# الگوریتم ژنتیک ترکیبی جهت بهینه سازی زمان بندی جریان کاری

# درسیستم های محاسباتی ناهمگن

**امیرایرانمنش[1]* ، حمیدرضا ناجی[2] .**

**۱-دانشجوی کارشناسی ارشد دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته کرمان**

**2-عضو هیات علمی دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته کرمان**

**چکیده**

زمان بندی جریان کاری قسمت کلیدی پردازش بهینه جریان کاری می باشد . زمان بندی جریان کار مساله NP-hard مشهوری است و پیچیدگی های زیادی در محیط های محاسباتی ناهمگن دارد . افزایش روز افزون پیچیدگی های جریان کاری محققان را وادار نموده است که روش های ترکیبی جدیدی برای حل مساله زمان بندی جریان کاری استفاده نمایند . کارآیی الگوریتم های ژنتیک با استفاده از تغییر عملگرهای ژنتیک و استفاده از هیوریستیک کارا می تواند افزایش یابد . از این ویژگی ها در الگوریتم ژنتیک ترکیبی[2] معرفی شده در این مقاله استفاده شده است . راه حل بدست آمده از هیوریستیک به عنوان جمعیت ورودی اولیه در الگوریتم استفاده شده است . راه حل هیوریستیک جهت اصلی را برای رسیدن الگوریتم اصلی به جواب بهینه برای ما مشخص می کند.

دو عملگر ژنتیک تغییر یافته جستجو را به دقت انجام می دهد و در کمترین زمان ممکن الگوریتم را به بهترین راه حل همگرا می سازد . قدرت الگوریتم معرفی شده در بهینه سازی ، هدف اصلی زمان بندی ( کمینه نمودن مقدار makespan) اثبات می شود .الگوریتم معرفی شده همچنین توازن بار بهینه را در زمان اجرا برای حداکثر کارآیی منابع ممکن می سازد . کارآیی الگوریتم معرفی شده با مراکز داده سنتزی مورد بررسی قرار می گیرد . همچنین نتایج الگوریتم HGA با الگوریتم های جدید و پیشرفته مقایسه می شود . نتایج آزمایشات نان دهنده این است که الگوریتم معرفی شده ما مجموع زمان اجرای کمتری نسبت به سایر الگوریتم های معرفی شده دارد .

**کلمات کلیدی :** جریان کاری ، الگوریتم ژنتیک ، هیوریستیک ، گراف های بدون جهت دوری

[1] دانشجوی کارشناسی ارشد دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته کرمان
Email : Iranmanesh.collegian@gmail.org
[2] HGA

**۱.مقدمه**

جوامع علمی همواره با شبیه سازی و آزمایشاتی سر و کار دارند که حجم زیادی از داده ها را تولید می کنند . چندین زمینه علمی از جمله بیوانفورماتیک ، علوم زمین شناسی ، علوم کیهان شناسی با حجم وسیعی از داده ها سرو کار دارند . مدیریت این حجم انبوه از داده ها پیچیدگی بالایی دارد[3] . افزایش پیچیدگی ها و ناهمگنی برنامه های کاربردی[۳] منجر به وجود آمدن چالش های جدیدی در زمینه مدیریت داده های کلان شده است . محققان برای مدیریت آزمایشات با مقیاس وسیع از سیستم های مدیریت جریان کاری[۴] استفاده می کنند[4]. سیستم های مدیریت جریان کاری داده های توزیع شده مقیاس بزرگ را سازماندهی و مدیریت می کنند . بعلاوه ، این سیستم ها برای اجرای برنامه های کاربردی داده محور ، پیچیدگی ها را ساده می کنند[26] .

تعداد بسیاری از سیستم های مدیریت جریان کاری در دهه اخیر معرفی شده و توسعه یافته اند و جهان علم به صورت گسترده از آنها استفاده می کند مانند Pegasus[11] ، Swift[23] ، Kepler[15] ، Taverna[13] ، Trident[5] و Konstanz[6] که داده کاو اطلاعاتی می باشد . بهینه سازی جریان کاری قسمت حیاتی همه ی سیستم های مدیریت جریان کاری می باشد . که در طول عملیات نگاشت منابع استفاده می شوند[10] . استراتژی اختصاص کارآی منابع به صورت مستقیم روی کارآیی کل سیستم مدیریت جریان کاری اثر می گذارد . بنابراین ، جامعه علمی تمام تلاش خود را برای معرفی روش های جدید جهت بهینه سازی جریان کاری به کار گرفته است و بر تمامی چالش هایی که در جهت توسعه برنامه های کاربردی علمی به وجود می آید غلبه کرده است .

مهم ترین قسمت این مقاله الگوریتم زمان بندی جریان کاری جدیدی تحت عنوان الگوریتم ژنتیک ترکیبی[۵] (HGA) می باشد . الگوریتم ژنتیک ترکیبی الگوریتمی است که یک هیوریستیک برای رسیدن به راه حل زمان بندی بهینه ( Makespan کمتر ) جهت دهی می شود . جریان های کاری معمولا تحت گراف های بدون دور جهت دار (DAG) مدل می شوند[24] ، که در آنها گره ها یا رئوس نمایش دهنده وظایف می باشند و لبه ها نمایش الویت یا وابستگی داده ای بین وظیفه ها می باشد . مدل جریان کاری که در کار ما استفاده شده است این چنین است . رفتار الگوریتم پیشنهاد شده با مجمعه داده های متفاوتی تحلیل شده است ، شامل جریان های کاری ، برنامه های کاربردی جهان واقعی[۶] و سنتزی از قبیل Montage و Cybershake .

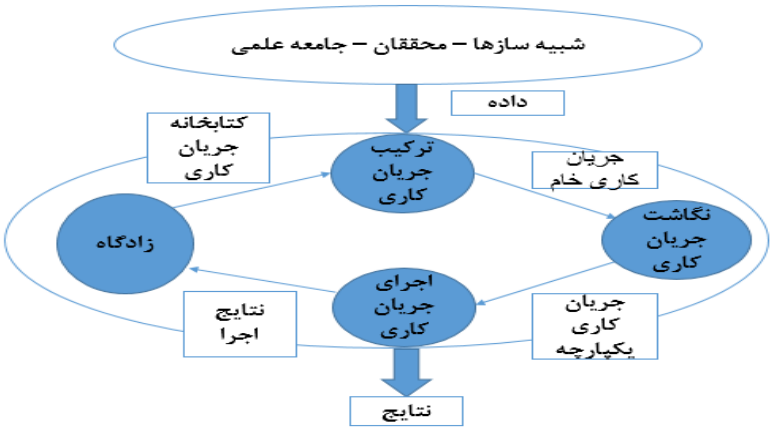
جریان های کاری Montage و Cybershake انتخاب مشترک همه ی محققان برای ارزیابی کارآیی الگوریتم های زمان بندی جریان کاری می باشد . این جریان های کاری ویژگی های متفاوتی دارند و اکثر الگوهای جریان کاری را در بر می گیرند به همین دلیل برای تحلیل کارآیی ارزیابی هر الگوریتم زمان بندی جدیدی ، ویژگی های لازم را دارا می باشند . جریان های کاری تصادفی زمینه را برای چک کردن رفتار الگوریتم معرفی شده در رابطه اشکال گوناگون جریان کاری با اندازه های متفاوت که مقادیر متفاوتی از درجه خروجی و نرخ محاسباتی دارند ، فراهم می کند .

[3] Applications
[4] Workflow Management System
[5] Hybrid Genetic Algorithm
[6] Real World

باقیمانده مقاله به صورت زیر سازماندهی می شود . قسمت ۲ پیش زمینه را فراهم می کند . جزییات کارهای مربوطه در قسمت ۳ آمده است . قسمت ۴ الگوریتم پیشنهادی را توصیف می کند . تحلیل کارآیی و بحث در مورد آن در قسمت ۵ آورده شده است و در قسمت ۶ نتیجه گیری مقاله آورده شده است .

## ۲.پیش زمینه

جریان های کاری با تجزیه برنامه های کاربردی به وظیفه های کوچکتر ، که این وظایف در جهت رسیدن به نتایج به ترتیب خاصی کامل می شوند ، مدیریت برنامه های کاربردی داده محور را ساده تر کرده اند . مفهوم جریان های کاری پیچیدگی آزمایشات مقیاس بزرگ را کاهش داده است [10]. سیستم مدیریت جریان کاری برای سازماندهی و مدیریت داده های بزرگ با استفاده از روش گام به گام ، طراحی شده است[8] . مهم ترین مراحل سیستم مدیریت جریان کاری در شکل ۱ نشان داده شده اند .




**شکل ۱. مراحل اصلی سیستم مدیریت جریان کاری**

سیستم های مدیریت جریان کاری متفاوت فرمت های ورودی متفاوتی دارند . مرحله ترکیب جریان کاری ، یک جریان کاری در سطح بالا می سازد . جریان کاری خام ، اجزای نرم افزاری و داده ای مورد نیاز برای اجرا را بدون مشخص کردن جزییات فیزیکی منابع تعیین می کند . در مرحله بعدی ، جریان کاری سطح بالا درون منابع فیزیکی نگاشت می شود و طرح اجرایی را به فرم جریان کاری یکپارچه تهیه می سازد . در مرحله نگاشت اختصاص منابع بهینه می شود . سپس جریان کاری نگاشت شده درون منابع فیزیکی زمان بندی و اجرا می شود . کارآیی اجرای جریان کاری مانیتور می شود و نتایج جمع آوری می شوند . اختصاص منابع در مرحله نگاشت بهینه می شود و در طول زمان اجرا ، زمان بندی منابع انجام می شود[4] . زمان بندی بهینه ، کارآیی منابع را افزایش می دهد[16] . ما روی مراحل بعدی بهینه سازی در این مقاله تمرکز داریم . الگوریتم ژنتیک ترکیبی ، الگوریتمی است که makespan یا مدت زمان اجرای جریان کاری را بهینه می سازد و میزان بار توزیع شده بین منابع ا بهینه می سازد .

در این مقاله ما از روشی ترکیبی استفاده نموده ایم یعنی از زمان بند هیوریستیک مشهور HEFT[7] (سریعترین زمان اتمام ناهمگن ) در جمعیت اولیه استفاده نموده ایم . بعلاوه عملیات توازن بار ، احتمال توزیع بار به صورت مساوی بین منابع را افزایش می دهد . این ویژگی ها قدرت الگوریتم معرفی شده ما را در بدست آوردن زمان بندی بهتر با makespan کمتر افزایش داده است. الگوریتم HEFT هیوریستیک زمان بندی مشهوری بر مبنای لیست است که از بسیاری از الگوریتم های زمان بندی بهتر است[21] . از خروجی زمان بند HEFT به عنوان جمعیت اولیه در روش ترکیبی معرفی شده استفاده می نماییم . زمان بندی بر مبنای لیست فرآیندی ۲ مرحله ای است . در مرحله ی اول لیستی از الویت وظیفه ها تهیه می شود و در مرحله ی دوم براساس لیست وظیفه ها ، پردازنده ها به وظایف اختصاص می یابند . در هیوریستیک HEFT ، لیست الویت به ترتیب نزولی مرتبه امتیاز سطح b تولید می شود . اختصاص پردازنده ها به وظیفه ها براساس کمترین زمان اتمام است . مشخصه هایی که در HEFT استفاده می شوند عبارتند از : b-level ، زمان آغاز سریعتر و زمان پایان سریعتر .

این ویژگی ها به صورت زیر تعریف می شوند :

- **سطح امتیاز b-level :**

$$rank_b(v_n) = \overline{w}_n + \max \{\bar{d}_{nm} + rank_b(v_m)\} \quad (1)$$

که $w_n$ میانگین هزینه اجراست (مقدار زمان مورد نیاز برای اجرای وظیفه بر روی گره اجرایی) $\bar{d}_{nm}$ میانگین هزینه ی ارتباطی است ( هزینه انتقال داده از والد به گره فرزند در جریان کاری) بین گره های n و m و $v_m$ گره بعد از $v_n$ می باشد . سطح امتیاز هر وظیفه با تابع بازگشتی معرفی شده در بالا محاسبه می شود و لیست الویت وظیفه به وسیله ترتیب مقادیر امتیاز نزولی ، تولید می شود .

- **کمترین زمان آغاز[8] (EST)**

کمترین زمان آغاز پردازنده $P_k$ برای وظیفه $v_n$ به صورت زیر تعریف می شود :

$$EST(v_n, P_k) = \max \{ avai[k] , \max FT(v_m) + \bar{d}_{nm}) \} \quad (2)$$

که در آن avai [k] نشان دهنده زمانی است که پردازنده k آماده اجرای وظیفه ای جدید است و $\bar{d}_{nm}$ هزینه ارتباطی وظیفه $v_n$ و $v_m$ است ، وظیفه $v_m$ پیش از وظیفه $v_n$ قرار دارد ( وظیفه $v_m$ والد وظیفه $v_n$ است ). وظیفه $v_n$ ممکن است پیش از یک والد داشته باشد . بنابراین ماکزیمم مجموع FT و $\bar{d}_{nm}$ بین والدهای آن در معادله شماره ۲ در نظر گرفته شده است . که این امر منجر به ماکزیمم تاخیر در اجرای وظیفه $v_n$ می شود .

- **کمترین زمان پایان[9] (EFT)**

[7] Heterogeneous Earliest Finish Time
[8] Earliest start time
[9] Earliest finish time

ویژگی EFT اختصاص پردازنده را تعیین می کند . کمترین زمان اتمام پردازنده $P_K$ برای وظیفه $v_n$ به صورت زیر تعریف می شود :

$$\mathrm{EFT}(v_n, P_K) = \overline{w}_n + \mathrm{EST}(v_n, P_K) \quad (3)$$

که $\overline{w}_n$ میانگین اجرای هزینه $v_n$ بر روی همه ی پردازنده ها می باشد و EST کمترین زمان شروع پردازنده $\mathrm{P_K}$ برای وظیفه $v_n$ می باشد که در معادله شماره ۳ آورده شده است .

**۳. کارهای مربوط**

مساله زمان بندی در طول دهه های اخیر به شکل گسترده ای مورد مطالعه قرار گرفته است . تعداد بیشماری از الگوریتم ها و هیوریستیک های تکاملی بری حل مساله زمان بندی جریان کاری پیشنهاد شده اند . مسیر بحرانی تغییر یافته (MCP) [21]الگوریتم زمان بندی است که از پارامتر ALAP[۱۰] برای ساخت لیست الویت وظیفه های جریان کاری استفاده می کند . پارامتر ALAP بیشترین زمانی است که اجرای یک وظیفه n می تواند تاخیر داشته باشد ، بدون آنکه در الویت های اجرای وظایف خللی وارد شود . الگوریتم mcp وظیفه ها را به ترتیب نزولی پارامترهای ALAP مرتب کرده است . با اختصاص وظیفه ها به ماشین ها برای اجرا شدن با توجه به کمترین زمان اتمام وظیفه ها انجام شده است .

در متاهیوریستیک ، تکنیک های متفاوتی وجود دارد از قبیل الگوریتم های ژنتیک[1] بهینه سازی کلونی مورچه[17] و بهینه سازی اجتماع ذرات[20] که برای حل مساله زمان بندی وظیفه موجودند . اگر چه الگوریتم ژنتیک بین محققان مقبولیت بیشتری دارد بر این اساس که قابلیت ترکیب با سایر پارادیم ها را دارد .

روش های ترکیبی گوناگونی برای مساله زمان بندی موجود است ولی تمرکز ما بر روی ترکیب هیوریستیک با الگوریتم ژنتیک برای بدست آوردن راه حل بهینه (کمترین Makespan) در مورد زمان بندی جریان کاری است . الگوریتم های ترکیبی متنوعی برای ترکیب الگوریتم ژنتیک و هیوریستیک ها معرفی شده اند . الگوریتم HSCGS (زمان بندی ترکیبی ژنتیک-هیوریستیک جانشینی) یکی از این الگوریتم ها می باشد که در آن نویسندگان هیوریستیک و الگوریتم ژنتیک را با هم ترکیب کرده اند[22] . در مرحله اول هیوریستیکی به نام SCLS (زمان بندی هیوریستیک براساس لیست جانشینی ) برای زمان بندی استفاده شده است . SCLS هیوریستیکی برمبنای لیست می باشد که در آن لیستی از الویت وظیف ها ساخته شده است که این لیست الویت براساس امتیاز وظیفه ها و جانشینان آنان ساخته شده است . در مرحله ی دوم زمان بند تولید شده به وسیله هیوریستیک SCLS با الگوریتم ژنتیک ترکیب شده است . بعد از چند نسل الگوریتم همگرا می شود . زمان بندی با مدت زمان منطقی تولید می شود . الگوریتم دیگری که اخیرا معرفی شده است ، الگوریتم PEGA (الگوریتم ژنتیک کارا ) می باشد که با استفاده از روش ترکیبی ، makespan را کمینه می نماید [1]. عیب آن پیچیدگی زمان بالای آن می باشد . الگوریتم معرفی شده ما علاوه بر اینکه makespan را کمینه می سازد توازن بار بین منابع را نیز مهیا می سازد . بعلاوه ، هیوریستک را با الگوریتم HGA ترکیب نموده ایم که کارآیی HGA را با استفاده از فراهم کردن جستجوی هدایت شده افزایش می دهد .

الگوریتم ژنتیکی که اخیرا معرفی شده ، الگوریتم ژنتیک با صف های الویت چندگانه (MPQGA) می باشد که صف های چندگانه وظایف ( لیست های الویت ) را در الگوریتم ژنتیک به کار می برد . کروموزوم ها با لیست های الویت نمایش داده

[10] As Late As Possible

می شوند و الویت ها با b-level ، t-level یا هردو پارامتر مشخص می شوند و DAG از این لیست های الویت استفاده می کند .

مقادیر سطح b ($rank_b(v_n)$) و سطح t ($rank_t(v_n)$) برای هر کدام از وظایف توسط معادلات شماره ۱ و شماره ۴ محاسبه می شوند .

$$rank_t(v_n) = \max\{\overline{w}_n + \bar{d}_{nm} + rank_t(v_m)\} \ , \ v_m \in \text{pred}(v_n) \qquad (4)$$

$\overline{w}_n$ میانگین هزینه اجرا می باشد . $\bar{d}_{nm}$ میانگین هزینه ارتباطی بین گره های m و n می باشد و $v_m$ والد $v_n$ می باشد . سطح- t ( امتیاز روبه پایین ) هر وظیفه با معادله بازگشتی شماره ۴ محاسبه می شود . لیست الویت وظیفه با ترتیب صعودی مقادیر سطح t ساخته می شود .

جمعیت اولیه از کروموزوم هایی تشکیل شده است که این کروموزوم ها شامل صف های الویت و نگاشت وظایف به پردازنده ها بر اساس پارامتر کمترین زمان اتمام می باشند[25] . برازش هر کدام از کروموزوم ها با روش فرمان چرخان محاسبه می شود و بعد از آن کروموزوم های مناسب برای عملیات الگوریتم ژنتیک انتخاب می شوند . عملگرهای ژنتیک الگوریتم MPQGA عبارتند از ترکیب تک نقطه ای و عملیات جابه جایی جهش . الگوریتم معرفی شده ما به نسبت MPQGA تفاوت هایی دارد . به این علت که نمایش کروموزوم در الگوریتم ما متفاوت است در HGA اندازه ی کروموزوم را دو برابر تعداد وظیفه های موجود در DAG در نظر می گیریم که نیمی از آن ها شامل اختصاص پردازنده تصادفی به هریک از وظیفه هاست و نیمه ی دیگر آن نشان دهنده ی لیست الویت وظیفه های درون DAG می باشد .الگوریتم HGA از لحاظ عملیات دوگانه ترکیب و جهش با MPQGA متفاوت می باشد و این عملیات دو برابر نسبت به عملیات سنتی الگوریتم های ژنتیک کارآیی بالاتری دارند . توازن بار یکی از نقاط برتری الگوریتم ما است .

داوود و کارما الگوریتم ۲ مرحله ای را پیشنهاد کرده اند به اسم الگوریتم ژنتیک هیوریستیک ترکیبی (H2GS)[9] . در مرحله اول هیوریستیک طولانی ترین مسیر بحرانی پویا (*LDCP*) زمان بندی ها را در فرم کروموزوم ها تولید می کند و پس از آن ، این زمان بندی ها در جمعیت اولیه الگوریتم ژنتیک شخصی سازی شده مورد استفاده قرار می گیرند . الگوریتم ژنتیک استفاده شده در این زمان بندی GAS نام دارد . زمان بندی های تولید شده توسط هیوریستیک به عنوان شتاب دهنده سرعت الگوریتم عمل می کنند و در مرحله ی دوم به GAS در رسیدن به زمان بندی نهایی کمک می کنند . نمایش دو بعدی کروموزوم در *GAS* استفاده می شود و عملگرهای شخصی سازی شده برای جستجوی فضای مساله استفاده می شوند . در روش ما از نمایش تک بعدی کروموزوم که پیچیدگی کمتری دارد استفاده می شود . عملگرهی ژنتیک (جهش و ترکیب) عملگرهای دو گانه ای هستند که فرآیند جستجو برای به دست آوردن راه حل بهینه (MAKESPAN کمینه ) را تسریع می بخشند . براساس ویژگی های اساسی ، الگوریتم HGA توانایی رسیدن کارا به بهترین راه حل را دارا می باشد . در جهت بهبود الگوریتم ژنتیک چند منظوره ، روش NSGA-II معرفی شده است . در محیط مبتنی بر گرید ، ارزش اجرای جریان کاری مهمه ترین محدودیت محسوب می شود . بنابراین بعلاوه MAKESPAN هزینه هم به عنوان هدف تابع در الگوریتم معرفی شده در نظر گرفته شده است . سیستم فازی برای رسیدن به هدف سوم که توازن بار می باشد به کار گرفته شده است .

الگوریتم معرفی شده این پارامترها را بهبود می بخشد اما به صورت قابل توجهی پیچیدگی ها را افزایش می دهد . اما ما در الگوریتم مان با زمان بسیار کمتر MAKESPAN و توازن بار رابهبود می بخشیم .

### ۴. الگوریتم معرفی شده HGA

برنامه ی کاربردی را فرض کنید که به صورت جریان کاری نمایش داده می شود و به صورت گراف DAG مدل می شود .گراف DAG با یک تاپل (V,D,P) نمایش داده می شود که V مجموعه تعداد N وظیفه است ، D مجموعه یال های نمایش دهنده الویت بینه هر زوج از وظیفه هاست و P مجموعه پردازنده های در دسترس است .

عناصر تاپل DAG عبارتند از :

$V=\{v_1,v_2,\ldots,v_n\}$

$D=\{d_1,d_2,\ldots,d_n\}$

که در آن d نمایش دهنده یال بین دو گره است .

$P=\{p_1,p_2,p_3,\ldots,p_m\}$

عنصر D در نمایش DAG بیان کننده الویت بین وظیفه هاست که معنی آن این است که برای هر زوج وظیفه $v_i,v_j(\ )\in V$ ، اجرای وظیفه $v_j$ تا زمان اتمام وظیفه $v_i$ نمی تواند آغاز شود . این یعنی اینکه $v_j$ فرزند $v_i$ می باشد . هر گره $v_i$بدون والد وظیفه ی ورودی نامیده می شود و وظیفه $v_j$ بدون هیچ گونه وظیفه والدی ، وظیفه ی خروجی نامیده می شود . ممکن است بیش از یک وظیفه ورودی داشته باشیم ولی تنها یک وظیفه خروجی داریم . در طول زمان بندی ، الویت ها نباید مختل شوند به این خاطر که اگر الویت بین وظیفه ها از بین رود زمان بندی نامعتبر خواهد بود . هزینه انتقال داده از پردازنده ای به پردازنده ی دیگر هزینه ارتباطاتی نامیده می شود که با عنصر D در تاپل DAG نمایش داده می شود ، هزینه ارتباطی با $d_i$ نمایش داده می شود که عددی حقیقی است . هزینه ی اجرای هر وظیفه $v_i$ بر روی پردازنده j از مجوعه ی پردازنده های در دسترس P به صورت $w_{ij}$ نمایش داده می شود . هزینه ی اجرای هر کدام از وظیفه ها بر روی هر کدام از پردازنده ها توسط مقادیر حقیقی بیان می شوند که با ماتریس n×m نمایش داده می شوند ، که در آن nتعداد وظیفه های برنامه کاربردی است و mتعداد پردازنده هاست . مساله زمان بندی جریان کاری به صورت اختصاص مجموعه ای از وظیفه های V به مجموعه ای از پردازنده ها تعریف می شود بطوریکه :

- *محدودیت الویت بین وظیفه ها حفظ شود .*
- *مدت زمان زمان بندی کاهش یابد .*
- *بارگزاری متوازن بین مناب انجام شود .*

اهداف الگوریتم زمان بندی جریان کاری معرفی شده ما ، بهینه سازی makespan و توازن بار بین منابع است . در این مقاله ، فرض بر این است که اطلاعاتی از قبیل اندازه جریان کاری ، الویت وظیفه ها و منابع از قبل مشخص شده اند و ما از این اطلاعات آگاه هستیم . در چنین الگوریتم هایی که همه اطلاعات از پیش دانسته شده اند ، الگوریتم های ایستا[11] یا معین نامیده می شوند . زمان بندی جریان کاری اثبات می شود که مساله ای Np-Hrad است .

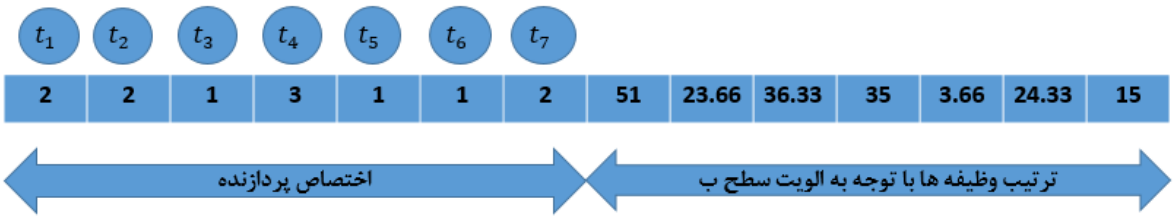
#### ۴,۲.الگوریتم پیشنهادی

[11] Static

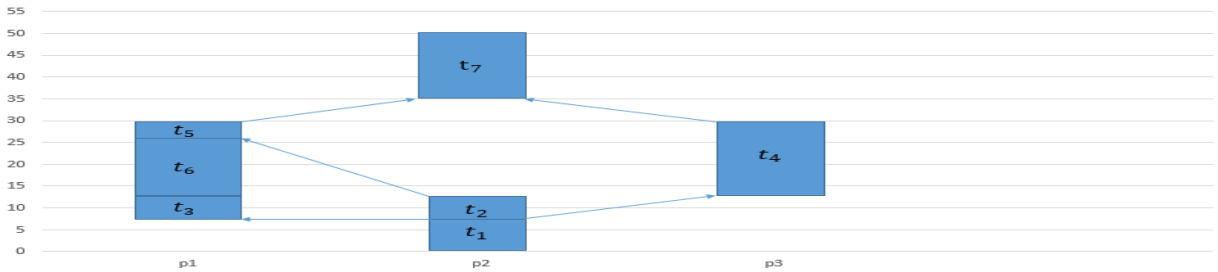
الگوریتم پیشنهادی روشی ترکیبی می باشد که ترکیب هیوریستیک و الگوریتم ژنتیک می باشد . طرح کلی HGA در الگوریتم شماره ۱ آورده شده است . الگوریتم HGA با مجموعه ای از زمان بندی ها به فرم جمعیت اولیه آغاز می شود . اندازه جمعیت $N_P$ ، تعداد نسل ها $N_g$ ، احتمالات نخبگی ، ترکیب و جهش به عنوان ورودی کاربر در نظر گرفته شده اند . الگوریتم شماره ۱ (خط اول ) با مجموعه ای از کروموزوم ها به عنوان جمعیت اولیه آغاز می شود . سپس خروجی زمان بندی HEFT درون جمعیت اولیه قرار می گیرد . هیوریستیک HEFT جهت افزایش کارآیی الگوریتم ، راهنمایی و جهت را فراهم می کند. جستجوی جهت دار به الگوریتم HEFT کمک می کند بعد از تکرار تعداد نسل کمتری به همگرایی برسد . متغیر $N_g$ که نشان دهنده ی تعداد نسل هاست به عنوان شرط خاتمه در نظر گرفته شده است[2] . دستورات موجود در حلقه خط سوم به تعداد نسل ها یعنی $N_g$ اجرا خواهند شد [7].

جمعیت از یکسری اشخاص به نام کروموزوم تشکیل شده است . نمایش کروموزوم مستقیم است و هر کروموزوم از ۲ قسمت تشکیل شده است . قسمت سمت چپ نمایش دهنده اختصاص منابع است و اندازه آن برابر با تعداد گره های DAG می باشد . ژن ها نمایش دهنده تعداد پردازنده ها از شماره ۱ تا شماره P می باشند ، که P بیشینه تعداد پردازنده های در دسترس است .

اندازه ی نیمه دوم برابر با تعداد گره های موجود در گراف DAG می باشد که نشان دهنده ترتیب یا توالی وظیفه هایی است که باید زمان بندی شوند . محدودیت الویت های ترتیب یا توالی وظیفه هایی است که باید زمان بندی شوند . محدودیت الویت ها نباید مختل شوند . در غیر این صورت کروموزوم ، کروموزومی نامعتبر تلقی خواهد شد و مایش دهنده زمان بندی صحیح نمی باشد . هر کروموزوم نشان دهنده ی یک زمان بندی معتبر است . به عنوان مثال کروموزوم نشان داده شده در شکل ۲ و زمان بندی متناظر آن روی ۳ پردازنده به فرم چارت گانت در شکل شماره ۳ نشان داده شده است .




**شکل ۲.نمایش کروموزوم**



**شکل۳.زمان بندی متناظر با کروموزوم مثال نشان داده شده در شکل ۲**

نیمه ی اول به صورت تصادفی تولید می شود و نیمه ی دوم که محاسبه ی امتیازات وظیفه هاست با استفاده از معادله ی شماره ۱ محاسبه می شود . ترتیب وظیفه ها در مثال کروموزوم نشان داده شده در شکل شماره ۲ به ترتیب نزولی امتیازات وظیفه ها می باشد . لیست الویت وظیفه ها عبارتند از {۵و۷و۲و۶و۴و۳و۱}. اگر وظیفه ها بر روی پردازنده براساس اختصاص پردازنده ها نشان داده شده در نیمه ی اول کروموزوم شکل شماره ۲ نگاشت شوند ، آنگاه مدت زمان بندی ۵ واحد زمانی خواهد شد . زمان بندی متناظر در شکل ۳ نشان داده شده است . در فرآیند ارزیابی الگوریتم شماره ۱ ( خطوط ۴ و ۵و ۶) ، هر زمان بندی در نسلی بر اساس تابع برازش ارزیابی می شود . برازش کروموزوم $X$ به صورت زیر تعریف می شود :

$$F(X) = C / Makespan(X) \quad (5)$$

که در معادله شماره ۵ ، C مقدار ثابتی است و Makespan به صورت زیر تعریف می شود :

$$Makespan(X) = F.T(t_{exit}) \quad (6)$$

که در معادله شماره ۶ $F.T(t_{exit})$ زمان اتمام گره خروجی است . در مواردی که بیش از یک گره خروجی داریم ، Makespan به وسیله معادله ی شماره ۷ تعریف می شود :

$$Makespan(X) = Max(F.T(t_i)) \ ,i=\{1,2,3,\ldots,n\} \quad (7)$$

ارزیابی براساس تابع برازش تعریف شده در معادله شماره ۵ انجام می شود . نخبگی در خطوط ۷ و ۸ در HGA تعریف می شود که بهترین کروموزوم های هر نسل را درنسل بعدی کپی می کند . جزئیات انتخاب خط شماره ۹ در الگوریتم شماره ۲ داده شده است که در آن کروموزوم های مناسب با انتخاب رقابتی دوبه دو برای عملیات ژنتیک به کارگیری می شوند . سپس بر روی کروموزوم های انتخاب شده عملیات ترکیب و جهش انجام می شود . هر دو عملیات الگوریتم ژنتیک در الگوریتم HGA تغییر یافته اند . نسبت به کارهای انجام شده معرفی شده که از الگوریتم های ژنتیک موجود استفاده می کنند ، با استفاده از ترکیب عملیات جهش و ترکیب تک نقطه ای و عملیات جهش وترکیب دونقطه ای ، قدرت هر دو عملگر ۲ برابر شده است . شبه کد تابع ترکیب در الگوریتکم شماره ۳ آورده شده است و شبه کد تابع جهش در الگوریتم شماره ۴ آورده شده است . در پایان هر نسل ، با استفاده از تابع توازن بار ، جستجوی همسایگی انجام شده است . شبه کد فرآیند توازن بار در الگوریتم شماره ۵ آورده شده است . هر نسل در خط ۱۴ الگوریتم ۱ کامل می شود و برای تولید نسل بعدی الگوریتم به خط شماره ۳ پرش می کند این عملیات تا زمانی که شرط خاتمه ارضا شود ، ادامه می یابد .

منابع باید به صورتی بکارگیری شوند که هیچ کدام از منابع در مدت زمان طولانی بی کاری یا در حالت پربار (داشتن بار کاری بیش از ظرفیت ) نباشد . الگوریتم شماره ۵ (توازن بار ) به نحوی وظایف را بین پردازنده ها توزیع می کند که همه پردازنده ها در کمترین تفاوت زمانی ، کار را پردازش و تکمیل کنند . در پایان هر تکرار ، الگوریتم شماره ۵ از طریق توزیع بار میان پردازنده ها کیفیت زمان بندی را افزایش می دهد .(از نظر مدت زمان بندی )

**الگوریتم شماره ۱ : الگوریتم ژنتیک ترکیبی**

**Input :** $N_p$(population size) - ∝(Elitism Rate) – β (Mutation Rate) - $N_g$(Number of Generation)
**Output :** S Near Optimal Solution
1 Intial population generation of size $N_P$-1
2 Seed HEFT scheduling as a chromosome in $N_p$
3 for i=1 to $N_g$ do
/* Evaluation
4 for j=1 to $N_p$ do
5 Compute fitness value of each chromosome
6 end for
/* Elitism
7 Number of Elite Chromosome E= ∝ * $N_p$
8 Select E chromosome having best fitness values as $N_E$
9 Select Routine as shown in Algorithm 2
10 Crossover Routine as shown in Algorithm 3
11 Mutation Routines as shown in Algorithm 4
/* Next generation
/* $N_s$ are the selected chromosome
12 $N_P = N_E + N_S$
13 Load Balancing Routine as shown in Algorithm 5
14 end for
15 Return Near optimal (makespan) solution S

**الگوریتم شماره ۲ : تابع انتخاب**

1 Pick two chromosome at random from initial population
2 for k=1 to $N_g$ do
3 if f(x) < f(y) then
4 Select chromosome x as $N_s$
5 end if
6 end for

**الگوریتم شماره ۳ : تابع ترکیب**

**Data :** Number of crossover C = $N_s/2$
1 for k = 1 to C do
2 Randomly select two chromosome $x_a$ and $x_b$ for mating
3 Off springs by single point crossover is $x_c$ and $x_d$
4 Off springs by double point crossover is $x_e$ and $x_f$
5 Compute fitness values of $x_c$ , $x_d$ , $x_e$ and $x_f$
6 Select two best off springs as $N_S$
7 end for

**الگوریتم شماره ۴ : تابع جهش**

**Data :** Number of mutations M = βx$N_s$
1 for l = 1 to M do
2 Randomly select a chromosome $x_i$ from $N_s$
3 Perfect single point mutation off springs = $x_j$
4 Perform double point mutation off springs = $x_k$
5 Compute fitness of $x_j$ and $x_k$
6 Select offsprings with better fitness value as $N_s$
7 end for

**الگوریتم شماره ۵ : تابع توزیع بار**

1 Calculate LB of chromosome for all chromosome using eq 8
2 Select 50% of chromosome with higher LB values
3 forall the selected chromosome do
4 Identify the overloaded processor $p_i$ from chromosome $y_{ol}$
5 Randomly select a processor $p_i$ such that $p_i \neq p_l$
6 Replace the $p_{ol}$ by $p_n$ at two places
7 A new chromosome $x_{lb}$ is formed
8 if f($x_{lb}$) > f($y_{ol}$) then
9 Replace $x_{lb}$ by $y_{ol}$
10 end if
11 end forall

پارامتر توازن بار (LB) در معادله شماره ۸ توضیح داده شده است . از این پارامتر در الگوریتم شماره ۵ برای تعیین کیفیت توازن بار در هر کدام از زمان بند ها استفاده شده است :

$$LB = S_{Length} - \min \{FT_1. FT_2 \dots, FT_n\} \quad (8)$$

در معادله شماره ۸ $S_{Length}$ مدت زمان بندی کروموزوم مربوطه می باشد و $FT_n$ زمان اتمام پردازنده ام می باشد . هر اندازه که مقدار LB بزرگتر باشد نشان دهنده توازن بار بدتر می باشد . پیچیدگی الگوریتم معرفی شده $nm+n^2$ است که n تعداد نسل هاست و m اندازه جمعیت است . با افزایش هر کدام از این دو فاکتور (n,m) زمان اجرای HGA هم افزایش می یابد .

**۵. تحلیل و بحث در مورد کارآیی**

در این قسمت کارآیی الگوریتم پیشنهاد شده مورد تحلیل قرار می گیرد . الگوریتم HGA با استفاده از مجموعه داده ها با ویژگی های متمایز مورد ارزیابی قرار می گیرد و نتایج به دست آمده با الگوریتم های انتخاب شده که در ادامه مطرح می شوند ، مقایسه شده است . ماهیوریستیک های MCP و HEFT ، الگوریتم ژنتیک تکاملی (PEGA) و در آخر الگوریتم های ژنتیک ترکیبی (HSCGS,MPQGA) برای تحلیل های مقایسه ای با الگوریتم پیشنهادی استفاده می کنیم .این الگوریتم ها

برا اساس روش متفاوتی که استفاده می کنند زمینه مناسبی برای مطالعه و مقایسه رفتار HGA می باشند .ما میانگین مدت زمان بندی[۱۲] (ASL) بدست آمده از ۱۰۰۰ اجرا را به عنوان پارامتر کارآیی انتخاب کرده ایم .

الگوریتم پیشنهادی در سیستمی بزرگ و ناهمگن شبیه سازی شده است . منابع همانند خطوط شبکه در محیط اجرایی ناهمگن می باشند . به این دلیل که وظیفه ها از نظر نوع بار کاری متفاوت هستند ، لذا برای محاسبه ی زمان های اجرایی هر وظیفه روی گره های اجرایی هم ناهمگنی گره های اجرایی و هم ناهمگنی وظیفه ها در نظر گرفته شده است . به صورت مشابه ناهمگنی لینک های شبکه با هزینه های متفاوت بر روی یال های گراف شبکه در نظر گرفته می شود . بعد از چندین تکرار بهترین پارامترها برای الگوریتم پیشنهادی برای اینکه بهترین نتایج به دست آورد عبارتند از احتمال ترکیب ۰,۸ و احتمال جهش ۰,۲ می باشد . برای ساده بودن شبیه سازی اندازه جمعیت و تعداد نسل ها هر دو ۱۰۰ در نظر گرفته می شوند .

HEFT هیوریستیکی مشهور است که زمان بندی مناسبی تولید می کند . ما از خروجی زمان بندی HEFT درون جمعیت اولیه الگوریتم HGA برای افزایش سرعت فرآیند رسیدن به makespan کمینه استفاده می کنیم . ما شبیه سازی را هم به صورت ترکیبی با هیوریستیک و هم به صورت غیر ترکیبی با هیوریستیک HEFT انجام داده ایم . نتایج را در جدول شماره ۱ آورده ایم . نتایج نشان دهنده این است که استفاده از راه حل HEFT به عنوان جمعیت اولیه زمان اجرای الگوریتم را تسریع می بخشد .

| تعداد گره های جریان کاری | الگوریتم HGA به همراه هیوریستیک HEFT | الگوریتم HGA بدون هیوریستیک HEFT |
|---|---|---|
| ۱۰۰ | 6.55 | 7.94 |
| ۵۰۰ | 30.66 | 39.99 |
| 1K | 72.11 | 80.07 |
| 2K | 143.52 | 159.45 |

**جدول شماره ۱ . مقایسه نتایج با و بدون استفاده از هیوریستیک HEFT**

مجموعه داده های انتخاب شده برای شبیه سازی شامل جریان های کاری ساخته شده و برنامه های جهان واقعی می باشند . ویژگی های مجموعه های داده ای در جدول شماره ۲ آورده شده است . محققان برای ارزیابی الگوریتم های جدیدی بر روی بنجمارک ها و الگوهای جریان کاری بسیاری کار کرده اند . به عنوان مثال ، بنجمارک گرید *NAS* برای ارزیابی کارآیی سیستم های موازی و توزیع شده طراحی شده است . بنجمارک ها شامل ۴ کلاس مساله که از اپلیکشن های پویا و روان محاسباتی (CFD)[۱۳] شامل توزیع نامنظم[۱۴](ED) ، زنجیره خورشیدی[۱۵](ED) ، مجازی سازی پایپ لاین[۱۶](VP) و کیف ترکیب شده[۱۷](MB) می باشد . این بنجمارک ها نمایش دهنده ی اپلیکیشن های نوعی می باشند که بر روی سیستم های ناهمگن شبیه گرید اجرا می شوند . هر کدان از اینها شامل وظیفه های محاسباتی اند که از بنجمارک های موازی NAS[۱۸] (NGB) بدست می آیند . ED نمایانگر کلاس مهمی به نام پارامتر مطالعاتی می باشد که در آن یک برنامه با مجموعه ای از پارامترهای ورودی متفاوت به صورت مستقل چندین بار اجرا می شوند . HC نمایش دهنده ی زنجیره های طولانی از محاسبات متوالی

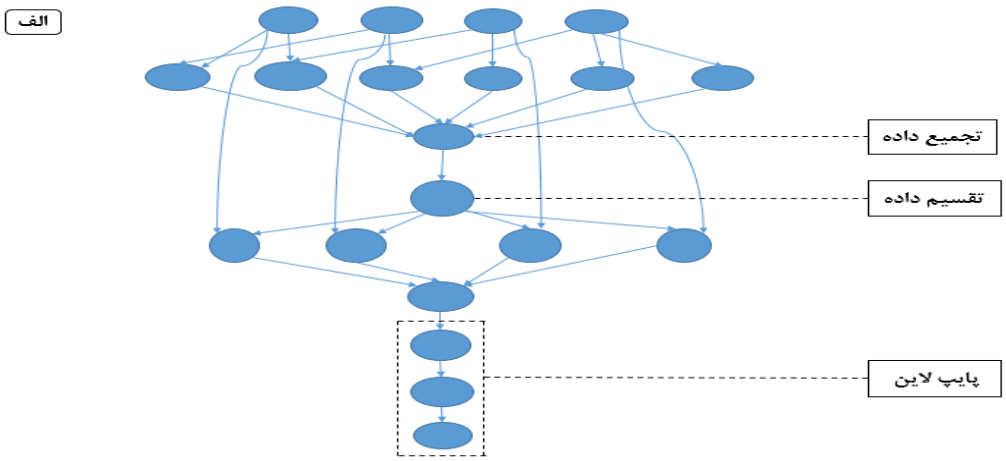
[12] Avarege scheduling time
[13] Comutional fluid dynamics
[14] Embarrassingly Distributed
[15] Helical Chain
[16] Visualization Pipeline
[17] Mixed Bag
[18] NAS parallel Benchmark

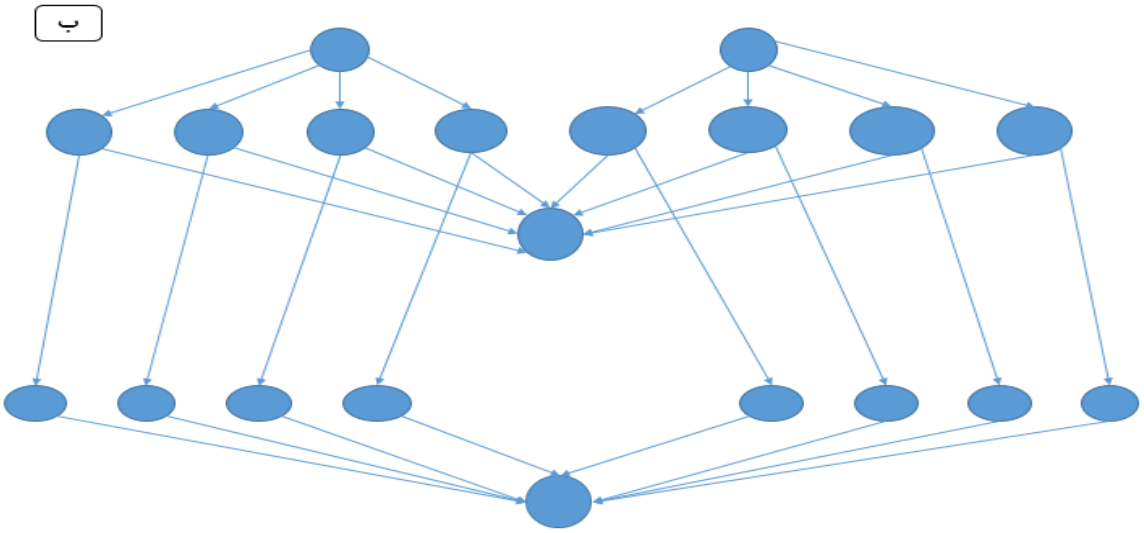
است . اجرای یک پردازش به صورت تکراری و متوالی در آن انجام می گیرد . HC درجاتی از موازی سازی و همچنین اجرای متوالی وظیفه ها را در برمی گیرد . ساختارهای MB و VP شبیه به یکدیگر هستند اما درجه ی توازی و ناهمگنی متفاوتی دارند . مجموعه های داده ای انتخاب شده یعنیMonatge،Cybershake [14]و حذف گوسی [2]، جریان های کاری بزرگ وپیچیده ای هستند که اکثر کلاس های NGB مساله را در برمی گیرند . یکسری جریان های کاری دنیای واقعیت هم وجود دارند که اجرای واقعی رفتار محیط توزیع شده را در بر دارند . این جریان های کاری به همراه جریان های کاری ترکیب شده سنتزی به صورت تصادفی تولید می شوند . ما مجموعه ای آزمایشی داریم که برای تحلیل کارآیی الگوریتم معرفی شده استفاده می شوند Montage ، Cybersake و حذف گوسی به عنوان بنجمارک های جریان کاری برای ارزیابی الگوریتم معرفی شده استفاده می شوند . این جریان های کاری نمایش دهنده ی مسائل دنیای واقعیت هستند .

### ۱.۵. بنچمارک های Cybershake و Montage

Montage موتور موزاییکی تصویر نجومی است که توسط ناسا ساخته شده است و برای تولید تصویر موزاییکی (به هم پیوسته ) از آسمان استفاده می شود . تصاویر ورودی در آن برای ساختن موزاییک (تصویر به هم پیوسته نهایی) با یکدیگر ترکیب می شوند . هندسه ی آخرین موزاییک که با تصاویر ورودی تعیین می شوند ، می تواند به شکل جریان کاری نمایش داده شود . شکل 4الف ساختار Montage با اندازه کوچک که دارای ۲۰ گره می باشد را نشان می دهد . در Montage تعداد زیادی کار با زمان اجرای کم از قبیل mprojectpp ،mDiffFIT ، mbackground و mJpeg وجود دارد که باید روی یکسری آیتم متفاوت اجرا شوند . از سویی برخی از کارها از قبیل mconcatfit ، mBgmodel و madd زمان طولانی برای اجرا نیاز دارند . جریان کاری Cybershake که برای مرکز زلزله شناسی کالیفرنیای جنوبی (scec ) استفاده می شود . جریان کاری Cybershake برای تعیین خطرات زلزله در یک منطقه استفاده می شود . Cybershake از نظر ساختار جریان کاری ساده می باشد اما می تواند مجموعه ای از داده های انبوه را مدیریت کند .

به عنوان مثالی از آن ، جریان کاری Cybershake کوچک با ۲۰ گره در شکل 4 قسمت ب نمایش داده شده است . Cybershake جریان کاری محاسبات محور و همچنین داده محور می باشد . جزییات ویژگی های هر دو جریان کاری در مقاله[14] موجود است که در آن مقاله نویسندگان اجرای ۶ جریان کاری متنوع شامل Montage و Cybershake را بحث کرده اند . بنابراین ، جریان های کاری با ویژگی های متفاوت برای ارزیابی پروژه پیشنهادی ما بسیار مناسب می باشد .






**شکل ۴ . بنچمارک های جریان کاری : (الف) Montage (ب) Cybershake [18]**

ما از جریان های کاری Montage و Cybershake برای ارزیابی HGA استفاده کرده ایم . داده این جریان های کاری را از [18] به دست آورده ایم . داده بدست آمده جزییات کاملی از اجراهای قبلی Montage و Cybershake را دارا بود . همه ی الگوریتم ها تحت شرایط یکسانی آزمایش شده اند و نتایج با مقایسه آنها جمه آوری شده است . پارامتر استفاده شده برای سنجش میزان کارآیی ، ASL برای ۱۰۰۰ اجرا بوده است . در نمودارها در محور افقی تعداد پردازنده ها (P) به نمایش گذاشته شده است . نمودارهای شکل ۵ و ۶ نمایش دهنده رفتار الگوریتم ها در پارامتر ASL است زمانی که تعداد پردازنده ها افزایش می یابد برتری الگوریتم HGA بیشتر به چشم می خورد . میانگین درصد افزایش کارآیی الگوریتم HGA در برابر الگوریتم های PEGA، MCP، HEFT، HSCGS و MPQGA به ترتیب ۷۳,۹۸٪ ، ۵۹,۵٪ ، ۲۹,۸۵٪ و۱۲,۵۸٪ و ۶,۳۲٪ می باشد که این ارقام برای جریان کاری Cybershake با ۳۰ گره به دست می آید .

| جریان های کاری | نوع | مرجع | ویژگی | تعداد گره ها | شکل |
|---|---|---|---|---|---|
| Montage | واقعی | [14] | منظم | ۲۵و۵۰و۱۰۰ | ثابت - شکل ۴- الف |
| Cybershake | واقعی | [۱۴] | منظم | ۳۰و۵۰و۱۰۰ | ثابت - شکل ۴- ب |
| حذف گوسی | شبیه سازی شده | تولید شده | منظم | ۱۴و۲۰و...و۱۰۴و۱۱۹ | ثابت - شکل ۸ |
| تصادفی | سنتزشده | تولید شده | تصادفی | ۲۰و۴۰و۶۰و۸۰و۱۰۰ | متغیر |

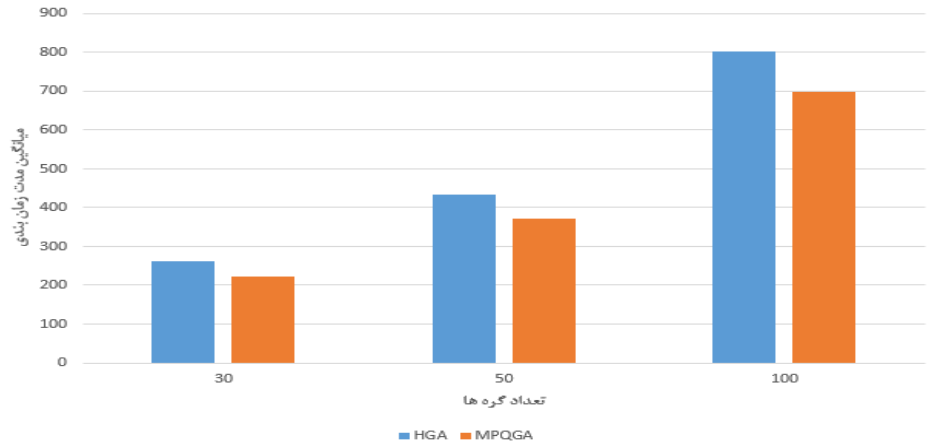
**جدول شماره ۲ – ویژگی های مجموعه های داده**

نتایج مشابه برای Cybershake با ۵۰ و ۱۰۰ گره به دست آمده که افزایش کارآیی را با افزایش اندازه جریان کاری بهتر نمایش می دهد .

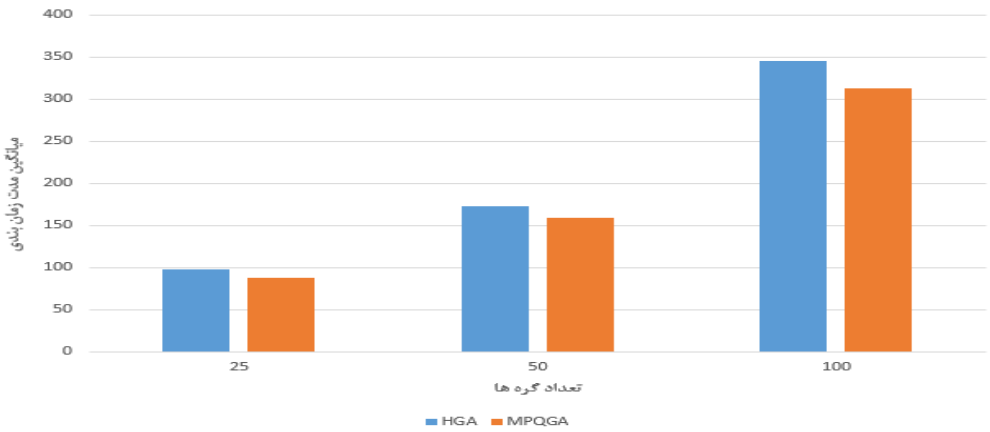
الگوریتم مطرح شده ۱۹,۵۳ و ۲۵,۰۵ درصد بهبود در مقایسه با الگوریتم HsCGS و ۲۹,۸۵ درصد و ۴۵,۹۹ درصد به نسبت الگوریتم HEFT بهبود داشته که این اعداد به ترتیب با جریان کاری Cybersahke با ۵۰ و ۱۰۰ گره به دست آمده است . هرچند الگوریتم HGA به صورت قابل توجهی در جریان کاری Cybersake نسبت به الگوریتم های *PEGA* و *MCP* برتری داشته است . برتر بودن کارآیی الگوریتم HGA با افزایش اندازه جریان های کاری Cybershake نسبت به سایر الگوریتم ها ، مقیاس پذیری الگوریتم HGA را اثبات می کند .

آزمایشات مشابه با جریان های کاری ۲۵ ، ۵۰ و ۱۰۰ گره ای Montage صورت گرفته است نتایج در شکل ۶ نشان دهنده برتر بودن کارآیی الگوریتم پیشنهادی در مقایسه با ۵ الگوریتم دیگر می باشد . در رابطه با ۲۵ گره (شکل ۶ الف) جریان کاری Montage ، الگوریتم *HGA* در مقایسه با الگوریتم PEGA ۳۲,۰۷ درصد بهتر است . نسبت به MCP ۱۶ درصد ، نسبت به HEFT ۶,۷ درصد و نسبت به HSCGS ۴۹,۷ درصد بهتر می باشد . هرچند ، کمینه میانگین درصد بهبود ۵,۴ درصد و ۸,۵۶ درصد نسبت به الگوریتم MPQGA برای جریان کاری ۵۰ گره ای و برای جریان کاری با ۱۰۰ گره می باشد . برای باقیمانده الگوریتم ها درصد میانگین افزایش کارآیی بیشتر است . قابل توجه است که کارآیی الگوریتم معرفی شده برای جریان های کاری Cybershake بسیار بهتر از جریان های کاری Montage می باشد . دلیل آن این است که هر ۲ جریان کاری ویژگی خاص خود را دارند و جریان کاری Cyberskae ، داده محورتر از جریان کاری Montage می باشد .

الف




ب



**شکل 5 . کارآیی جریان های کاری با اندازه متفاوت (الف)CyberShake (ب) Montage**

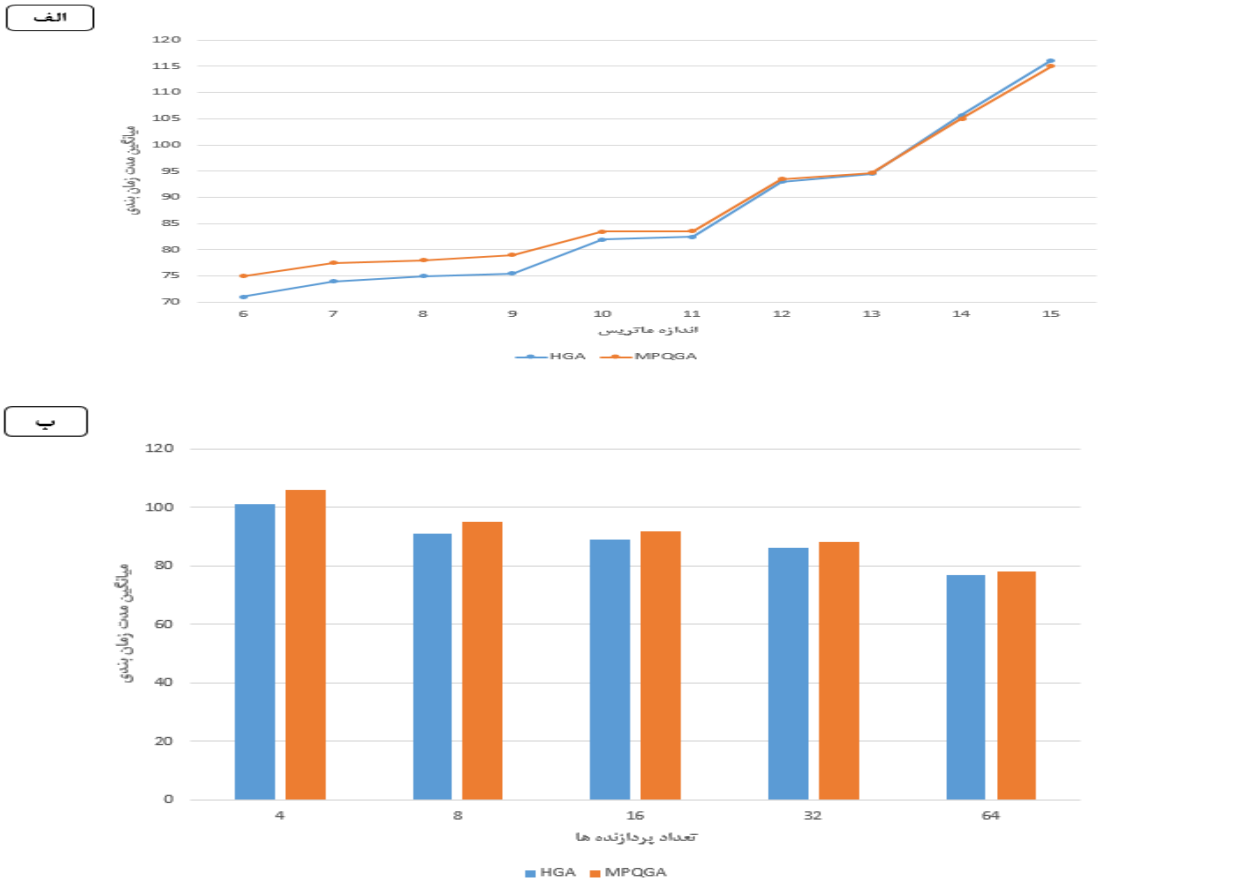
نمودارهای شکل ۷ نشان دهنده کارآیی کلی جریان کاری Cybershake و Montage با اندازه های متفاوت می باشد . میانگین مدت زمان بندی حاصل شده از الگوریتم پیشنهادی به صورت قابل توجهی کمتر از سایر ۵ الگوریتم است . کارآیی الگوریتم HGA بر روی جریان کاری Cybershake در مقایسه با MPQGA ۶۴,۵ درصد ، در مقایسه با HSCGS ۷,۹۹ درصد ، در مقایسه با HEFT ۱۰,۷۳ درصد ، در مقایسه با MCP ۱۷,۷ درصد و در مقایسه با PEGA تا ۲۹,۳۳ بهبود می یابد . در مورد جریان های کاری Montage میانگین درصد افزایش کارآیی الگوریتم HGA در مقایسه با MPQGA مقدار ۵,۴۶ درصد و بقیه نتایج شبیه جریان کاری Cybershake می باشد . الگوریتم پیشنهادی ما بهتر از الگوریتم های فوق الذکر می باشد و کارآیی آن به صورت قابل توجهی بهتر از الگوریتم PEGA می باشد . در الگوریتم پیشنهاد شده ، هیوریستیک ، فرآیند جستجوی زمان بندی بهینه (makespan کمینه ) را تسریع می دهد و عملگرهای ژنتیک تغییر یافته به جستجوی کارای فضای مساله کمک شایانی می کند . این ویژگی ها الگوریتم HGA را در برابر سایر الگوریتم ها برتر می سازد .

### ۵.۳.جریان های کاری سنتزی

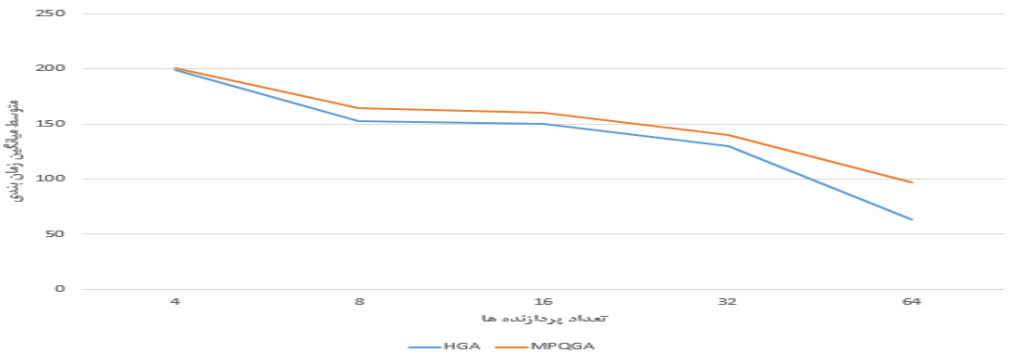
جریان های کاری سنتزی می توانند براساس پارامترهای ورودی به صورت تصادفی تولید شوند[21] و برای ارزیابی کارآیی الگوریتم HGA مورد استفاده قرار گیرند.

- اندازه جریان کاری (n) : پارامتر n تعداد گره های موجود در جریان کاری می باشد که نشان دهنده اندازه جریان کاری است . اندازه های مختلف جریان های کاری تصادفی با استفاده از مقادیر مختلف n در مجموعه ی n=*{20,40,60,80,100}* به دست می آید که در ارزیابی کارآیی HGA این جریان های کاری با اندازه متفاوت مورد استفاده قرار می گیرند .
- پارامتر شکل (∝) : شکل جریان کاری می تواند با پارامتر ∝ مدیریت شود اگر$\propto<1$ باشد آنگاه جریان های کاری طولانی با توازی کمتر تولید می شوند . اگر $\propto>1$ باشد آنگاه جریان های کاری کوچک با توازی بالا تولید می شوند . اگر $\propto=1$ باشد ، جریان های کاری با اندازه متوازن تولید می شوند که نه طولانی هستند و نه کوتاه . الگوریتم HGA به وسیله جریان کاری با ۳ اندازه متفاوت ( $\propto=\{0,1,2\}$ ) ارزیابی می شود.
- ارتباطات نسبت به نرخ محاسبات (CCR) : پارامتر CCR تعیین کننده این است که آیا جریان کاری محاسبات محور است یا ارتباط محور . اگر CCR>1 باشد آنگاه جریان کاری ارتباط محور است . اگر CCR<1 باشد آنگاه جریان کاری محاسبات-محور است . اگر CCR=1باشد آنگاه جریان کاری نه ارتباطاتی ونه محاسباتی است . مقادیر CCR در نظر گرفته شده در ارزیابی آزمایشات ۱۰و۱و۰,۱ می باشد .
- تعداد منابع معرفی شده با پارامتر P مشخص می شود ، ما تعداد منابع را به صورت توانی از ۲ $(2^X)$ در نظر می گیریم که درآن X={2,3,4,5,6} می باشد .
- درجه ی خروجی : این پارامتر تعیین کننده تعداد یال های خارج شده از یک گره می باشد . به این علت که جریان کاری تصادفی می باشد . درجه ی خروجی هم به صورت تصادفی انتخاب می شود .
- درصد محدوده ی هزینه های محاسباتی روی پردازنده ها ($\beta$) : برای تعیین فاکتور ناهمگنی سرعت پردازنده ار این پارامتر استفاده می شود . مقادیر بالاتر منجر به تفاوت قابل توجه در هینه های محاسباتی وظیفه ها می باشد و مقدار

پایین تر نشان دهنده ی کمتر بودن یا مساوی بودن هزینه های محاسباتی وظیفه ها می باشد . مقادیر $\beta$ استفاده شده در شبیه سازی ها ۰,۱ و ۰,۵ و ۱ می باشد .

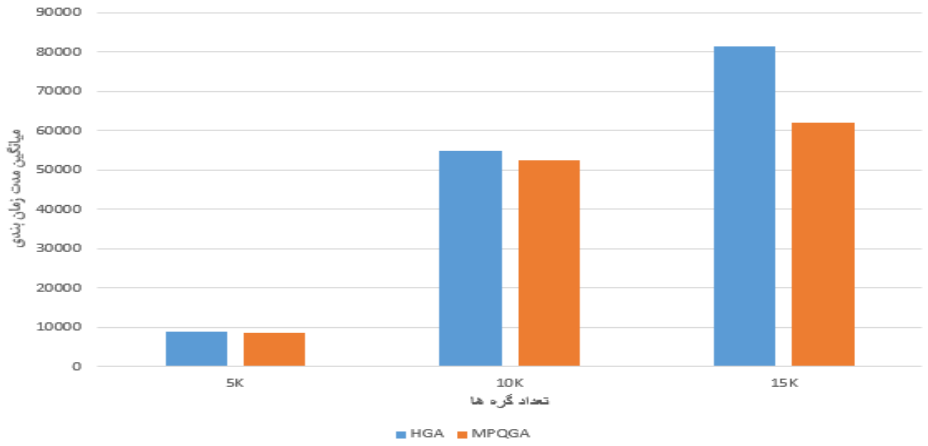



**شکل 6 . نتایج کارآیی با جریان کاری حذف گوسی به وسیله (الف)افزایش اندازه ماتریس (ب)افزایش تعداد پردازنده**

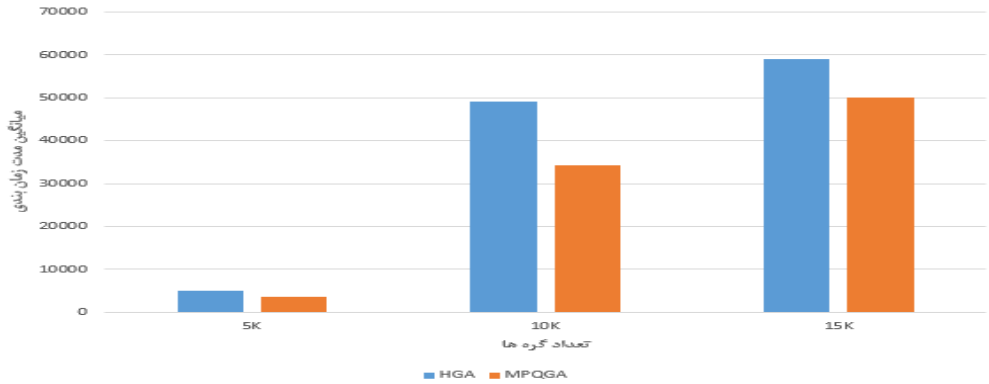



**شکل 7.کارآیی نتایج با جریان های کاری تصادفی به همراه افزایش تعداد پردازنده ها**

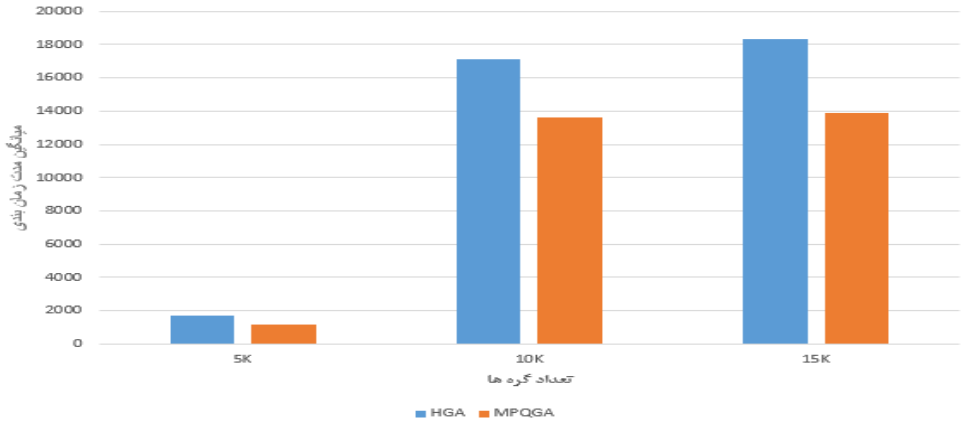
الف




ب

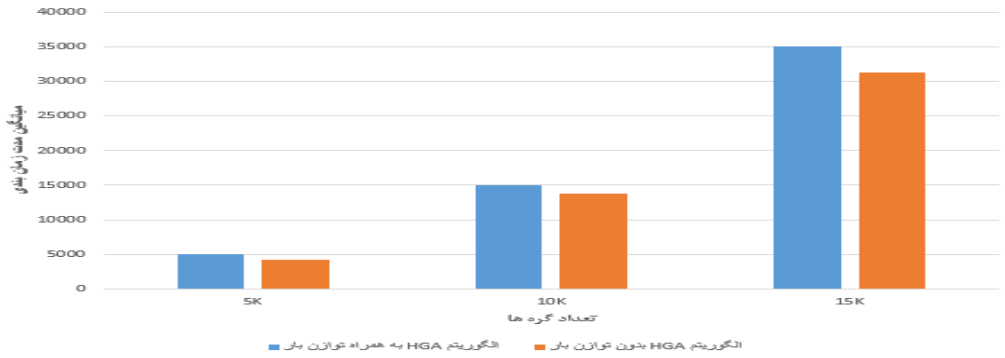



ج




**شکل 8. کارآیی جریان های کاری سنتزی برای مقدار ccr (الف)۰,۱ (ب)۱ (ج)۱۰**

جریان های کاری متنوع با ویژگی های متفاوت برای تحلیل کارآیی الگوریتم معرفی شده تولید شده اند . شکل ۱۰ نمایش دهنده رفتار الگوریتم ها برای جریان کاری با ۱۰۰ گره تصادفی است . زمانی که تعداد پردازنده ها افزایش می یابد ، بایستی توجه کنیم که همانگونه که منابع افزایش می یابد میانگین مدت زمان بندی کاهش می یابد تا به اندازه معینی برسد . این موضوع نشان می دهد که با افزایش توازی جریان های کاری زمان اجرا کاهش می یابد اما بر اساس محتوای متوالی (سریالی) جریان های کاری ، نرخ توازی محدود شده است . الگوریتم HGA به طورمیانگین ۵۰,۶ درصد نسبت به PEGA ، ۲۲,۷۶ درصد نسبت به HEFT و ۱۳ درصد نسبت به HSCGS و ۶,۷۸ درصد نسبت به MPQGA بهبود داشته است . مجموعه های جریان کاری 15K، 10K،5K با ویژگی های متفاوت به نحوی که در بالا اشاره شد تولید شده اند و نتایج شبیه سازی با مقادیر CCR ۰,۱ ، ۱ و ۱۰ به دست آمده است . نتایج در شکل ۱۱ نشان داده شده اند . الگوریتم HGA افزایش قابل توجهی نسبت به MPQGA ، HSCGS ، HEFT ، PEGA و MCP برای نتایج به دست آمده در رابطه با مقادیر CCR ۱ و ۱۰ به دست آمده است . کارآیی الگوریتم معرفی شده برای مقدار CCR ۰,۱ متقاعد کننده نمی باشد . این نشان دهنده این موضوع است که الگوریتم HGA برای جریان های کاری ارتباط محور به خوبی عمل می کند .




**شکل9.نتایج کارآیی الگوریتم HGA به همراه توازن بار و بدون توازن بار**

ویژگی توازن بار در الگوریتم معرفی شده با مقایسه نتایج به دست آمده از الگوریتم با توازن و بدون توازن مورد ارزیابی قرار گرفته است . نتایج در شکل ۱۲ آمده است . شکل نشان می دهد که الگوریتم HGA با متوازن ساز بار (ASL) میانگین مدت زمان بندی کمتری نسبت به الگوریتم HGA بدون توازن بار دارد. نتایج الگوریتم HGA نشان دهنده ی برتری این الگوریتم نسبت به ۵ الگوریتم دیگر است . پیچیدگی زمانی HGA ، $nm+n^2$ است که n تعداد نسل هاست و m اندازه جمعیت است . الگوریتم معرفی شده در مقایسه با هیوریستیک ها پیچیدگی بالاتری دارد . به این علت که الگوریتم معرفی شده برای زمان بندی ایستا طراحی شده اند بنابراین عیب پیچیدگی بالای HGA بر روی کارآیی سیستم تاثیری نخواهد داشت .

## ۶-نتیجه گیری

در این مقاله الگوریتم ژنتیک ترکیبی جدیدی (HGA) برای زمان بندی جریان کاری معرفی شد . الگوریتم معرفی شده از راه حل هیوریستیک زمان بندی کوچکترین زمان پایان ناهمگن (HEFT) به عنوان جمعیت اولیه استفاده می کند . استفاده از HEFT منجر به افزایش سرعت الگوریتم برای رسیدن به زمان بندی بهینه (makespan کمینه) با نسل های کمتر

می شود . جستجوی دقیق با عملگرهای جهش و ترکیب فضای بزرگ مساله را پوشش می دهد و کارآیی الگوریتم HGA را افزایش می دهد . الگوریتم معرفی شده مدت زمان بندی جریان های کاری را کاهش می دهد و بعلاوه دارای ویژگی توازن بار است که این ویزگی به کار گیری کارای منابع را ممکن می سازد. الگوریتم های زمان بندی که از روش های مختلفی استفاده می کنند با الگوریتم HGA مقایسه شده اند . شبیه سازی با مجموعه های داده با اندازه های متنوع و متفاوت نشان دهنده این است که الگوریتم پیشنهادی ما علاوه بر مقیاس پذیری عنصر تنوع را نیز دارا می باشد . نتایج اثبات می کند که الگوریتم HGA بهتر از سایر الگوریتم ها عمل می کند و کیفیت زمان بندی آن به دلیل کاهش مدت زمان بندی بهتر می باشد . شبیه سازی با توجه به نرخ CCR نشان داد که الگوریتم ما برای جریان های کاری با CCR بزرگتر از ۱ بهتر عمل می کند که این موضوع نشان دهنده این است که الگوریتم ما عملکرد بهتری برای جریان های کاری ارتباط-محور دارد .

در پروژه آینده قصد آزمایش کارآیی الگوریتم پیشنهادی با جریان های کاری پیچیده تر و بزرگتر را داریم که این جریان های کاری در محیطی با پردازنده های بیشتر و ناهمگن تر پردازش می شوند . بعلاوه ما قصد داریم بهینه سازی جریان های کاری داده محور را انجام دهیم . به این علت که برنامه های کاربردی علمی اغلب داده محور می باشند و جوامع علمی در تلاش برای حل چالش های بوجود آمده از داده های کلان می باشند .

**مراجع**


[1] S.G. Ahmad, E.U. Munir, M.W. Nisar, PEGA: A performance effective genetic algorithm for task scheduling in heterogeneous systems, in: IEEE 14th International Conference on High Performance Computing and Communications,2012, pp. 1082–1087.

[2] H. Arabnejad, J.G. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 682–694.

[3] M.P. Atkinson, R. Baxter, P. Besana, M. Galea, M. Parsons, P. Brezany, O.Corcho, J. van Hemert, D. Snelling, The DATA Bonanza—Improving Knowledge Discovery for Science, Engineering and Business, John Wiley & Sons, Inc., 2013.

[4] M.P. Atkinson, C.S. Liew, M. Galea, P. Martin, A. Krause, A. Mouat, O. Corcho,D. Snelling, Data-intensive architecture for scientific knowledge discovery,Distrib. Parallel Databases 30 (5–6) (2012) 307–324.

[5] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, Y. Simmhan, The tridentscientific workflow workbench, in: Proceedings of the 2008 Fourth IEEE International Conference on eScience, ESCIENCE ’08, IEEE Computer Society,Washington, DC, USA, 2008, pp. 317–318.

[6] M.R. Berthold, N. Cebron, F. Dill, T.R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel,B. Wiswedel, KNIME-the Konstanz information miner version 2.0 and beyond,SIGKDD Explor. Newsl. 11 (2009) 26–31.

[7] M. Caramia, S. Giordani, A fast metaheuristic for scheduling independent tasks with multiple modes, Comput. Ind. Eng. 58 (1) (2010) 64–69.

[8] A.L. Chervenak, D.E. Smith, W. Chen, E. Deelman, Integrating policy with scientific workflow management for data-intensive applications, in: 7th Workshop on Workflows in Support of Large-Scale Science, WORKS’12, 2012, pp. 140–149.

[9] M.I. Daoud, N.N. Kharma, A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous processor networks, J. Parallel Distrib. Comput. 71 (11) (2011) 1518–1531.

[10] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-Science: An overview of workflow system features and capabilities, Future Gener. Comput. Syst. 25 (5) (2009) 528–540.

[11] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, D.S. Katz, Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Sci. Program. J. 13 (3) (2005) 219–237.

[12] R.F.V. der Wijngaart, M. Frumkin, NAS Grid Benchmarks Version 1.0, Tech. Rep., NASA Advanced Supercomputing (NAS) Division, NASA Ames Reserach Center,Moffett Field, 2002, CA 94035-1000.

[13] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M.R. Pocock, P. Li, T. Oinn, Taverna: a tool for building and running workflows of services, Nucleic Acids Res. 34 (2) (2006) 729–732.

[14] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Gener. Comput. Syst. 29 (3) (2013) 682–692.

[15] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, Y. Zhao, Scientific workflow management and the Kepler system: Research articles, Concurr. Comput. Pract. Exp. 18 (2006) 1039–1065.

[16] E.U. Munir, S. Mohsin, A. Hussain, M.W. Nisar, S. Ali, SDBATS: A novel algorithm for task scheduling in heterogeneous computing systems, in: 2013 IEEE 27th International arallel and Distributed Processing Symposium Workshops Ph.D. Forum, IPDPSW, 2013, pp. 43–53.

[17] R.T. Neto, M.G. Filho, Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research, Eng. Appl. Artif. Intell. 26 (1) (2013) 150–161.

[18] Pegasus Team, Workflow Generator, 2013, https://confluence.pegasus.isi.edu/ display/pegasus/WorkflowGenerator.

[19] R. Salimi, H. Motameni, H. Omranpour, Task scheduling using NSGA II with fuzzy adaptive operators for computational grids, J. Parallel Distrib. Comput.74 (5) (2014) 2333–2350.

[20] J. Taheria, Y.C. Lee, A.Y. Zomaya, H.J. Siegel, A Bee Colony based optimization approach for simultaneous job scheduling and data replication in grid environments, Comput. Oper. Res. 40 (6) (2013) 1564–1578.

[21] H. Topcuoglu, S. Hariri, Min-You, Performance-effective and low complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.

[22] C. Wang, J. Gu, Y. Wang, T. Zhao, A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous multi-core system, in: Y. Xiang, I. Stojmenovic, B. Apduhan, G. Wang, K.

Nakano, A. Zomaya (Eds.), Algorithms and Architectures for Parallel Processing, in: Lecture Notes in Computer Science, vol. 7439, Springer, Berlin Heidelberg, 2012, pp. 153–170.

[23] M. Wilde, M. Hategan, J.M. Wozniak, B. Clifford, D.S. Katz, I. Foster, Swift: A language for distributed parallel scripting, Parallel Comput. 37 (9) (2011) 633–652.

[24] Y. Xu, K. Li, L. He, T.K. Truong, A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization, J. Parallel Distrib. Comput. 73 (9) (2013) 1306–1322.

[25] Y. Xu, K. Li, J. Hu, K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, Inf. Sci. 270 (2014) 255–287.

[26] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for grid computing, SIGMOD Rec. 34 (2005) 44–49.