	208
متوسط زمان اتلاف(میلی‌ثانیه)	22/25	18/25
متوسط کارایی(%)	91/47	90/68

5- نتیجه‌گیری

این مقاله مبتنی بر زمانبندی وظایف در محیط ابر با استفاده از الگوریتم‌های تکاملی است. در بخش دوم ادبیات تحقیق مرور شد و در بخش سوم الگوریتم‌های رقابت استعماری و بهینه‌سازی ازدحام ذرات بررسی شدند. در بخش چهارم به شبیه‌سازی و پیاده‌سازی مسأله پرداخته شد و نتایج کمی و کیفی استفاده از الگوریتم‌های رقابت استعماری و بهینه‌سازی ازدحام ذرات در زمانبندی وظایف در محیط رایانش ابری نمایش داده شد. با توجه به جدول (3) الگوریتم رقابت استعماری با زمان تکمیل وظایف 196 و 245، متوسط زمان پاسخ 100.22 و 108.22، متوسط زمان اتلاف 18.25 و 8.25 و متوسط درصدکارایی 90.68 و 96.63 به ترتیب به‌ازای 25 و 30 وظیفه زمانبندی وظایف در محیط ابر را انجام می‌دهد در حالی‌که با توجه به جدول (6) الگوریتم بهینه‌سازی ازدحام ذرات با زمان تکمیل وظایف 214 و 261، متوسط زمان پاسخ 108.3 و 135.33، متوسط زمان اتلاف 6 و 22.25 و متوسط درصدکارایی 97.19 و 91.47 به ترتیب به‌ازای 25 و 30 وظیفه زمانبندی وظایف در محیط ابر را انجام می‌دهد. بنابراین می‌توان نتیجه گرفت الگوریتم رقابت استعماری عملکرد بهتری در مقایسه با الگوریتم بهینه‌سازی ازدحام ذرات دارد و به عنوان مدل برتر انتخاب می‌شود.

مراجع

[1]. Abrishami, Saeid, Mahmoud Naghibzadeh, and Dick HJ Epema. (2013), ***Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds.*** Future Generation Computer Systems 29.1 (2013): 158-169.

[2]. Atashpaz-Gargari, E., & Lucas, C. (2007). **Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition**. Paper presented at the Evolutionary Computation, 2007. CEC 2007. IEEE Congress on.

[3]. Chen, Wei-Neng, and Jun Zhang. (2009), ***An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements.***, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 39.1: 29-43.

[4]. Kumar, P. and A. Verma, (2012), ***Independent task scheduling in cloud computing by improved genetic algorithm***. International Journal of Advanced Research in Computer Science and Software Engineering.

[5]. Malawski, Maciej, et al. (2012), ***Cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds.*** Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press.

[6]. Mao, Ming, and Marty Humphrey. (2011), ***Auto-scaling to minimize cost and meet application deadlines in cloud workflows.*** Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM.

[7]. Pandey, Suraj, et al. (2010), ***A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments.*** Advanced information networking and applications (AINA), 2010 24th IEEE international conference on. IEEE.

[8]. Rahman, Mustafizur, Srikumar Venugopal, and Rajkumar Buyya. (2007), ***A dynamic critical path algorithm for scheduling scientific workflow applications on global grids.*** e-Science and Grid Computing, IEEE International Conference on. IEEE.

[9]. Rodriguez, Maria Alejandra, and Rajkumar Buyya. (2014), ***Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds.*** Cloud Computing, IEEE Transactions on 2.2 : 222-235.

[10]. Shi,Y. Eberhart, R.C. (2010), ***Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization*** ,In: Proceedings of IEEE International Congress on Evolutionary Computation, pp.84-82.

[11]. Wu, Zhangjun, et al. (2010), ***A revised discrete particle swarm optimization for cloud workflow scheduling.*** Computational Intelligence and Security (CIS), 2010 International Conference on. IEEE.

[12]. Yu, Jia, and Rajkumar Buyya. (2006), ***A budget constrained scheduling of workflow applications on utility grids using genetic algorithms.*** Workflows in Support of Large-Scale Science, 2006. WORKS'06. Workshop on. IEEE.