مجله تحقیق در عملیات در کاربردهای آن، سال سیزدهم، شماره دوم (پیاپی ۴۹)، تابستان ۹۵، ص ص ۵۲-۳۵
شاپا ۷۲۸۶-۲۲۵۱

بهینه‌سازی زمان‌بندی الگوریتم‌های موازی با استفاده از الگوریتم ژنتیک

خدیجه نعمتی[۱]، امیرحسین رفاهی شیخانی[*۲]، سهراب کردرستمی[۳]

۱- دانشجوی کارشناسی‌ارشد، دانشگاه آزاد اسلامی، واحد لاهیجان، دانشکده علوم ریاضی، گروه علوم کامپیوتر، لاهیجان، ایران

۲- استادیار، دانشگاه آزاد اسلامی، واحد لاهیجان، دانشکده علوم ریاضی، گروه علوم کامپیوتر، لاهیجان، ایران

۳- استاد، دانشگاه آزاد اسلامی، واحد لاهیجان، دانشکده علوم ریاضی، گروه علوم کامپیوتر، لاهیجان، ایران

رسید مقاله: ۹ دی ۱۳۹۴

پذیرش مقاله: ۷ خرداد ۱۳۹۵

چکیده

زمان‌بندی[۱] مجموعه‌ای از ماشین‌های موازی که در یک محیط هستند، هم از نظر تئوری و هم از نظر کاربردی مهم است. از نظر تئوری، تعمیم مساله‌ی زمان‌بندی یک ماشین است و از نظر کاربردی صحت منابع موازی در جهان واقعی می‌باشد. وقتی ماشین‌ها، کامپیوتر باشند یک برنامه‌ی موازی نیاز است زیرا اعضای مجموعه به طور موازی اجرا می‌شوند و این اجرا براساس ارتباطات تقدمی[۲] آن‌ها است. مزیت اجرای زمان‌بندی وظایف، قدرت محاسباتی کامل را فراهم می‌کند که به وسیله‌ی سیستم چندپردازنده یا چندکامپیوتری به دست می‌آید. در این مقاله نشان می‌دهیم مساله‌ی تخصیص تعدادی وظایف ناهمسان در سیستم‌های چندپردازنده یا چندکامپیوتری چگونه است. مدل فرضی سیستم شامل تعدادی پردازنده‌ی همسان است و در یک زمان، فقط یک وظیفه، روی یک پردازنده اجرا می‌شود و نیز همه‌ی زمان‌بندی‌ها و وظایف، غیرانحصاری[۳] هستند.

کلمات کلیدی: الگوریتم ژنتیک، الگوریتم‌های موازی، گراف وظیفه، زمان‌بندی وظایف.

۱ مقدمه

سیستم‌های پردازش موازی امروزه کاربردهای زیادی در زمینه‌ی پردازش اطلاعات، مدل‌سازی آب‌وهوا، سیستم‌های پایگاه‌داده، شبیه‌سازی بلادرنگ سیستم‌های پویا و... دارند. بیش‌ترین کارایی از این سیستم‌ها زمانی به دست می‌آید که وظایف بین پردازنده‌ها به صورت کارا تقسیم‌بندی شود [۱]. کارهای محاسباتی پیچیده نمی‌تواند در بازه زمانی قابل قبول بر روی یک پردازنده اجرا شود و به همین دلیل باید به زیرکارهای کوچک‌تر تقسیم گردد و با زمان‌بندی مناسب و اختصاص دادن آن‌ها به یک سیستم چندپردازنده، زمان انجام کلیه‌ی زیرکارها

* عهده‌دار مکاتبات

آدرس الکترونیکی: Ah_refahi@liau.ac.ir

[1] scheduling
[2] precedence relationship
[3] non-preemptive

کاهش یابد یا به عبارتی زمان اجرای آخرین کار مینیمم شود [۲]. زمان‌بندی کارها به طور مناسب از اهمیت بالایی برخوردار است. چون زمان‌بندی نامناسب [۱] می‌تواند منجر به شکست پتانسیل یک سیستم موازی شود و یا به علت سربار ارتباطی بیش از اندازه [۲] یا منابع تحت بهره‌برداری [۳] مزایای موازی‌سازی را خنثی کند [۳].

راه‌حل‌های کلاسیک برای این مساله [۶-۴] از ترکیب روش‌های جستجو و اکتشافی (یا ابتکاری) استفاده می‌کند در حالی‌که این روش‌ها اغلب راه‌حل‌های مناسب تولید می‌کنند، پس نتایج زمان‌بندی معمولاً زیربهینه [۴] است. این روش‌ها اغلب به علت فقدان ضمانت کارایی و مقیاس‌پذیری مورد انتقاد قرار می‌گیرند. در مقابل، الگوریتم ژنتیک [۷ و۸] راه‌حل‌های اتفاقی [۵] برای مساله‌های زمان‌بندی بزرگ فراهم می‌کند.

این مقاله پیاده‌سازی یک الگوریتم ژنتیک را برای مینیمم کردن طول زمان‌بندی‌های گراف وظیفه‌ای که باید روی یک سیستم چندپردازنده اجرا شود توصیف می‌کند. اهمیت مساله‌ی زمان‌بندی روی سیستم چندپردازنده ما را به مطالعه‌ی مقایسه‌ای [۶] رهنمون می‌کند. راه‌حل‌های زیادی برای مساله‌ی زمان‌بندی کارها روی سیستم چندپردازنده ارایه شده است [۳].

الگوریتم ژنتیک هولاند در زمان‌بندی [۷] به صورت یک استراتژی تکاملی [۷] ارایه شده است که کشف و جستجوی سریع فضای زمان‌بندی را مجاز دانسته و راه‌حل‌های خوبی را به سرعت پیدا می‌کندو آن‌ها را برای ایجاد زمان‌بندی در مسایل عمومی اعمال می‌نماید [۸].

کیم و همکاران [۹] یک مساله‌ی زمان‌بندی قطعی را که در آن وظایف چندگانه با تعداد S رابطه تقدمی، که روی ماشین‌های موازی چندگانه برابر، پردازش می‌شود در نظر می‌گیرد. هدف مینیمم کردن زمان اجراست. ارتباط تقدمی بین دو وظیفه j و i موقعیت وظیفه j را نشان می‌دهد که تا زمانی‌که i پردازش نشود از پردازش شدن آن جلوگیری می‌کند. این مورد با تعریف‌های استاندارد روابط تقدمی، متفاوت است به طوری‌که j نمی‌تواند شروع شود تا زمانی‌که i تکمیل گردد.

وو و همکاران [۱۰] یک الگوریتم ژنتیک جدید پیشنهاد داده است که هم افراد معتبر و هم نامعتبر را در جمعیت می‌پذیرد. این GA یک تابع ارزیابی شایستگی [۸] را به تدریج [۹] افزایش می‌دهد و مقادیر حاصل از تابع را تا تا زمانی‌که یک راه‌حل رضایت‌بخش [۱۰] پیدا شود بررسی می‌کند. این روش قابل قیاس با مساله‌های بزرگ نیست چون زمان زیادی صرف ارزیابی افراد نامعتبر می‌شود که شاید معتبر باشند.

مساله تخصیص وظیفه توسط بسیاری از محققان [۵، ۱۱-۱۷] بررسی شده است. چندین روش اکتشافی نیز پیشنهاد شده است مانند روش اکتشاف برمبنای کوتاه‌ترین مسیر [۱۱]، دوبخشی بازگشتی متعامد [۱۲]، الگوریتم ژنتیک

---

[1] inappropriate
[2] excessive
[3] under-utilization
[4] suboptimal
[5] stochastic solution
[6] comparative studies
[7] evolutionary
[8] incremental fitness function
[9] gradually
[10] satisfactory
[11] mincut-based heuristic
[12] orthogonal recursive bisection

و شبکه‌های عصبی[1]. روش‌های محاسبه‌ی تکاملی مختلفی ابداع شده است که زمان‌بندی وظایف را انجام می‌دهد. دهد. البته این کار با توجه به تفاوت نمایش‌های مستقیم و غیرمستقیم پروژه انجام می‌شود.

در این مقاله یک رویکرد مستقیم با روش ترکیب احتمالی سریع[2] و یک نمایش رمزگشایی غیرمستقیم[3] را بررسی می‌کنیم به طوری‌که فقط روی نتایج نمایش مستقیم متمرکز می‌شویم. ساختار باقی‌مانده‌ی این مقاله به صورت زیر است: در بخش دوم، الگوریتم‌های موازی شرح داده شده است. در بخش سوم، خلاصه‌ای در مورد الگوریتم ژنتیک بیان شده است. در بخش چهارم مفاهیم الگوریتم ژنتیک ارایه شده است. بخش پنجم به مساله زمان‌بندی وظایف موازی و الگوریتم ژنتیک اختصاص یافته است. انواع نمایش کروموزوم‌ها برای الگوریتم ژنتیک در بخش ششم آورده شده است. بخش هفتم به الگوریتم‌های اولویت‌دهی وظایف تخصیص یافته است. در بخش هشتم نتایج حاصل از بررسی‌های انجام شده شرح داده شده است. در بخش نهم نیز نتیجه‌گیری پایانی آورده شده است.

۲ الگوریتم‌های موازی[4]

یک برنامه‌ی موازی، مجموعه‌ای از کارهاست که بعضی از آن‌ها باید قبل از شروع دیگری کامل شود و ارتباطات تقدمی بین کارها معمولاً به صورت اجمالی در گراف بدون چرخه جهت‌دار[5] نشان داده می‌شود. مانند یک گراف وظیفه[6] که گره‌های این گراف همان وظایف[7] و کمان جهت‌دار بین گره‌ها نشان‌دهنده‌ی ارتباطات تقدمی عامل‌هاست. تعداد کارها و محدودیت‌های تقدمی بین آن‌ها، انجام یک وظیفه را به نحو احسن، مشکل می‌سازد. مساله مورد نظر شامل یافتن یک زمان‌بندی مناسب روی تعداد $m > ۲$ پردازنده با ظرفیت برابر و مینیمم کردن زمان پردازش کارهای مستقل، که به کلاس غیرچندجمله‌ای[8] تعلق دارد، می‌باشد. به طور غیررسمی یک کار موازی، وظیفه‌ای است که از عملیات اصلی یا مقدماتی نتیجه می‌شود. مانند، یک روال عددی، که شرایط کافی برای اجرا شدن به وسیله‌ی بیش از یک پردازنده را دارا هستند. این دیدگاه کلی‌تر از موارد استاندارد است و کارهای متوالی را مانند یک مورد ویژه شامل می‌شود [۱۸].

۲-۱ انواع الگوریتم‌های موازی[9]

چندین نسخه از کارهای موازی وجود دارد که به اجرایشان روی سیستم موازی و توزیع‌شده وابسته هستند.

- انعطاف‌ناپذیر[10]: وقتی تعداد پردازنده‌ها برای اجرای PT، ثابت باشد. تعداد آن می‌تواند هم توانی از ۲ یا هر عدد صحیح دیگری باشد [۱۹].

[1] neural network
[2] as soon as possible crossover
[3] indirect-decode
[4] Parallel Algorithm
[5] directed acyclic graph
[6] task graph
[7] task
[8] NP-complete
[9] Typology of Parallel Task
[10] rigid

- قالبی[1]: در این مورد تعداد پردازنده‌ها برای اجرا کردن PT ثابت نیست اما قبل از اجرا تعیین می‌شود به طوری‌که این تعداد تا زمانی که PTها تکمیل شوند تغییر نمی‌کند.
- انعطاف‌پذیر[2]: در بیش‌تر موارد تعداد پردازنده‌ها ممکن است در طول اجرا به وسیله‌ی الگوریتم انحصاری و یا ساده با دوباره توزیع کردن داده‌ها تغییر کند.

از نظر کاربردی بیش‌ترین برنامه‌های کاربردی موازی انعطاف‌پذیرند. یک توسعه‌دهنده‌ی کاربردی، تعداد دقیق پردازنده‌هایی را که قرار است در زمان اجرا به‌کار گیرد نمی‌داند [۱۸].

۳ خلاصه‌ای در مورد الگوریتم ژنتیک

ایده‌ی اصلی الگوریتم‌های ژنتیک بر پایه‌ی نظریه‌ی داروین می‌باشد. براساس نظریه‌ی داروین نسل‌هایی که از ویژگی‌ها و خصوصیات برتری نسبت به نسل‌های دیگر برخوردارند شانس بیش‌تری نیز برای بقا و تکثیر خواهند داشت و ویژگی‌ها و خصوصیات برتر آن‌ها به نسل‌های بعدی آنان نیز منتقل خواهد شد. جواب مساله‌ای که از طریق الگوریتم ژنتیک حل می‌شود مرتبًا بهبود می‌یابد. الگوریتم ژنتیک با مجموعه‌ای از جواب‌ها که از طریق کروموزوم‌ها نشان داده می‌شود شروع می‌گردد. در این الگوریتم جواب‌های حاصل از یک جمعیت[3] برای تولید جمعیت بعدی استفاده می‌شود که عمل تولیدنسل جدید با ترکیب و یا جهش[4] همراه خواهد بود. در این فرایند امید است که جمعیت جدید نسبت به جمعیت قبلی بهتر باشد. انتخاب بعضی از جواب‌ها (کروموزوم ها[5]) از میان کل جواب‌ها والدین[6] به منظور ایجاد جواب‌های جدید فرزندان[7] براساس میزان مطلوبیت آن‌ها می‌باشد که این کار با استفاده از تابع برازش[8] صورت می‌گیرد. طبیعی است که جواب‌های مناسب‌تر شانس بیش‌تری برای تولید مجدد داشته باشند. این فرایند تا برقراری شرطی که از پیش تعیین شده است (مانند تعداد جمعیت‌ها یا میزان بهبود جواب) ادامه می‌یابد [۲۰].

٤ مفاهیم الگوریتم ژنتیک

به طور کلی، الگوریتم‌های ژنتیکی از اجزای زیر تشکیل می‌شوند.

٤-۱ کروموزوم[9]

در الگوریتم‌های ژنتیکی، هر کروموزوم نشان‌دهنده‌ی یک نقطه در فضای جستجو و یک راه‌حل ممکن برای مساله مورد نظر است. خود کروموزوم‌ها (راه‌حل‌ها) از تعداد ثابتی ژن (متغیر) تشکیل می‌شوند.

[1] moldable
[2] malleable
[3] Population
[4] mutation
[5] chromosome
[6] Parent
[7] Offspring
[8] fitness
[9] Chromosome

٤-۲ جمعیت[1]

مجموعه‌ای از کروموزوم‌ها یک جمعیت را تشکیل می‌دهند. با تاثیر عملگرهای ژنتیکی بر روی هر جمعیت، جمعیت جدیدی با همان تعداد کروموزوم تشکیل می‌شود. یکی از ویژگی‌های ژنتیک این است که به جای تمرکز بر روی یک نقطه از فضای جستجو یا یک کروموزوم، بر روی جمعیتی از کروموزوم‌ها کار می‌کند [۲۰].

٤-۳ تابع برازندگی[2]

به منظور حل هر مساله با استفاده از الگوریتم‌های ژنتیکی، ابتدا باید یک تابع برازندگی برای آن مساله ابداع شود. برای هر کروموزوم، این تابع عددی غیر منفی را برمی‌گرداند که نشان‌دهنده‌ی شایستگی یا توانایی فردی آن کروموزوم است [۲۰].

٤-٤ عملگرهای ژنتیکی

در الگوریتم‌های ژنتیکی، در طی مرحله‌ی تولیدمثل[3] از عملگرهای ژنتیکی استفاده می‌شود. با تاثیر این عملگرها بر روی یک جمعیت، نسل[4] بعدی آن جمعیت تولید می‌شود. عملگرهای انتخاب[5]، ترکیب و جهش معمولاً بیش‌ترین ترین کاربرد را در الگوریتم‌های ژنتیکی دارند. حال هر یک از عملگرهای فوق به صورت جداگانه معرفی می‌شوند.

٤-٤-۱ عملگر انتخاب

این عملگر از بین کروموزوم‌های موجود در یک جمعیت، تعدادی کروموزوم را برای تولیدمثل انتخاب می‌کند. کروموزوم‌های برازنده‌تر شانس بیشتری دارند تا برای تولیدمثل انتخاب شوند. برخی از روش‌های متداول انتخاب عبارت است از: انتخاب از طریق چرخ رولت، انتخاب از طریق نخبه‌سالاری، انتخاب از طریق رقابتی، انتخاب از طریق مسابقه.

٤-٤-۲ عملگر ترکیب

عملگر ترکیب بر روی یک زوج کروموزوم از نسل مولد عمل کرده و یک زوج کروموزوم جدید تولید می‌کند. عملگرهای ترکیب متعددی از قبیل، ترکیب تک‌نقطه‌ای[6] و ترکیب دونقطه‌ای[7] وجود دارد.

٤-٤-۳ عملگر جهش

پس از اتمام عمل ترکیب، عملگر جهش بر روی کروموزوم‌ها اثر داده می‌شود. این عملگر یک ژن از یک کروموزوم را به طور تصادفی انتخاب نموده و سپس محتوای آن ژن را تغییر می‌دهد [۲۱].

---

[1] Population
[2] Fitness Function
[3] Reproduction
[4] Generation
[5] Selection
[6] One-point Crossover
[7] Two-point Crossover

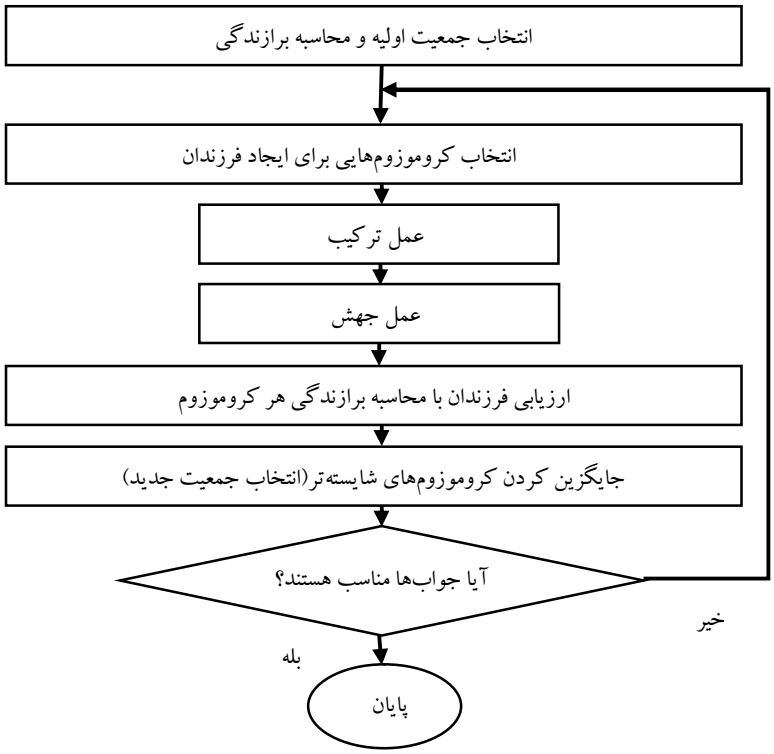
٤-٥ تعیین جمعیت اولیه

الگوریتم ژنتیک کار خود را با تولید جمعیت اولیه‌ای از کروموزوم‌ها آغاز می‌کند و سپس در یک حلقه به طور مکرر تعدادی از کروموزوم‌های برتر نسل فعلی را انتخاب کرده و سپس نسل جدیدی را از این کروموزوم‌ها تولید می‌کند. منظور از تولید جمعیت اولیه، تولید تعدادی جواب برای مساله خواهد بود. تولید جواب‌های اولیه نیز به دو صورت تصادفی و اکتشافی می‌تواند انجام پذیرد.

٤-٦ شرایط توقف الگوریتم ژنتیک

- به تعداد ثابتی نسل برسیم.
- زمان محاسبه‌ی اختصاص داده شده تمام شود.
- بیش‌ترین درجه‌ی برازش برای فرزندان حاصل شود یا دیگر نتایج بهتری حاصل نشود.
- ترکیبی از شرایط بالا.

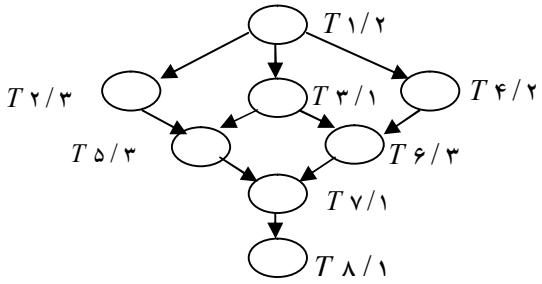
٤-٧ تشریح کلی الگوریتم ژنتیک



شکل ۱. نمودار مراحل الگوریتم ژنتیک

۵ مساله زمان‌بندی وظایف موازی و الگوریتم ژنتیک

مساله زمان‌بندی وظایف در یک گراف وظیفه به صورت بهینه با به کار بردن m پردازنده، عبارت است از تخصیص وظایف به مجموعه پردازنده‌ها به طوری‌که اولویت بین آن‌ها رعایت شود و همچنین پردازش وظایف در کوتاه‌ترین زمان ممکن انجام شود. یک گراف وظیفه، یک نمایش ساده از اجرای برنامه‌های موازی است. البته با نادیده گرفتن سربارها با توجه به توقف‌ها برای دسترسی به منابع و غیره. با وجود این، مبنایی برای تخصیص پردازنده به صورت ایستا ارایه می‌دهد. یک زمان‌بندی[1]، به صورت تخصیص کارها یا وظایف به پردازنده‌ها است که معمولاً با نمودار گانت[2] به تصویر کشیده می‌شود. نمودار گانت به زمان شروع و پایان هر کار در پردازنده‌های در دسترس، اشاره دارد.



شکل ۲. مدل گراف وظیفه

متغیرهای عملکردی دیگری مثل استفاده از پردازنده‌های انفرادی[3] یا توزیع بار یکنواخت را نیز می‌توان در نظر گرفت. یک برنامه‌ی زمان‌بندی ساده می‌تواند به صورت بهینه با زمان چندجمله‌ای[4] حل شود [۲۲].

$P_۲$			$T_۳$	$T_۴$		$T_۶$				
$P_۱$	$T_۱$		$T_۲$			$T_۵$			$T_۷$	$T_۸$
Time Slot	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
Time	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰

شکل ۳. زمان‌بندی ۸ وظیفه روی ۲ پردازنده با روش LSA

٦ انواع نمایش کروموزوم‌ها برای الگوریتم ژنتیک

٦-۱ نمایش مستقیم راه‌حل‌ها

در اینجا یک زمان‌بندی را به صورت نمایش یک کروموزوم پیشنهاد می‌دهیم. فرض می‌کنیم دو زمان‌بندی مختلف a و b را داریم

شکل ۴ برای مدل گراف وظیفه شکل ۲ نشان داده شده است که به وسیله‌ی نمودار گانت ارایه می‌شود.

[1] schedule
[2] Gant Chart
[3] individual processor
[4] P-complete

P_2			T_3	T_4		T_6				
P_1	T_1		T_2			T_5			T_7	T_8
TS	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰

Schedule (a)

P_2			T_3			T_5						T_8
P_1	T_1		T_2			T_4		T_6			T_7	
TS	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲

Schedule (b)

شکل ٤. نمودار گانت زمان‌بندی احتمالی برای مدل گراف وظیفه شکل ۲

ارتباطات تقدمی توصیف شده در گراف وظیفه را می‌توان در ماتریس تقدمی A نشان داد به طوری‌که در آن وقتی عضو a_{ij} در این ماتریس برابر ۱ باشد، یعنی وظیفه‌ی i اُم مقدم بر وظیفه‌ی j اُم است و در غیر این صورت برابر صفر می‌باشد. یک ژن[1] در کروموزوم را به صورت یک چهارتایی زیر نمایش می‌دهیم:

$\langle task\_id \quad proc\_id \quad inite\_time \quad end\_time \rangle$

$task\_id$: شناسه‌ای است که به یک وظیفه اختصاص می‌یابد.

$proc\_id$: شناسه‌ای که به پردازنده‌ای که وظیفه‌ی $task\_id$ را انجام می‌دهد اختصاص می‌یابد.

$inite\_time$: زمان شروع وظیفه‌ی $task\_id$ در پردازنده‌ی $proc\_id$ می‌باشد.

end_time: زمان پایان وظیفه‌ی $task\_id$ در پردازنده‌ی $proc\_id$ می‌باشد.

C_a:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۲،۳،۵	۵،۱،۵،۸	۶،۲،۵،۸	۷،۱،۸،۹	۸،۱،۹،۱۰
C_b:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۱،۵،۷	۵،۲،۵،۸	۶،۱،۷،۱۰	۷،۱،۱۰،۱۱	۸،۲،۱۱،۱۲

شکل ۵. نمایش مستقیم کروموزوم‌ها شکل ۲

نمایش شکل ۵ یک مشکل دارد اگر یک مسیر قراردادی[2] مثل یک ترکیب تک‌نقطه‌ای نامعتبر به کار گرفته شود، فرزندان نامعتبر(زمان‌بندی‌های غیرمحتمل[3]) ایجاد می‌شوند.

به عنوان مثال اگر بخواهیم عملگر ترکیب را در شکل ۵ بعد از موقعیت پنجم اعمال کنیم دو کروموزوم نامعتبر به دست می‌آید (شکل ۶). هر دو کروموزوم این محدودیت که در یک زمان یک پردازنده یک وظیفه را انجام دهد نقض می‌کنند. به طور مثال ژن ۵ و ۶ در کروموزوم‌های C_a و C_b زمان‌بندی نامعتبری را توصیف می‌کنند چون پردازنده‌ی P_1 برای کروموزوم C_a و P_2 برای کروموزوم C_b دو وظیفه را در مدت زمان مشابه پردازش می‌کنند.

[1] gene
[2] conventional
[3] feasible scheduling

$C_{a'}$:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۲،۳،۵	۵،۱،۵،۸	۶،۱،۷،۱۰	۷،۱،۱۰،۱۱	۸،۲،۱۱،۱۲
$C_{b'}$:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۱،۵،۷	۵،۲،۵،۸	۶،۲،۵،۸	۷،۱،۸،۹	۸،۱،۹،۱۰

شکل ٦. فرزندان (کروموزوم‌های) نامعتبر مربوط به شکل ۲

الگوریتم ترمیم سعی می‌کند تا راه‌حل‌های معتبر را از میان نامعتبرها پیدا کند این روش از عمل ترکیب پیشنهادی برانز [۲۲] استخراج شده است اگر یک عمل (پردازش وظیفه[1]) از یکی از والدین‌ها به ارث برسد، نمی‌تواند در آن مدت زمان مشخصی که به پردازنده نسبت داده شده، زمان‌بندی شود در این مورد پردازنده نسبت داده شده به وظیفه، تغییر نمی‌کند، ولی با تأخیر، در پردازش بعدی در دسترس قرار می‌گیرد (شکل ۷).

$C_{a''}$:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۲،۳،۵	۵،۱،۵،۸	۶،۱،۸،۱۱	۷،۱،۱۱،۱۲	۸،۲،۱۲،۱۳
$C_{b''}$:	۱،۱،۰،۲	۲،۱،۲،۵	۳،۲،۲،۳	۴،۱،۵،۷	۵،۲،۵،۸	۶،۲،۸،۱۱	۷،۱،۱۱،۱۲	۸،۱،۱۲،۱۳

شکل ۷. فرزندان یا زمان‌بندی محتمل برای مدل گراف وظیفه شکل ۲

$P_۲$			$T_۳$	$T_۴$								$T_۸$
$P_۱$	$T_۱$		$T_۲$			$T_۵$			$T_۶$		$T_۷$	
TS	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳

Schedule (a")

$P_۲$			$T_۳$			$T_۵$			$T_۶$			
$P_۱$	$T_۱$		$T_۲$			$T_۴$					$T_۷$	$T_۸$
TS	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳

Schedule (b")

شکل ۸. نمودار گانت فرزندان یا زمان‌بندی محتمل برای مدل گراف وظیفه شکل ۲

٦-۲ نمایش غیرمستقیم راه‌حل‌ها

دراین روش، یک زمان‌بندی به صورت یک کروموزوم رمزگذاری[2] می‌شود به طوری‌که شماره‌ی وظیفه مورد نظر متناظر با موقعیت ژن و به پردازنده‌های تخصیص یافته به هر وظیفه با محتوای آن ژن نشان داده می‌شود. مانند شکل ۹.

Proccessor	۱	۲	۳	۲	۱	۳	۱	۲
Task	۱	۲	۳	۴	۵	۶	۷	۸

شکل ۹. ساختار کروموزوم برای مساله تخصیص وظایف

[1] task processing
[2] encoding

این یک ایده برای رمزگشایی[1] است. یک رمزگشا[2] یک مسیردهی[3] از فضای نمایش که فضای راه‌حل احتمالی استنتاج شده را ارزیابی می‌کند. در اینجا کروموزوم با دستور دادن به رمزگشا چگونگی ساخت زمان‌بندی احتمالی را تعیین می‌کند. در این روش، محدوده‌ی ممنوعه برای پردازنده‌ی بیکار این است که استراحت کند در حالی‌که وظیفه‌ی آماده‌ای وجود داشته باشد. از مزایای وجود رمزگشا توانایی ایجاد کردن فرزند معتبر است، حتی با روش‌های عملکردی متداول ساده است. از معایب آن هم ارزیابی کُند این روش است. برای مدل گراف وظیفه‌ی شکل ۲ با سه پردازنده، والدین کروموزوم C_1 و C_2 و با ترکیب تک‌نقطه‌ای بعد از موقعیت چهارم، فرزند معتبر[4] $C_{1'}$ و $C_{2'}$ به دست می‌آیند.

C_1:	۱	۲	۳	۱	۱	۱	۳	۲	$C_{1'}$:	۱	۲	۳	۱	۳	۱	۲	۳
C_2:	۳	۱	۲	۲	۳	۱	۲	۳	$C_{2'}$:	۳	۱	۲	۲	۱	۱	۳	۲

شکل ۱۰. نمایش غیرمستقیم کروموزوم‌ها

۷ الگوریتم‌های اولویت‌دهی وظایف

الگوریتم‌های متنوعی برای اولویت‌دهی وظایف موازی استفاده می‌شود که چند نمونه را در ادامه به طور مختصر بیان می‌کنیم.

۷-۱ الگوریتم زمان‌بندی لیست LSA[5]

روشی که در این مقاله مورد بررسی قرار می‌گیرد این الگوریتم می‌باشد که نتایج حاصل از این روش را با دو روش ذکر شده در بخش‌های ۷-۲ و ۷-۳ مقایسه می‌کنیم. حال به شرح این الگوریتم می‌پردازیم.

برای به دست آوردن لیستی از وظایف مرتب شده براساس تقدم، کارهایی را به پردازنده‌ها نسبت می‌دهیم. با اختصاص دادن هرپردازنده در دسترس (پردازنده‌هایی که وظایف‌شان قبلاً تمام شده باشد) به اولین وظیفه‌ای که اختصاص داده نشده است این کار صورت می‌گیرد. پس داریم:

- $T = \{T_1,...,T_n\}$: یک مجموعه از وظیفه‌هاست.
- $\mu : T \to (0 \quad \infty)$: یک تابع است که زمان اجرایی را به هر وظیفه نسبت می‌دهد که بزرگ‌تر مساوی یک ترتیب جزیی $(\geq)$ در T می‌باشد.
- L: لیست تقدمی از وظایف مجموعه‌ی T.

در هر زمان یک پردازنده بیکار[6]، فوراً از لیست L، اولین وظیفه آماده را حذف می‌کند که همان وظیفه‌ی زمان‌بندی نشده می‌باشد که اجداد آن کوچک‌تر مساوی (ترتیب جزیی $\leq$)همه وظایف تمام (کامل) شده است.

[1] decoding
[2] decoder
[3] mapping
[4] valid offspring
[5] List Scheduling Algorithm
[6] idle

در این مورد دو تا یا بیش‌تر از پردازنده‌ها سعی می‌کنند تا وظایف مشابه را انجام دهند (همان وظایف را انجام دهند). پردازنده‌ای موفق می‌شود که دارای کم‌ترین شناسه و پردازنده‌های باقی‌مانده هم به دنبال وظیفه‌ی مناسب می‌گردند. به کار بردن این روش اکتشافی LSA در مقابل شواهد[1] غیرمستدل[2] می‌تواند اتفاق بیفتد به عنوان مثال در شکل ۱۱ افزایش تعداد پردازنده‌ها و کاهش زمان اجرای یک یا تعداد بیش‌تری از وظایف یا با حذف بعضی محدودیت‌های تقدمی که می‌تواند makespan[3] را افزایش دهد [۱۴].

۷-۲ زمان‌بندی و اولویت‌دهی وظایف برحسب ارتفاع

هدف از زمان‌بندی در یک سیستم چندپردازنده، انتساب n وظیفه به m پردازنده در یک سیستم چندپردازنده است تا زمان اتمام اجرای آخرین وظیفه در این سیستم مینیمم شود. برای سادگی، اگر دو وظیفه بر روی دو پردازنده‌ی مختلف زمان‌بندی شود هزینه‌ی ارتباطی برای ارسال داده‌ها بین دو وظیفه صفر در نظر گرفته می‌شود. از روش‌های پرکاربرد در زمان‌بندی، استفاده از اولویت‌دهی اجرای وظایف براساس ارتفاع آن‌ها می‌باشد. الگوریتم تولید زمان‌بندی کارها برحسب ارتفاع به صورت زیر می‌باشد:

۱. وظایف را به ترتیب افزایشی برحسب ارتفاع آن‌ها در یک صف مرتب نمایید.
۲. مراحل ۳ و ۴ را به تعداد وظایف تکرار کنید.
۳. یک عدد تصادفی r به طوری‌که $1 \le r \le m$ تولید کنید (m تعداد پردازنده می‌باشد). اولین کار را از صف وظایف مرتب شده انتخاب کرده و به پردازنده‌ی r اُم تخصیص دهید و سپس آن را حذف کنید.
۴. با تکرار به کار بردن الگوریتم تولید زمان‌بندی، یک جمعیت اولیه از گره‌های گراف مورد جستجو مورد نیاز است.

جدول ۱. شماره و ارتفاع مربوط به هر وظیفه شکل ۱۲

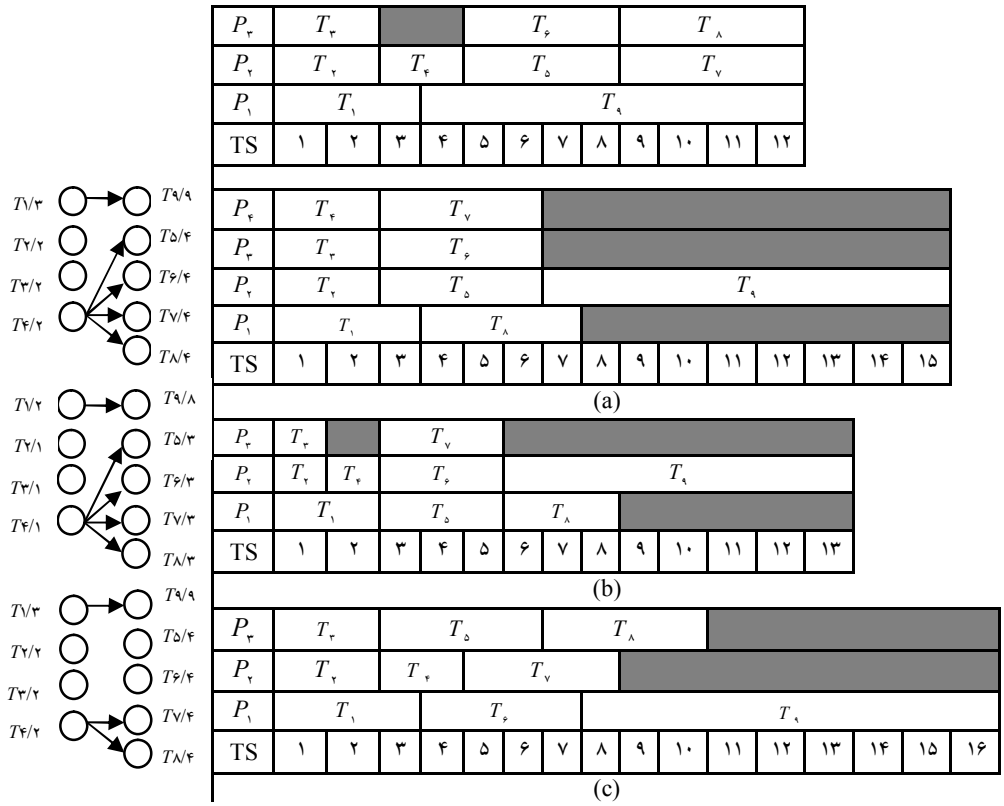
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
۰	۱	۱	۲	۲	۲	۲	۳	۳	۳	۳	۳	۴	۴	۴	۵

جدول ۲. شماره وظایف و زمان مربوط به هر وظیفه شکل ۱۲

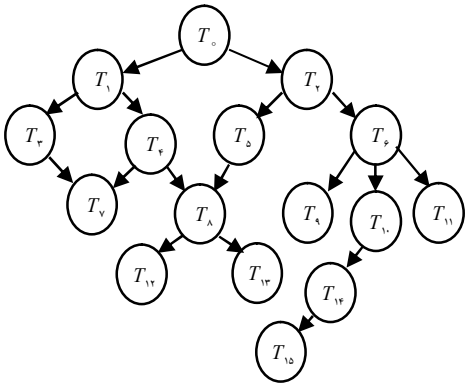
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
۳	۲	۴	۱	۱۰	۳	۶	۹	۷	۱۱	۵	۵	۸	۱۰	۱۵	۲

[1] intuition
[2] anomalies

[3] کل زمان اجرای زمان اجرای برنامه موازی



شکل ۱۱. افزایش تعداد پردازنده‌ها و کاهش مدت زمان انجام وظایف و حذف محدودیت تقدم می‌تواند باعث افزایش MAKESPAN به‌کار رفته در روش اکتشافی گراهام شود.



شکل ۱۲. نمونه‌ای از یک گراف وظیفه

۳-۷ زمان‌بندی و اولویت‌دهی وظایف برحسب تعداد نوادگان

این روش وظایف را براساس تعداد فرزندان و نوادگان[1] هر کار به پردازنده‌ها ارجاع می‌دهد تا اجرا شود و یعنی هر وظیفه که تعداد فرزندان بیش‌تری دارد، باید زودتر زمان‌بندی شود. اولویت‌دهی براساس تعداد فرزندان و تعداد نوادگان بهتر از اولویت‌دهی براساس ارتفاع است[۲۳]. به این دلیل که ممکن است تعدادی از کارها با

[1] offspring

ارتفاع یکسان برگ باشند و یا این‌که به برگ‌ها نزدیک باشند و تعدادی از آن‌ها زیرشاخه‌های متعددی داشته باشد و از برگ‌ها دور باشد که در روش اولویت‌دهی براساس ارتفاع، زمان‌بندی بدون توجه به این نکته انجام می‌شود در حالی‌که در روش اولویت‌دهی براساس تعداد فرزندان و نوادگان به این نکته توجه می‌شود و وظایف با زیرشاخه‌ها و نوادگان بیش‌تر، اولویت بالاتری دارند و بالطبع شاخه‌های شلوغ‌تر یا وظایف با تعداد نواده بیش‌تر، زودتر زمان‌بندی شوند و با اتمام آن‌ها تعداد وظایف بیش‌تری (فرزندان و نوادگان آن‌ها) امکان اجرا می‌یابند و زمان‌بندی بهتری به دست می‌آید. الگوریتم تولید زمان‌بندی کارها وظایف براساس تعداد نوادگان:

۱. وظایف را به ترتیب کاهشی برحسب تعداد نوادگان آن‌ها در یک صف مرتب کنید.
۲. وظایف با تعداد offspringها را در یک گروه مجزا قرار دهید و مراحل ۳ و ۴ را تا خالی شدن هر گروه انجام دهید.
۳. یک وظیفه از بین وظایف گروه به طور تصادفی انتخاب و از گروه حذف کنید.
۴. سپس وظیفه را براساس روش EST[1] به یکی از پردازنده‌های P_1 تا P_m اختصاص دهید به طوری‌که زمان شروع آن وظیفه بر روی آن پردازنده از پردازنده‌های دیگر کم‌تر باشد.

جدول ۳. شماره و OFFSPRING های مربوط به هر وظیفه شکل ۱۲

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
۱۵	۶	۱۰	۱	۴	۳	۵	۰	۲	۰	۲	۰	۰	۰	۱	۰

جدول ٤. مرتب کردن وظایف براساس تعداد فرزندان مربوط به شکل ۱۲

T_0	T_2	T_1	T_6	T_4	T_5	T_8	T_{10}	T_3	T_{14}	T_7	T_9	T_{11}	T_{12}	T_{13}	T_{15}
۱۵	۱۰	۶	۵	۴	۳	۲	۲	۱	۱	۰	۰	۰	۰	۰	۰

۸ نتایج بررسی

به منظور ارزیابی الگوریتم پیشنهادی از نرم افزار MATLAB R2014a استفاده شده است. تعداد جمعیت اولیه برای الگوریتم پیشنهادی برابر تعداد وظایف و تعداد نسل‌های الگوریتم کم‌تر از ۱۰۰۰ و آلفا احتمال راه‌حل بهینه با مقدار ۰/۵ و بتا احتمال کل راه‌حل‌های بهینه با مقدار ۰/۹ ونیز زمان اجرای وظایف بین ۱ تا ۱۰۰ واحد زمانی در نظر گرفته شده است. ارزیابی الگوریتم ژنتیک با اولویت‌دهی براساس لیست تقدمی پیشنهادی و الگوریتم‌های ژنتیک با اولویت‌دهی براساس تعداد فرزندان و اولویت‌دهی براساس ارتفاع در جدول‌های ۵ تا ۸ آورده شده است.

[1] Earliest Start Time

جدول ۵. نتایج بررسی سه الگوریتم با ۳ پردازنده

تعداد وظایف	۳۰	۵۰	۷۰	۹۰
زمان اتمام الگوریتم فرزندان	۱۰۰۲	۱۴۴۸	۱۹۱۵	۲۵۴۹
زمان اتمام الگوریتم ارتفاع	۱۱۵۸	۱۴۶۵	۱۹۶۹	۲۸۰۶
زمان اتمام الگوریتم لیست	۳۰۱	۴۵۲	۱۵۸۰	۱۹۵۴

جدول ٦. نتایج بررسی سه الگوریتم با ۵ پردازنده

تعداد وظایف	۳۰	۵۰	۷۰	۹۰
زمان اتمام الگوریتم فرزندان	۹۱۸	۱۶۳۰	۲۳۰۰	۳۱۹۲
زمان اتمام الگوریتم ارتفاع	۱۰۰۳	۱۶۸۳	۲۳۹۵	۳۲۴۲
زمان اتمام الگوریتم لیست	۲۶۹	۳۹۱	۶۰۳	۱۷۴۶

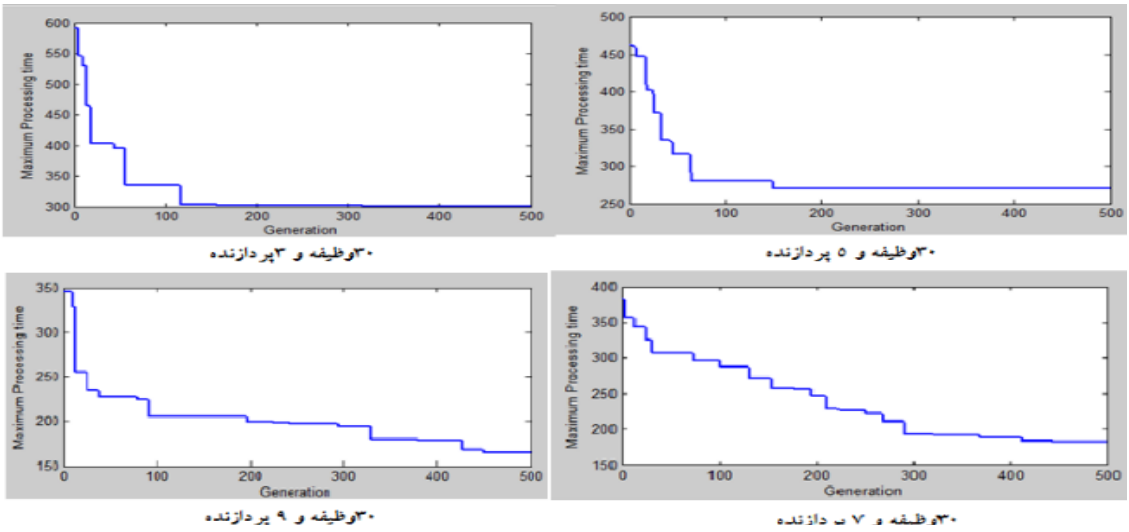
جدول ۷. نتایج بررسی سه الگوریتم با ۷ پردازنده

تعداد وظایف	۳۰	۵۰	۷۰	۹۰
زمان اتمام الگوریتم فرزندان	۱۲۲۲	۱۷۴۷	۱۹۸۱	۳۱۹۷
زمان اتمام الگوریتم ارتفاع	۱۲۶۰	۱۷۹۵	۱۹۹۰	۲۳۰۳
زمان اتمام الگوریتم لیست	۱۸۲	۳۰۹	۵۰۶	۶۳۷

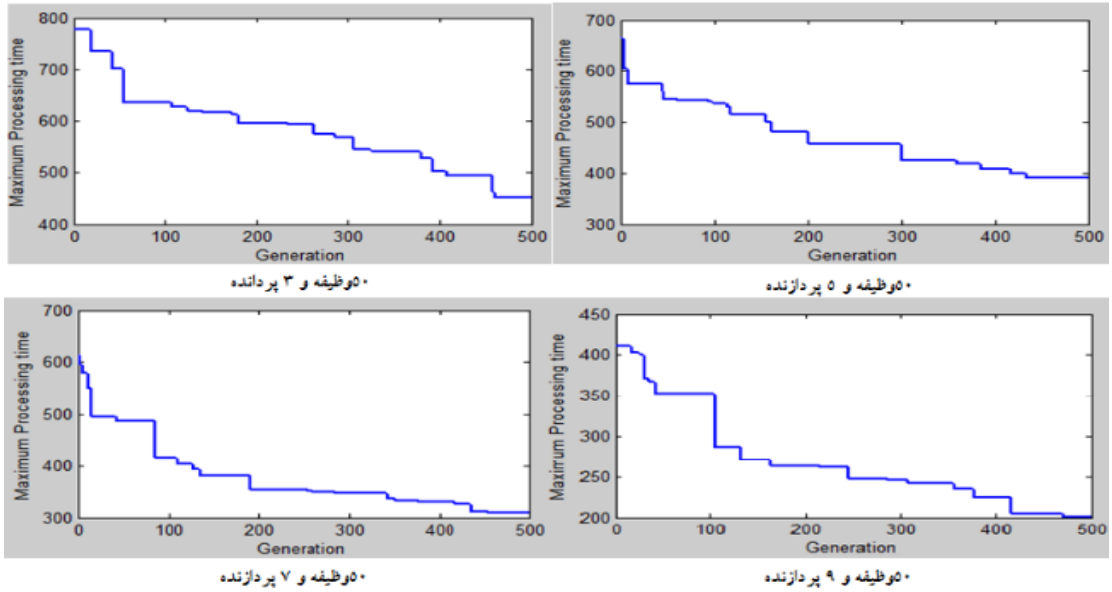
جدول ۸. نتایج بررسی سه الگوریتم با۹ پردازنده

تعداد وظایف	۳۰	۵۰	۷۰	۹۰
زمان اتمام الگوریتم فرزندان	۸۰۹	۱۷۲۰	۲۰۶۵	۲۷۴۲
زمان اتمام الگوریتم ارتفاع	۸۵۶	۱۷۵۳	۲۱۹۱	۲۷۶۰
زمان اتمام الگوریتم لیست	۱۶۵	۲۰۱	۴۶۱	۵۲۸

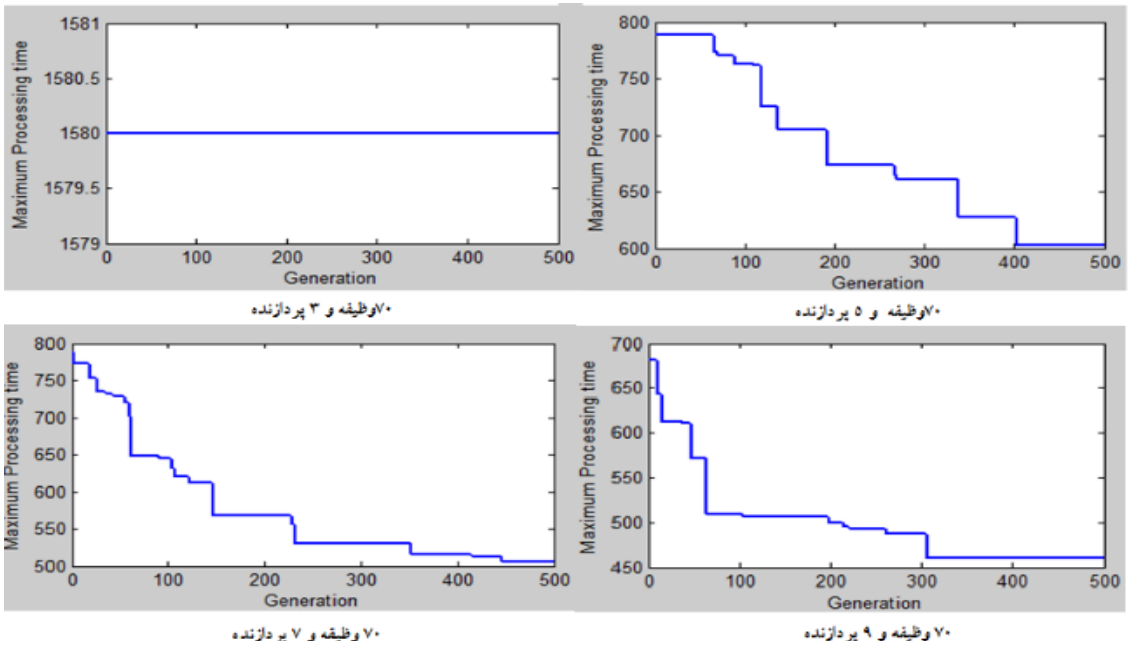
الگوریتم پیشنهادی را برای نمونه گراف‌های ۳۰، ۵۰، ۷۰، ۹۰ وظیفه‌ای اجرا کرده‌ایم نتایج به دست آمده نشان می‌دهد که، زمان اتمام این الگوریتم نسبت به الگوریتم‌های ارتفاع و فرزندان بهبود قابل توجهی یافته است و نیز با افزایش تعداد پردازنده‌ها، تأثیر مناسبی در زمان پایان اجرا برای هر یک از این گراف‌ها دارد. طبق مشاهده نتایج در جداول فوق، با افزایش تعداد وظایف تعداد پردازنده‌ها نیز باید به نسبت افزایش یابد وگرنه کاهش زمان اتمام اجرا اتفاق نمی‌افتد که این مساله چندان مطلوب نمی‌باشد. به طور مثال برای گراف ۹۰ وظیفه‌ای با تعداد ۳ و ۵ پردازنده و برای گراف ۷۰ وظیفه‌ای با تعداد ۳ پردازنده زمان اتمام اجرا زیاد بوده که با افزودن تعداد پردازنده‌ها نتیجه‌ای بهتر حاصل شده است. روند بهبود زمان اتمام اجرا در شکل‌های ۱۳ تا ۱۶ نشان داده شده است.



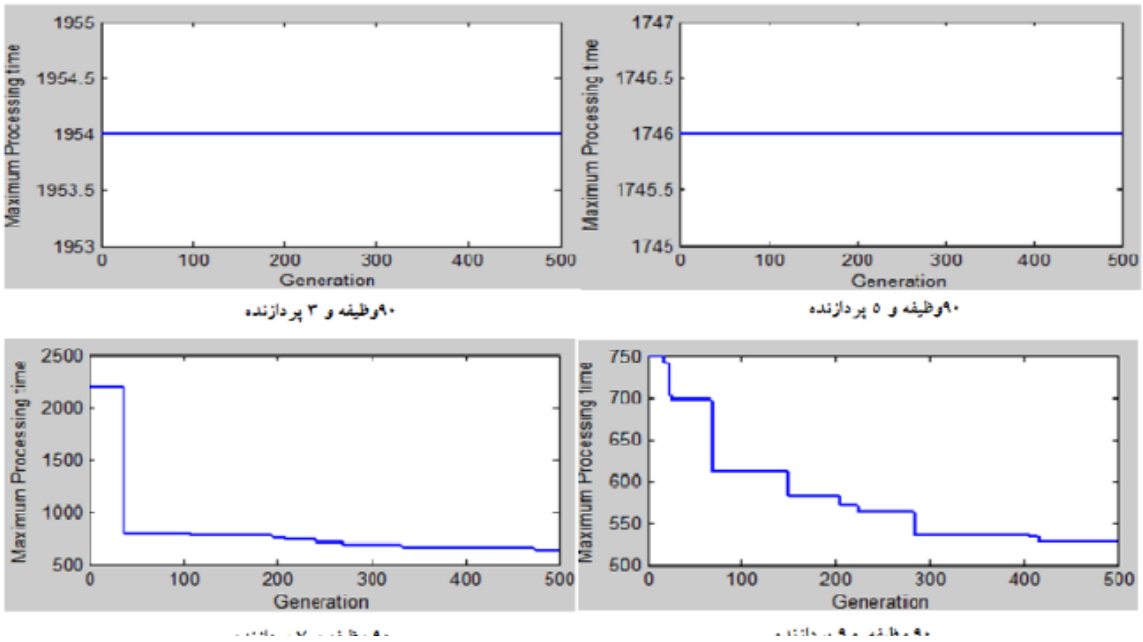
شکل ۱۳. نمودار زمان اجرای گراف ۳۰ وظیفه‌ای با تعداد پردازنده‌های مختلف با الگوریتم لیست



شکل ۱۴. نمودار زمان اجرای گراف ۵۰ وظیفه‌ای با تعداد پردازنده‌های مختلف با الگوریتم لیست



شکل ۱۵. نمودار زمان اجرای گراف ۷۰ وظیفه‌ای با تعداد پردازنده‌های مختلف با الگوریتم لیست



شکل ۱۶. نمودار زمان اجرای گراف ۹۰ وظیفه‌ای با تعداد پردازنده‌های مختلف با الگوریتم لیست

۹ نتیجه‌گیری پایانی

یکی از مسایل مهم در بحث بکارگیری الگوریتم‌های موازی، مساله بهینه سازی زمان‌بندی این نوع الگوریتم است. ما در این مقاله به دنبال تخصص تعداد بهینه وظایف به هر پردازنده روی سیستم های چند پردازنده یا

چندکامپیوتری بودیم که در نهایت منجر به کاهش زمان نهایی اجرای وظایف گردید. برای انجام این مهم از الگوریتم ژنتیک به همراه الگوریتم زمان‌بندی لیست LSA به طور همزمان استفاده نمودیم که در پایان ، نتایج بهتری به لحاظ دقت و زمان محاسبات نسبت به دو روش اولویت دهی دیگر که در متن مقاله به آن اشاره کردیم ، حاصل گردید.

منابع

[۱] مهرابی داودآبادی، ع.، مهرابی داودآبادی، س.، (۱۳۸۸). زمانبندی کارها روی سیستم‌های چند پردازنده‌ای با استفاده از الگوریتم‌های ژنتیک. اولین کنفرانس ملی نرم‌افزار ایران، رودهن.

[۲] عبدیزدان، م.، (۱۳۸۶). زمان‌بندی کارها در سیستم چندپردازنده‌ای با استفاده از یک الگوریتم ژنتیک جدید اولویت براساس تعداد فرزندان. سیزدهمین کنفرانس ملی انجمن کامپیوتر ایران، جزیره کیش، خلیج فارس، ایران.

[۲۰] محمدی، ل.، (۱۳۹۳). الگوریتم ژنتیک.

[۲۱] شمس، م.، (۱۳۹۰). پیاده سازی و حل مسائل کاربردی با الگوریتم ژنتیک، آریا پروژه.

[۲۳] نژادحسین، س.، حیدری، ع.، (۱۳۹۳). حل دسته‌ای از مسایل کنترل بهینه با استفاده ازالگوریتم ژنتیک ترکیبی. تحقیق در عملیات در کاربردهای آن، ۱۱(۳)، ۱۳۷-۱۲۵.

[3] Dhingra, S., Bal Gupta, S., Biswas, B., (2014). Genetic Algorithm Parameters Optimization for bi-Criteria Multiprocessor Task Scheduling Using Design of Experiments. World Academy of Science, Engineering and Technology International Journal of Computer Electrical Automatian control and information, 8.

[4] Zahorjan, J., McCann, C., (1990). Processor Scheduling in Shared Memory Multiprocessors Performance Evaluation Review, 18(1), 214-225.

[5] Ercal, F., (1988). Heuristic approaches to talk allocation for parallel computing. Doctoral Dissertation, Ohio State University.

[6] DeJong, K. A., Spears, W. M., (1989). Using Genetic Algorithms to Solve NP-complete Problems. Proceedings of the 3rd International Conference on Genetic Algorithms, 124-132.

[7] Holland, J. H., (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

[8] Thomas, J., Naughton, A. J., (2005). Dynamic task schedulingusing genetic algorithms for heterogeneous distributed computing. Proceedings of the 19th IEEE/ACM International parallel and distributed processing symposium, Denver USA. 207-1530.

[9] Kim, E. S., Sung, C. S., Lee, I. S., (2009). Scheduling of parallel machinesto minimize total completion time subject to S-precedence constraints. The Journal of Computers and Operations Research, 36, 689-710.

[10] Wu, A. S., Jin, Yu. H. S., Lin, K. C., Schiavone, G., (2004). An incremental genetic algorithm approach to multiprocessor scheduling. IEEE Transactions on Parallel and Distributed Systems, 15(9), 824–834.

[11] Cena, M., Crespo, M., Gallard, R., (1995). Transparent remote execution in LAHNOS by means of a neural network device. ACM Press Operating Systems Review, 29(1), 17-28.

[12] Flower, J., Otto, S., Salama, M., (1987). Optimal mapping of irregular fininte element domains to parallel processors, 292.

[13] Fox, G. C., (1988). A review of automatic load balancing and decomposition method for the hipercube, In M. Shultz ed. Numerical algorithms for modern parallel computer architectures, Springer Verlag, 63-76.

[14] Graham, R. L., (1972). Bounds on multiprocessing anomalies and packing algorithm. Proceedings of the AFIPS (1972) Spring Joint Computer Conference, 205-217.

[15] Kidwell, M., (1993). using genetic algorithm to schedule task on bus based system, Proceeding of the Fifth International Conference on Genetic Algorithms, 368-374.

[16] Fox, G. C., Kolawa A., Williams, R., (1987). The implementation of a dynamic load balancer. 2nd Conference on Hipercube multiprocessors, 114-121.
[17] Mansour, N., Fox, G. C., (1991). A hybrid genetic algorithm for task allocation in multicomputers. Proceedings of the Fourth International Conference on Genetic Algorithms, 466-473.
[18] Dutot, P. F., Mounie, G., Trystram, D., (2004). Scheduling Parallel Tasks Approximation Algorithms.
[19] Lodi, A., Martello, S., Monaci, M., Two dimensional packing problems. A survey European.
[22] Bruns, R., (1993). Direct choromosome representation and advanced genetic operators for production scheduling. Proceedings of the Fifth International Conference on Genetic Algorithms, 352-359.