چهاردهمین کنفرانس ملی سالانه انجمن کامپیوتر ایران
دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران) ایران، تهران – ۲۰ و ۲۱ اسفندماه ۱۳۸۷

زمان‌بندی بار محاسباتی تقسیم‌پذیر با در نظرگرفتن زمان بازگشت نتایج در سیستم‌های ناهمگن با استفاده از الگوریتم ژنتیک

رضا منصفی،

عضو هیئت علمی دانشگاه،

دانشگاه فردوسی مشهد، دانشکده مهندسی، گروه کامپیوتر

monsefi@um.ac.ir

جواد حمیدزاده،

دانشجوی دکترای کامپیوتر،

دانشگاه فردوسی مشهد، دانشکده مهندسی، گروه کامپیوتر و عضو هیئت علمی مؤسسه آموزش عالی سجاد مشهد

Ja_ha47@stu-mail.um.ac.ir

چکیده: امروزه مسئله زمان‌بندی کارها در سیستم‌های ناهمگن به دلیل لزوم استفاده بهینه از ماشین‌های محاسباتی موجود و همچنین صرف زمان کمتر برای اجرای الگوریتم‌های زمان‌بندی، از اهمیت خاصی برخوردار است. در این مقاله زمان‌بندی بار محاسباتی تقسیم‌پذیر با درنظرگرفتن زمان بازگشت نتایج در یک سیستم ناهمگن دارای شبکه ارتباطی درختی تک‌سطحی بررسی شده است. یکی از اهداف زمان‌بندی در این‌گونه سیستم‌ها، کمینه‌سازی زمان کل پاسخ است. تاکنون الگوریتمی معین با پیچیدگی زمانی چند جمله‌ای که بتواند در تمام حالت‌ها جواب بهینه را تولید کند، برای این منظور ارائه نشده است. این مسئله مانند مسائل ترکیباتی، پیچیده به نظر می‌رسد و راه حل‌های موجود برای آن، راه حل‌های ابتکاری است. در این مقاله الگوریتم ژنتیک به عنوان یک راه حل مسئله پیشنهاد شده است. با انجام شبیه‌سازی و مقایسه نتایج مشاهده می‌شود که این راه حل، در مقایسه با سایر روش‌های موجود جواب‌های بهتری تولید می‌کند. در میان روش‌های موجود، الگوریتم ژنتیک پیشنهادی دارای کمترین میانگین کل درصد خطای نسبی است.

واژه‌های کلیدی: زمان‌بندی، بار محاسباتی تقسیم‌پذیر، زمان بازگشت نتایج، سیستم‌های ناهمگن، الگوریتم ژنتیک.

۱- مقدمه

امروزه کارهای[۱] محاسباتی حجیم و زمان‌بندی اجرای آنها در سیستم‌های ناهمگن[۲] بسیار مورد توجه قرار گرفته است. در این گونه سیستم‌ها، کارایی ماشین‌ها، هم‌بندی شبکه ارتباطی و سرعت خطوط ارتباطی[۳] شبکه می‌تواند متفاوت باشد بطوری که یک سیستم ناهمگن تشکیل شود. مسئله تطبیق و زمان‌بندی کارها در این سیستم‌ها، از جمله مسائل سخت[۴] است. یکی از مدل‌های محاسباتی مطرح در این سیستم‌ها، مدل بار محاسباتی تقسیم‌پذیر[۵] است. تئوری این دسته از محاسبات برای اولین بار در سال ۱۹۸۸ میلادی توسط آقای «ربرتازی» در مقاله [۱۲] معرفی شد که با چاپ کتاب [۱] توسط ایشان و همکاران، باعث توجه بیشتر محققین به این تئوری شد.

در تئوری بار محاسباتی تقسیم‌پذیر، بار محاسباتی از تعداد زیادی واحدهای محاسباتی[۶] ریزدانه[۷] تشکیل شده است که پردازش این واحدها می‌تواند به طور کاملاً مستقل و موازی با هم انجام گیرد. فرض بر این است که این واحدهای محاسباتی قابل تقسیم نباشند. درنتیجه، حجم زیادی از محاسبات، می‌تواند به هر نسبتی برحسب واحدهای محاسباتی ریزدانه تقسیم شده و سپس برای اجرای موازی به ماشین‌های موجود ارسال شود.

این تئوری ابزار مناسبی جهت مدل‌سازی کاربردهای بسیاری با داده‌های حجیم[۸] است [۲۲و۸-۶]. از جمله کاربردهایی که در این دسته از محاسبات می‌توان به آنها اشاره نمود عبارتند از: پردازش تصویر[۱۳]، پردازش سیگنال، جستجو در بانک‌های اطلاعاتی[۱۴]، داده‌کاوی[۹]، محاسبات جبرخطی [۱۵] و کاربردهای چند رسانه‌ای[۱۶].

در این تئوری، از مدل ارباب و کارگر[۱۰] جهت پیاده‌سازی استفاده می‌شود. کل بار محاسباتی در این تئوری بر روی کامپیوتر ارباب قرار دارد که از طریق یک شبکه ارتباطی دارای هم‌بندی[۱۱] مشخص، با کامپیوترهای کارگر در ارتباط است. کامپیوتر ارباب موظف است بار محاسباتی تقسیم‌پذیر را بین کامپیوترهای کارگر تقسیم نماید و کامپیوترهای کارگر تا دریافت کامل سهم بار

محاسباتی خود منتظر مانده و سپس پردازش خود را آغاز نمایند. هر کامپیوتر کارگر پس از اتمام پردازش، موظف است طی ترتیبی که در زمان‌بندی مشخص شده نتیجه را به کامپیوتر ارباب بازگرداند. هدف از زمان‌بندی در این سیستم، تعیین سهم هر کامپیوتر کارگر، ترتیب توزیع سهم‌ها و ترتیب بازگشت نتایج پردازش شده به کامپیوتر ارباب است، به‌گونه‌ای که زمان کل پاسخ[۱۲] کمینه شود.

در اغلب تحقیقات انجام شده در این زمینه، حجم نتایج حاصل از پردازش در کامپیوترهای کارگر بقدری کم فرض شده است که می‌توان از در نظرگرفتن تاخیر زمانی لازم جهت ارسال این داده‌ها به کامپیوتر ارباب صرفنظر نمود. این در حالی است که می‌توان کاربردهای بسیاری را در نظر گرفت که در آنها، حجم نتایج بدست آمده در کامپیوترهای کارگر، ضریب مشخصی از حجم بار تحویلی به آنهاست. بنابراین در این گونه موارد نمی‌توان از در نظرگرفتن تاخیر زمانی مورد نیاز جهت بازگشت نتایج به کامپیوتر ارباب صرفنظر نمود. متأسفانه تحقیقات اندکی در این زمینه انجام شده و نتایج بدست آمده در این باره، بسیار محدود است.

امروزه تحقیقات در این زمینه با در نظرگرفتن تاخیر زمانی برای بازگشت نتیجه پردازش‌ها مطرح گردیده است، به‌این ترتیب که اگر m، تعداد کامپیوترهای کارگر باشد، با در نظرگرفتن ترتیب‌های مختلف جهت توزیع و بازگشت نتیجه، پیچیدگی زمانی راه حل بهینه، از مرتبه $O(m!^2)$ خواهد شد و چنانچه تعداد کامپیوترهای کارگر زیاد باشد، بدست آوردن زمان‌بندی بهینه، بسیار زمانبر خواهد بود. تاکنون نیز راه حلی بهینه با پیچیدگی زمانی چند جمله‌ای برای این مسئله ارائه نشده است. اما روش‌های ابتکاری موجود برای حل این مسئله عبارتند از: روش‌های LIFOC[۱۳]، FIFOC[۱۴] [۱۸و۲۱]، ITERLP[۱۵] [۲۰] و SPORT[۱۶] [۹].

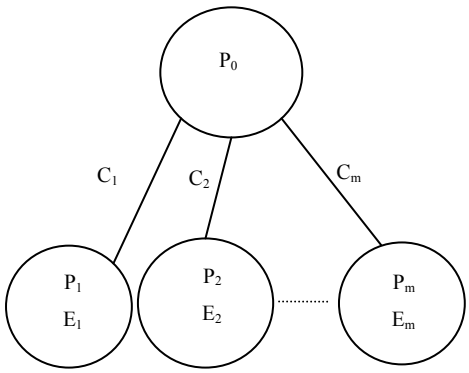
نتایج تحقیقات، نشان دهنده این است که در یک سیستم همگن[۱۷] روش FIFOC عملکرد بهتری نسبت به روش LIFOC داشته و به جواب بهینه نزدیک‌تر است ولی در یک سیستم ناهمگن، عکس این مسئله صادق است، یعنی عملکرد روش LIFOC بهتر از روش FIFOC است [۱۸و۱۰]. آخرین روش قابل بررسی در این زمینه، روش SPORT است که در مقالات [۹] و [۱۰] معرفی و تحلیل گردیده است.

یکی از روش‌های ابتکاری هوشمند در حل این مسئله، استفاده از روش‌های محاسبات تکاملی، از جمله الگوریتم ژنتیک است. تنها الگوریتم ژنتیک ارائه شده در این زمینه الگوریتم[۱۹] است که در آن، تاخیر زمانی بازگشت نتیجه در نظر گرفته نشده است. هدف ما در این مقاله، پیشنهاد یک الگوریتم ژنتیک برای حل این مسئله است که در آن، تاخیر زمانی بازگشت نتیجه درنظر گرفته شود.

ادامه این مقاله از بخش‌های زیر تشکیل شده است. در بخش دوم مسئله زمان‌بندی بار تقسیم‌پذیر در یک شبکه دارای هم‌بندی درختی تک سطحی بیان شده و مدل ریاضی برای آن ارائه گردیده است. در بخش سوم، الگوریتم ژنتیک جهت این زمان‌بندی بیان شده است. در بخش چهارم، نتایج شبیه‌سازی‌های انجام شده و مقایسه‌ها بیان گردیده است و در نهایت، در بخش پایانی نتیجه‌گیری و پیشنهاد ادامه کار بیان شده است.

۲- بیان مدل مسئله زمان‌بندی بار تقسیم‌پذیر

در شکل (۱)، هم‌بندی درختی تک سطحی مورد استفاده در شبکه ارتباطی مدل ارباب و کارگر نشان داده شده است. دراین شکل، کامپیوتر ارباب در ریشه و کامپیوترهای کارگر در برگ‌ها واقع شده‌اند. در ابتدا کل بار محاسباتی L بر روی کامپیوتر ارباب ذخیره گردیده است.



شکل (۱): شبکه درختی تک سطحی ناهمگن [۹]

در این مدل، $P=\{P_0,\cdots,P_m\}$ بیانگر مجموعه m+1 کامپیوتر است که در آن P_0 کامپیوتر ارباب و بقیه، کامپیوترهای کارگر هستند؛ $E=\{E_1,\cdots,E_m\}$ بیانگر شاخص سرعت پردازش کامپیوترهای کارگر است؛ E_k بیانگر مدت زمان لازم جهت پردازش یک واحد محاسباتی بر روی کامپیوتر کارگر شماره k است؛ مجموعه $C=\{C_1,\cdots,C_m\}$ بیانگر پارامترهای مربوط به سرعت خطوط ارتباطی شبکه است؛ C_k بیانگر مدت زمان لازم جهت ارسال یک واحد داده محاسباتی از کامپیوتر ارباب به کامپیوتر کارگر شماره k است. دراین مدل، L بیانگر کل حجم بار محاسباتی اولیه موجود در کامپیوتر ارباب است. همچنین با توجه به اینکه به کلیت مسئله لطمه‌ای وارد نخواهد آمد، فرض می‌کنیم L=1 باشد.

در این مدل، $\alpha=\{\alpha_1,\cdots,\alpha_m\}$ بیانگر درصد سهم بار محاسباتی هر یک از کامپیوترهای کارگر است؛ δ نیز بیانگر نسبت حجم داده حاصل از پردازش در هر کامپیوتر کارگر به سهم بار محاسباتی آن است. این پارامتر یک مقدار ثابت است که به ماهیت برنامه کاربردی[۱۸] وابسته بوده و به میزان سهم هر کامپیوتر کارگر بستگی

ندارد. همچنین فرض می‌کنیم که $0\leq\delta\leq1$ باشد.

حال اگر سهم کامپیوتر کارگر شماره i، α_i باشد، زمان لازم برای ارسال این میزان بارمحاسباتی از کامپیوتر ارباب به کامپیوتر کارگر شماره i برابر است با: $\alpha_i\times C_i$؛ زمان لازم برای پردازش این سهم در کامپیوتر کارگر شماره i برابر است با: $\alpha_i\times E_i$؛ زمان لازم جهت بازگشت نتیجه از کامپیوتر کارگر شماره i به کامپیوتر ارباب نیز برابر است با: $\delta\times\alpha_i\times C_i$ [۹].

دو مجموعه σ_a و σ_c، به‌ترتیب بیانگر ترتیب توزیع بارمحاسباتی به کامپیوترهای کارگر و ترتیب بازگشت نتایج از آنها به کامپیوتر ارباب است. در واقع اولین مجموعه، ترتیب تخصیص بار[۱۹] و دومین مجموعه، بیانگر ترتیب جمع آوری نتایج[۲۰] است؛ $\sigma_a[k]$ بیانگر شماره کامپیوتر کارگری است که به عنوان kاُمین کامپیوتر جهت ارسال یا تخصیص بار از سوی کامپیوتر ارباب انتخاب شده است؛ $\sigma_c[k]$ بیانگر شماره کامپیوتر کارگری است که به عنوان kاُمین کامپیوتر، نتیجه را به کامپیوتر ارباب برمی‌گرداند؛ $\sigma_a(k)$ بیانگر اندیس محل کامپیوتر کارگر شماره k در ترتیب تخصیص بار محاسباتی است؛ $\sigma_c(k)$ بیانگر اندیس محل کامپیوتر کارگر شماره k در ترتیب بازگشت نتایج است. در این مدل فرض براین است که قبل از زمان‌بندی، مقادیر مجموعه‌های C، E و پارامترهای m و δ مشخص هستند.

در اینجا هدف از زمان‌بندی، تعیین ترتیب‌های بهینه تخصیص و جمع‌آوری نتایج و میزان سهم هر کامپیوتر کارگر از بار محاسباتی است. به عبارت دیگر، هدف از زمان‌بندی، تعیین مجموعه‌های σ_a، σ_c و α است به‌گونه‌ای که زمان کل پردازش کمینه شود. زمان کل پردازش عبارت است از زمان شروع توزیع بار محاسباتی تا زمان دریافت آخرین نتیجه پردازش شده توسط کامپیوتر ارباب.

در صورتی‌که دو مجموعه σ_a و σ_c مشخص باشند، مجموعه α را می‌توان با حل برنامه ریزی خطی ارائه شده در شکل (۲) محاسبه کرد. در شکل(۲)، متغیر T، بیانگر زمان کل پاسخ است که قرار است کمینه شود؛ روابط (۱) تا (۴)، بیانگر محدودیت‌های مسئله زمان‌بندی هستند؛ بخش اول محدودیت شماره (۱) بیانگر زمان لازم جهت تخصیص بار محاسباتی کامپیوترهای کارگر یکم تا kاُم، طبق ترتیب σ_a است؛ بخش دوم محدودیت شماره (۱) بیانگر مدت زمان لازم جهت پردازش در کامپیوتر kاُم است؛ بخش سوم محدودیت مزبور نیز در برگیرنده مدت زمان لازم جهت بازگشت نتایج بدست آمده از کامپیوترهای کارگر شماره k تا m، طبق ترتیب σ_c است. محدودیت شماره (۲) بیانگر جمع زمان‌های لازم برای تخصیص کل بار محاسباتی به کامپیوترهای کارگر و جمع‌آوری تمامی نتایج از آنها است. محدودیت شماره (۳) نیز یک رابطه نرمال‌سازی به این منظور است که تخصیص کل بار محاسباتی را به کامپیوترهای کارگر تضمین کند.

مسئله کمینه‌سازی نشان داده شده در شکل (۲) با در نظرگرفتن محدودیت‌های (۱) تا (۴)، یک مسئله برنامه‌ریزی خطی است که می‌تواند به کمک روش‌های استاندارد برنامه‌ریزی خطی، با پیچیدگی زمانی چند جمله‌ای حل شود [۲].

هدف ما از به کارگیری الگوریتم ژنتیک، تعیین ترتیب‌های بهینه σ_a و σ_c است. برای حل این مسئله در حالت معمول براساس روش «جستجوی همه حالت‌ها[۲۱]»، نیاز به بررسی تمامی جایگشت‌های دو مجموعه σ_a و σ_c است که برابر است با: $m!\times m!=m!^2$. این روش که جواب بهینه سراسری را تولید می‌کند، در این مقاله الگوریتم بهینه نامیده می‌شود.

Minimize T

Subject To:

$$\sum_{j=1}^{\sigma_a(k)} \alpha_{\sigma_a[j]}C_{\sigma_a[j]} + \alpha_k E_k + \sum_{j=k}^{m} \delta\alpha_{\sigma_c[j]}C_{\sigma_c[j]} \leq T$$

$$k=1,\cdots,m \quad (۱)$$

$$\sum_{j=1}^{m} \alpha_{\sigma_a[j]}C_{\sigma_a[j]} + \sum_{j=1}^{m} \delta\alpha_{\sigma_c[j]}C_{\sigma_c[j]} \leq T \quad (۲)$$

$$\sum_{j=1}^{m} \alpha_j = 1 \quad (۳)$$

$$T\geq0,\ \alpha_k\geq0 \qquad k=1,\cdots,m \quad (۴)$$

شکل(۲): مسئله برنامه‌ریزی خطی [۹]

در حال حاضر الگوریتم معینی برای حل این مسئله وجود ندارد که بتواند در زمان چند جمله‌ای، جواب بهینه را بیابد. در این زمینه فقط راه حل‌های ابتکاری یا حریصانه[۲۲] وجود دارد. روش‌های LIFOC، FIFOC، ITERLP و SPORT نیز جزو راه حل‌های ارائه شده و مطرح هستند [۹]. الگوریتم ژنتیک پیشنهادی برای حل مسئله زمان‌بندی فوق، در مقایسه با روش‌های مطرح توانسته است جواب‌های بهتر و نزدیک‌تری به جواب بهینه تولید کند که نتایج حاصل از اجرای این الگوریتم و مقایسه آنها با جواب‌ها در سایر روش‌ها، در بخش شبیه‌سازی این مقاله بیان شده است.

۳- الگوریتم ژنتیک برای بهینه‌سازی زمان‌بندی بار محاسباتی تقسیم‌پذیر

الگوریتم‌های ژنتیک، تکنیک‌های جستجوی تصادفی بر پایه انتخاب و تکامل طبیعی هستند [۳و۵]. برای حل یک مسئله، این الگوریتم‌ها، با یک مجموعه از راه حل‌های تصادفی آغاز می‌گردند که جمعیت اولیه

نامیده می‌شود. هر عضو این جمعیت، یک کروموزوم (فرد) نامیده می‌شود. هر کروموزوم نشان دهنده یک راه حل مورد بررسی است و شامل دو بخش از ژن‌ها است. یک بخش، بیانگر ترتیب σ_a و بخش دیگر بیانگر ترتیب σ_c است. تعداد ژن‌های هر بخش برابر m است که در این مسئله، برابر است با تعداد کامپیوترهای کارگر.

مجموعه کروموزوم‌های مورد بررسی در هر بار تکرار الگوریتم ژنتیک، یک نسل نامیده می‌شود. کروموزوم‌ها توسط تابع برازندگی[۲۳] مورد ارزیابی قرار گرفته و به‌منظور تولید نسل بعد، کروموزوم‌های جدید (فرزندان) با اِعمال عملگرهای ژنتیک ترکیب[۲۴] و جهش[۲۵] تولید می‌شوند. سپس با استفاده از عملگر تکاملی انتخاب[۲۶] و با توجه به مقادیر تابع برازندگی کروموزوم‌ها، تعدادی از والدین و فرزندان به نسل بعد انتقال می‌یابند. لازم به ذکراست که ما از نخبه‌گرایی[۲۷] نیز استفاده کرده‌ایم. بنابراین بدترین فرد نسل بعد، با بهترین فرد نسل قبل جایگزین می‌شود.

مراحل سه گانه تولید، ارزیابی و انتخاب کروموزوم‌ها، تا مشاهده شرایط توقف، تکرار می‌شوند. با پایان یافتن الگوریتم، کروموزوم دارای بهترین مقدار تابع برازندگی، بیانگر مجموعه‌های σ_a، σ_c و α خواهد بود. عکس مقدار تابع برازندگی، برابر زمان T است.

در ادامه، عملگرهای انتخاب، ترکیب و جهش و تابع برازندگی مورد استفاده در الگوریتم ژنتیک پیشنهادی بطور دقیق‌تر تشریح خواهد شد.

عملگر انتخاب، یکی از عملگرهای حیاتی و مهم در الگوریتم ژنتیک است. عملگر انتخاب یک عملگر احتمالی است که در آن شانس انتخاب بالاتر به افرادی داده می‌شود که از میزان برازندگی بیشتری برخوردارند. عملگر انتخاب مورد استفاده در الگوریتم ژنتیک پیشنهادی، بر اساس روش رتبه‌دهی با توزیع هندسی نرمال شده است [۱۱]. مزیت استفاده از این روش، جلوگیری از هم‌گرایی سریع در نسل‌های متوالی الگوریتم ژنتیک است. در این روش افراد بر اساس میزان برازندگی‌شان به‌ترتیب نزولی مرتب می‌شوند. سپس رتبه‌دهی به آنها بصورت یک لیست مرتب با شروع از عدد ۱ صورت می‌پذیرد به‌گونه‌ای که بهترین فرد، رتبه ۱ و بدترین فرد، رتبه N را کسب خواهد نمود. در لیست مرتب، برای فرد شماره j رتبه $R_j=j$ در نظر گرفته می‌شود که براساس آن احتمال انتخاب S_j برای فرد شماره j از رابطه (۱) محاسبه می‌شود.

$$S_j = q(1-p)^{R_j-1} \qquad (۱)$$

در رابطه (۱)، p معرف احتمال انتخاب بهترین فرد است. لازم به ذکر است که p جزو پارامترهای الگوریتم ژنتیک پیشنهادی است. در رابطه (۱) مقدار q مطابق رابطه (۲) محاسبه می‌شود.

$$q = \frac{p}{1-(1-p)^N} \qquad (۲)$$

در رابطه (۲)، پارامتر N، معرف اندازه جمعیت است. به راحتی می‌توان درستی رابطه (۳) را تحقیق نمود.

$$\sum_{j=1}^{N} S_j = 1 \qquad (۳)$$

عملگر ترکیب مورد استفاده، عملگر «ترکیب نگاشت شده جزئی[۲۸]» است [۱۷] که در آن بطور تصادفی محل دو ژن از دو کروموزوم، انتخاب شده، سپس ژن‌های بین این دو محل از دو کروموزوم، با یکدیگر تعویض می‌شود. در ادامه برای حذف ژن‌های تکراری در هر دو کروموزوم کافی است محل‌های حاوی ژن‌های تکراری از دو کروموزوم را در نظر گرفته، سپس ژن‌های محل‌های مشابه در دو کروموزوم، با یکدیگر تعویض شوند.

در الگوریتم ژنتیک، به‌منظور یافتن نقاط جدید در فضای جستجو به‌گونه‌ای که تنوع جمعیتی نیز حفظ شود و همچنین به دلیل خارج شدن از دام بیشینه‌های محلی، از عملگر جهش استفاده می‌شود. در الگوریتم ژنتیک پیشنهادی به‌منظور جهش یک کروموزوم، ابتدا بطور تصادفی محل دو ژن، انتخاب شده و سپس ژن‌های این دو محل با یکدیگر تعویض می‌شوند[۴].

بطور کلی الگوریتم ژنتیک، به دنبال یافتن نقطه بهینه بیشینه است ولی هدف روش پیشنهادی در زمان‌بندی، یافتن نقطه بهینه کمینه است. در نتیجه تابع برازندگی را به صورت رابطه (۴) در نظر گرفته‌ایم.

$$F=\frac{1}{T} \qquad (۴)$$

۴- نتایج شبیه‌سازی و مقایسه‌ها

در آزمایش‌های انجام شده، کارایی الگوریتم ژنتیک را به کمک الگوریتم بهینه، با چهار الگوریتم SPORT، LIFOC، ITERLP و FIFOC مقایسه کرده‌ایم. پیاده‌سازی‌ها، در محیط MatLab و بر روی کامپیوتری با پردازنده Intel Core 2 Due 2.4 GHz و دارای یک گیگا بایت حافظه اصلی انجام شده است. مقادیر پارامترهای C و E جهت انجام آزمایش‌ها، برای نمایش یک سیستم ناهمگن، مشابه روش ارائه شده در مقاله [۹] تولید شده‌اند که در آن ۲۵ مورد مختلف طبق جدول شماره (۱) در نظر گرفته شده است.

جدول(۱): محدوده پارامترهای سیستم ناهمگن [۹]

Case	C_k	E_k	Case	C_k	E_k
1	[1,10]	[1,10]	14	[10,100]	[1,100]
2	[1,10]	[10,100]	15	[10,100]	[10,1000]
3	[1,10]	[100,1000]	16	[10,1000]	[1,10]
4	[1,10]	[1,100]	17	[10,1000]	[10,100]
5	[1,10]	[10,1000]	18	[10,1000]	[100,1000]
6	[1,100]	[1,10]	19	[10,1000]	[1,100]
7	[1,100]	[10,100]	20	[10,1000]	[10,1000]
8	[1,100]	[100,1000]	21	[100,1000]	[1,10]
9	[1,100]	[1,100]	22	[100,1000]	[10,100]
10	[1,100]	[10,1000]	23	[100,1000]	[100,1000]
11	[10,100]	[1,10]	24	[100,1000]	[1,100]
12	[10,100]	[10,100]	25	[100,1000]	[10,1000]
13	[10,100]	[100,1000]			

برای هر یک از ۲۵ مورد جدول (۱)، m مقدار برای پارامترهای C و E، در محدوده‌های تعیین شده، بصورت تصادفی و با توزیع یکنواخت تولید شده‌اند. با توجه به اینکه الگوریتم ژنتیک یک الگوریتم تصادفی است، برای هر مورد فوق، تولید پارامترهای C و E، صد بار تکرار شده است که منجر به صد بار اجرای الگوریتم ژنتیک و سایر الگوریتم‌ها شده است. برای هر کدام از صد نمونه فوق، اجرای الگوریتم ژنتیک، ده مرتبه تکرار شده است و نتیجه (T) در نظر گرفته شده برای الگوریتم ژنتیک، میانگین این ده مرتبه از اجرا است. مقادیر پارامترهای الگوریتم ژنتیک در جدول شماره (۲) نشان داده شده است.

جدول(۲): مقادیر پارامترهای الگوریتم ژنتیک

نام پارامتر	وظیفه	مقدار
P_m	احتمال جهش	0.05
P_c	احتمال ترکیب	0.6
MaxGen	ماکزیمم تعداد نسل‌ها	100
q	احتمال انتخاب بهترین فرد	0.08
N	اندازه جمعیت	10

در تمام آزمایش‌ها، زمان اتمام پردازش کل بار محاسباتی، برای هر الگوریتم محاسبه شده است. اگر متغیر T_{opt} بیانگر این زمان برای الگوریتم بهینه باشد و متغیر T_v نیز بیانگر زمان فوق‌الذکر برای سایر الگوریتم‌ها باشد آنگاه درصد خطای نسبی (ΔT_v) طبق رابطه (۵) محاسبه می‌شود.

$$\Delta T_v = \frac{T_v - T_{opt}}{T_{opt}} \times 100\% \qquad (۵)$$

با توجه به اینکه برای هر مورد از جدول شماره (۱)، صد داده آزمایشی تولید شده است، میانگین رابطه (۵)، بصورت فرمول (۶) محاسبه می‌گردد.

$$\overline{\Delta T_v} = \frac{\sum_{k=1}^{100} \Delta T_v^k}{100} \qquad (۶)$$

فرمول (۶) را میانگین درصد خطای نسبی می‌نامیم.

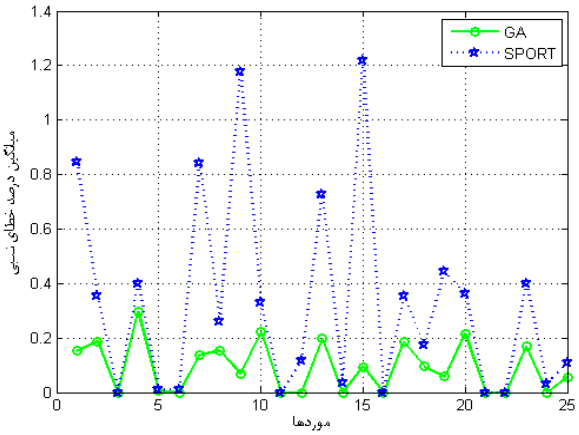
در شکل (۳)، میانگین درصد خطای نسبی برای دو الگوریتم ژنتیک و SPORT نشان داده شده است. همان‌گونه که در نمودار دیده می‌شود، حداکثر میزان خطای نسبی الگوریتم ژنتیک، کمتر از ۰/۴ درصد است در حالی که این مقدار برای الگوریتم SPORT بیش‌از ۱ درصد است. همچنین به‌طور متوسط، الگوریتم ژنتیک نسبت به الگوریتم دیگر جواب‌های نزدیک‌تری به جواب بهینه تولید کرده است.

در شکل (۴)، الگوریتم ژنتیک با الگوریتم LIFOC مقایسه شده است. با دقت در این نمودار در می‌یابیم که حداکثر خطای الگوریتم ژنتیک از حداکثر خطای الگوریتم LIFOC کمتر است. حداکثر خطای الگوریتم LIFOC بیش‌از ۰/۵ درصد است.

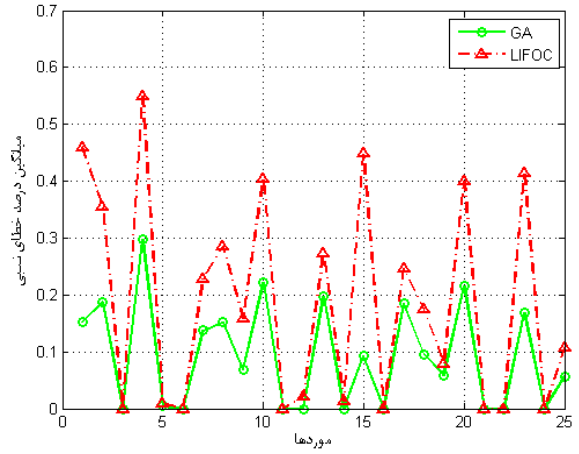
در شکل (۵)، الگوریتم ژنتیک را با الگوریتم ITERLP مقایسه کرده‌ایم. همان‌گونه که این نمودار نشان می‌دهد، در اکثر موردها، الگوریتم ژنتیک نتایج بهتری نسبت به الگوریتم ITERLP تولید کرده است. همچنین حداکثر خطای نسبی الگوریتم ITERLP، سه برابر حداکثر خطای نسبی الگوریتم ژنتیک است.

با توجه به نتایج ضعیفی که الگوریتم FIFOC در آزمایش‌ها، برای سیستم‌های ناهمگن تولید کرده است، از ترسیم نمودار مقایسه با این الگوریتم صرفنظر کرده و فقط نتیجه مقایسه در شکل (۸) و جدول شماره (۳) نشان داده شده است.

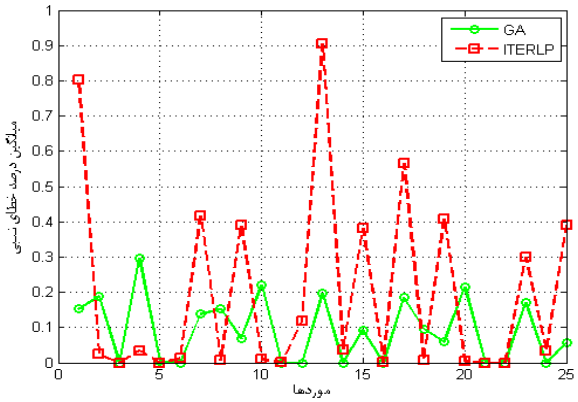
همان‌طور که این نمودارها نشان می‌دهند، در اکثر موارد الگوریتم ژنتیک نتیجه بهتری را تولید می‌کند، به قسمی که حداکثر خطای آن نسبت به روش بهینه کمتر از ۰/۴ درصد است.



شکل(۳): نمودار میانگین درصد خطای نسبی دو الگوریتم ژنتیک و SPORT برای m=5, δ=0.5



شکل(۴): نمودار میانگین درصد خطای نسبی دو الگوریتم ژنتیک و LIFOC برای m=5, δ=0.5

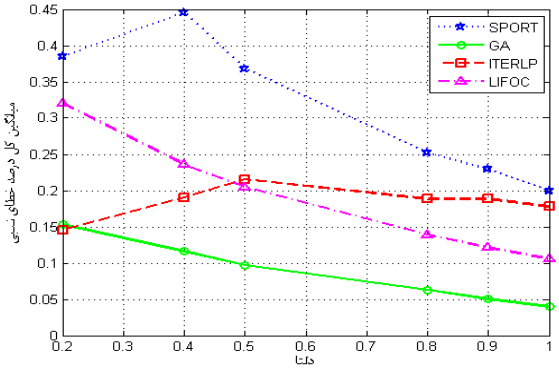


شکل(۵): نمودار میانگین درصد خطای نسبی دو الگوریتم ژنتیک و ITERLP برای m=5, δ=0.5

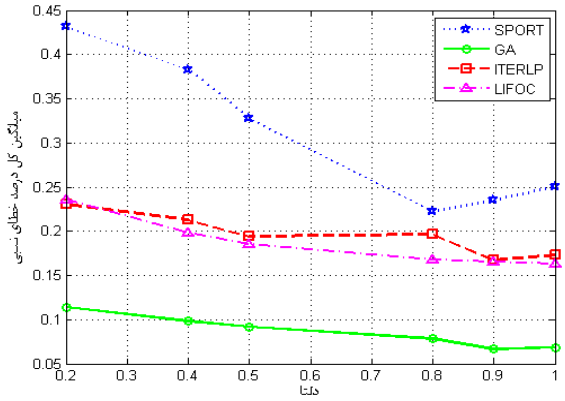
به‌منظور در نظرگرفتن میزان تاثیر پارامتر دلتا بر روی زمان پاسخ‌های بدست آمده در الگوریتم‌های ذکر شده، آزمایش‌های قبلی را برای m=4,5 و δ=0.2,0.4,0.5,0.8,1 تکرار کرده و سپس میانگین مقدار $\overline{\Delta T_v}$ را برای ۲۵ مورد محاسبه نموده‌ایم که حاصل آن در نمودارهای شماره (۶) و (۷) نشان داده شده است. رابطه (۷) را میانگین کل درصد خطای نسبی نامیده‌ایم.

$$\overline{\overline{\Delta T_v}} = \frac{\sum_{i=1}^{25} \overline{\Delta T_v^i}}{25} \qquad (۷)$$

در نمودارهای (۶) و (۷)، میانگین کل درصد خطای نسبی برای چهارالگوریتم ژنتیک، LIFOC، ITERLP و SPORT و به‌ترتیب برای چهار و پنج کامپیوتر کارگر ترسیم شده است. همان طور که در نمودارهای فوق الذکر دیده می‌شود با افزایش مقدار پارامتر دلتا، میانگین کل درصد خطای نسبی الگوریتم ژنتیک کاهش می‌یابد. برای تمام مقادیر دلتا، میانگین کل درصد خطای نسبی الگوریتم ژنتیک در مقایسه با سایر الگوریتم‌ها کمتر است. همچنین الگوریتم ژنتیک، زمان پاسخ بهتری را با افزایش مقدار پارامتر دلتا تولید می‌کند.

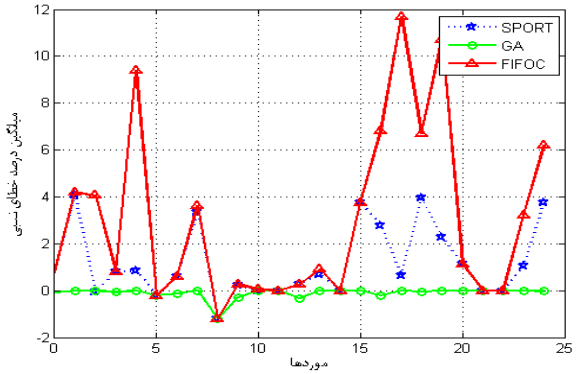


شکل(۶): نمودار میانگین کل درصد خطای نسبی برای m=4



شکل(۷): نمودار میانگین کل درصد خطای نسبی برای m=5

به‌منظور در نظرگرفتن تاثیر افزایش تعداد کامپیوترهای کارگر(m) نیز در نتایج بدست آمده، آزمایش‌های قبلی برای مقادیر m=100 و δ=0.5 تکرار شده است ولی به این دلیل که امکان اجرای الگوریتم بهینه و ITERLP از نظر زمانی وجود نداشته، اختلاف نتایج بدست آمده، با الگوریتم LIFOC مقایسه شده است که نتایج در نمودار شکل (۸)، نشان داده شده است.



شکل(۸): نمودار میانگین درصد خطا نسبت به الگوریتم LIFOC برای m=100, δ=0.5

در نمودار شکل (۸)، بخش‌هایی از نمودار که نتیجه منفی را نشان می‌دهد، به این معناست که زمان پاسخ الگوریتم مورد نظر، از الگوریتم LIFOC بهتر بوده است. بنابراین مشاهده می‌شود که عملکرد الگوریتم ژنتیک با افزایش m، از سایر الگوریتم‌های مورد بررسی نیز بهتر خواهد بود.

در جدول شماره (۳)، الگوریتم‌ها از نظر میانگین مدت زمان اجرا و میانگین کل درصد خطای نسبی با یکدیگر مقایسه شده‌اند. این زمان‌ها برای m=5 و δ=0.5 بیان شده‌اند. همان‌طور که در جدول دیده می‌شود، میانگین کل درصد خطای نسبی الگوریتم ژنتیک نسبت به بقیه الگوریتم‌ها کمتر است اما زمان لازم برای اجرای این الگوریتم نسبت به دیگر الگوریتم‌های غیربهینه، بیشتر

است که این زمان نیز با توجه به زمان اجرای الگوریتم بهینه، قابل قبول خواهد بود.

جدول(۳): مقایسه زمان اجرا و میانگین کل درصد خطای نسبی الگوریتم‌ها برای m=5, δ=0.5

نام الگوریتم	میانگین زمان اجرا (ثانیه)	$\overline{\overline{\Delta T_v}}=\frac{\sum_{i=1}^{25}\overline{\Delta T_v^i}}{25}$
الگوریتم بهینه	115.8594	0
الگوریتم ژنتیک	12.50	0.09162
الگوریتم ITERLP	0.64063	0.19391
الگوریتم FIFOC	0.4375	5.89357
الگوریتم LIFOC	0.0625	0.18530
الگوریتم SPORT	0.0625	0.32769

۵- نتیجه‌گیری و کارهای آینده

مسئله زمان‌بندی بار محاسباتی تقسیم‌پذیر با درنظرگرفتن زمان بازگشت نتیجه در یک سیستم ناهمگن، جزو مسائل ترکیباتی با درجه پیچیدگی بالا و از مرتبه $O(m!^2)$ است که هنوز الگوریتمی بهینه از مرتبه چند جمله‌ای برای حل آن ارائه نشده است. روش‌های موجود نیز روش‌هایی ابتکاری هستند که نمی‌توانند در تمامی حالات، جواب قابل قبول تولید کنند. در این میان، الگوریتم ژنتیک پیشنهاد شده، در اکثر موارد جواب‌های بهتر و نزدیک‌تری به جواب بهینه تولید می‌کند. این نتیجه‌گیری حتی برای مقادیر بزرگ m و مقادیر نزدیک به یک برای پارامتر دلتا نیز صادق است.

در پایان و در راستای این تحقیق، می‌توان در نظرگرفتن مواردی عملی مانند محدودیت حافظه در کامپیوترهای کارگر و همچنین در نظرگرفتن مدل تاخیر غیرخطی در شبکه را به عنوان کارهای آتی در نظر داشت. استفاده از روش‌های ترکیبی[۲۹] در الگوریتم ژنتیک جهت کاهش مدت زمان اجرای این الگوریتم را نیز می‌توان به عنوان ادامه کار بیان نمود.

مراجع

[1] Bharadwaj, V., Ghose, D., Mani, V., Robertazzi, T. G., *Scheduling Divisible Loads in parallel and Distributed Systems*, IEEE CS Press, 1996.

[2] Vanderbei, R. J., *Linear Programming: Foundations and Extensions*, 2nd Ed., International Series in Operations Research & Management, vol. 37, Kluwer Academic Publishers, 2001.

[3] Holland, H. J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[4] Michalewicz, Z., *Genetic + Data Structures = Evolution Programs*, New York: AI Series Springer-Verlag, 1994.

[5] Haupt, R. L., Haupt, S. E., *Practical Genetic Algorithms*, 2nd Ed., John Wiley & Sons, 2004.

[6] Bharadwaj, V., Ghose, D., Robertazzi, T. G., "*Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems*", Cluster Computing, vol.6, no.1, pp.7-17, jan. 2003.

[7] Robertazzi, T. G., "*Ten Reasons to Use Divisible Load Theory*", Computer, pp. 63-68, May 2003.

[8] Jingxi, J., Bharadwaj, V., Ghose, D., "*Adaptive Load Distribution Strategies for Divisible Load Processing on Resource Unaware Multilevel Tree Networks*", IEEE Transactions on Computers, vol. 65, no. 7, pp. 999-1005, 2007.

[9] Ghatpande, A., Nakazato, H., Beaumont, O., Watanabe, H., "*SPORT: An Algorithm for Divisible Load Scheduling With Result Collection on Heterogeneous Systems*", IEICE Transactions on Communications, vol. E91-B, no. 8 August 2008.

[10] Ghatpande, A., Nakazato, H., Beaumont, O., Watanabe, H., "*Analysis of Divisible Load Scheduling with Result Collection on Heterogeneous Systems*", IEICE Transactions on Communications, vol. E91-B, no. 7, July 2008.

[11] Joines, J., Houck, C., "*On the Use of Non-Stationary Penalty Functions to Solve Constrained Optimization Problems with Genetic Algorithms*", IEEE International Symposium on Evolutionary Computation, pp. 579-584, 1994.

[12] Cheng, Y. C., Robertazzi, T. G., "*Distributed Computation with Communication Delays*", IEEE Transactions on Aerospace and Electronic Systems, vol. 24, no. 6, pp. 700-712, Nov. 1988.

[13] Li, X., Bharadwaj, V., Ko, C. C., "*Distributed Image Processing on a Network of Workstations*", Int'l J. Computers and Applications, vol. 25, no. 2, pp. 1-10, 2003.

[14] Blazewicz, J., Drozdowski, M., Markiewicz, M., "*Divisible Task Scheduling: Concept and Verification*", Parallel Computing, vol. 25, pp. 87-98, 1990.

[15] Chan, S., Bharadwaj, V., Ghose, D., "*Large Matrix-Vector Products on Distributed Bus Networks with Communication Delays Using the Divisible Load Paradigm: Performance and Simulation*", Math. and Computers in Simulation, vol. 58, pp. 71-92, 2001.

[16] Altilar, D. Paker, Y., "*Optimal Scheduling Algorithms for Communication Constrained Parallel Processing*", Proc. Eighth Int'l Euro-Par Conf. pp. 197-206, 2002.

[17] Goldberg, D.E., Lingle, R., "*Allel Loci and Traveling Salesman Problem*", in: J.J. Grefenstette(ed.) Presented at the first International Conf. on Genetic Algorithms, pp. 154-159, 1985.

[18] Rosenberg, A. L., "*Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier Is not Better*", IEEE International Conf. on Cluster Computing, pp. 124-131, Newport Beach, CA, Oct. 2001.

[19] Suresh, S., Mani,V., Omkar, S. N., Kim, H. J., "*Divisible Load Scheduling in Distributed Systems with Buffer Constraints: Genetic Algorithm and Linear Programming Approach*", International Journal of Parallel, Emergent and Distributed Systems, Vol. 21, No. 5, pp. 303-321, Oct. 2006.

[20] Ghatpande, A., Nakazato, H., Watanabe, H., Beaumont, O., "*Divisible Load Scheduling with Result Collection on Heterogeneous Systems*", Proc. Heterogeneous Computing Workshop(HCP 2008), April 2008.

[21] Beaumont, O., Marchal, L., Rehn, V., Robert Y., "*FIFO Scheduling of Divisible Loads with Return Messages Under the One Port Model*", Proc. Heterogeneous Computing Workshop HCW'06, April 2006.

[22] Robertazzi, T., http://www.ece.sunysb.edu/~tom/dlt.html.

آخرنویس‌ها

[1] Jobs
[2] Heterogeneous
[3] Links
[4] NP_Hard
[5] Divisible Load
[6] Load Units
[7] Fine-Granularity
[8] Massive-Data
[9] Data Mining
[10] Master-Worker
[11] Topology
[12] Makespan
[13] Last In First Out Channel
[14] First In First Out Channel
[15] Iteratively Solving Linear Programs
[16] System Parameters based Optimized Result Transfer
[17] Homogeneous
[18] Application
[19] Allocation Sequence
[20] Collection Sequence
[21] Exhaustive Search
[22] Greedy
[23] Fitness Function
[24] Crossover
[25] Mutation
[26] Selection
[27] Elitism
[28] Partially Mapped Crossover (PMC)
[29] Hybrid