روش نوین زمان‌بندی ایستای کارها در سیستم‌های توزیع شده ناهمگن با استفاده از الگوریتم ژنتیک و شبیه‌سازی گداختگی

امیر مسعود رحمانی [۱]، مجتبی رضوانی [۲]

چکیده

مساله تطبیق و زمان‌بندی ایستای کارها در سیستم‌های توزیع شده محاسباتی ناهمگن به دلایل استفاده بهینه از ماشینهای محاسباتی موجود و همچنین صرف زمان کمتر برای اجرای الگوریتم زمان‌بندی، از اهمیت ویژه‌ای برخوردار است. حل این مساله با استفاده از الگوریتم‌های کلاسیک چون برنامه نویسی پویا و برگشت به عقب نیاز به زمان محاسبه زیادی دارد، به همین دلیل تلاشهای بسیاری برای حل آن با استفاده از روشهای ابتکاری صورت پذیرفته است؛ یکی از این روشهای ابتکاری، الگوریتم ژنتیک است. در این مقاله الگوریتم ژنتیک جدیدی بنام TDGASA ارائه می‌شود که زمان محاسبه آن وابسته به تعداد کارهای موجود در هر مساله زمان‌بندی است و برای کاهش زمان محاسبه الگوریتم از شبیه‌سازی گداختگی استفاده می‌نماید. با انجام شبیه‌سازی‌ها مشاهده می‌شود که الگوریتم پیشنهادی، زمان محاسبه برای زمان‌بندی ایستای کارها را به نحو محسوسی کاهش می‌بخشد در حالی که زمان پاسخ یا زمان اتمام آخرین کار در سیستم نیز کاهش اندکی می‌یابد.

کلمات کلیدی

الگوریتم ژنتیک، زمان‌بندی ایستای کارها، سیستم‌های توزیع شده محاسباتی ناهمگن، شبیه‌سازی گداختگی، الگوریتم ژنتیک وابسته به تعداد کارها با استفاده از شبیه‌سازی گداختگی TDGASA[۱].

A new Static Task Scheduling in Distributed Heterogeneous Systems using Genetic Algorithm and Simulating Annealing

Amir Masoud Rahmani, Mojtaba Rezvani

Abstract

The task matching and static task scheduling problem in distributed heterogeneous computing systems is very important because of optimal usage of available computing machines and accepted CPU time for scheduling algorithm. Solving this problem using the dynamic programming and the back tracking needs much more time. Therefore, there are more attempts to solve it using the heuristic methods; one of the heuristic methods is Genetic Algorithm (GA). In this paper, a new genetic algorithm named TDGASA is presented which its calculated time is depended on the number of tasks in the scheduling problem. Then, the calculated time of the new algorithm is improved by Simulating Annealing (SA). The results show that the proposed algorithm decreases the calculated time of static task scheduling, however, the response time or the completion time of the last task in the system will decrease a little.

Keywords

Genetic algorithm, static task scheduling, distributed heterogeneous computing systems, simulated annealing, (TDGASA) Task Depended Genetic Algorithm using Simulated Annealing.

[1] استادیار و عضو هیأت علمی دانشگاه آزاد اسلامی واحد علوم و تحقیقات تهران، دانشکده فنی و مهندسی، گروه کامپیوتر. Rahmani@sr.iau.ac.ir
[2] Mrezvani83@yahoo.com

۱- مقدمه

کارهای محاسباتی پیچیده نمی‌توانند در یک بازه زمانی قابل قبول بر روی یک ماشین محاسباتی اجرا شوند و به همین دلیل بایستی آنها را به زیرکارهای کوچکتری تقسیم نمود. برای اجرای این زیرکارها می‌توان از سیستم‌های چند پردازنده‌ای گران قیمت و یا از سیستم‌های محاسباتی توزیع شده بهره جست. انتخاب دوم به دلیل برخورداری از نسبت بالاتر کارایی به هزینه ارجحیت دارد. از طرفی در بسیاری مواقع به علت محدودیت‌های پردازشی سیستم‌های چند پردازنده‌ای و یا ماهیت توزیع شدگی کارها، تنها انتخاب موجود، استفاده از سیستم‌های محاسباتی توزیع شده ناهمگن است [۱].

سیستم‌های محاسباتی توزیع شده ناهمگن شامل مجموعه‌ای از ماشینهای محاسباتی با کارایی متفاوت هستند که توسط اتصالات با سرعت بالا به یکدیگر متصل بوده و برای کاربردهای با محاسبات بسیار زیاد و متنوع مناسب هستند [۲].

مساله تطبیق و زمان‌بندی ایستای کارها در یک سیستم توزیع شده ناهمگن، از رده مسائل NP_Hard است. زیرا برای انتساب مجموعه کارهای T به مجموعه ماشینهای H، تعداد تخصیص کارها برابر $|H|^{|T|}$ یعنی تعداد ماشینها به توان تعداد کارها و تعداد حالات ممکن برای ترتیب اجرا برابر $|T|!$است. یکی از اهداف زمان‌بندی کارها بر روی ماشینهای محاسباتی مختلف، کمینه نمودن زمان پاسخ کارها یا به عبارت دیگر زمان اتمام اجرای آخرین کار در سیستم است.

چنانچه تعداد کارها و ماشینهای محاسباتی زیاد باشند، بدست آوردن زمان‌بندی کاملا بهینه برای این مساله بسیار زمانبر بوده و در بسیاری مواقع از اجرای تصادفی کارها نیز زمان بیشتری را طلب می‌نماید. به همین دلیل بایستی به جای روشهای کلاسیک حل مساله مانند روشهای برنامه نویسی پویا و یا برگشت به عقب، از الگوریتم‌های ابتکاری متناسب با شرایط مساله مورد بررسی، استفاده نمود.

الگوریتم‌های ابتکاری در هر مرحله از ساختار عملیاتی خود، از بروز خطاهای شناخته شده جلوگیری نموده و با هدفمندی خاص به سمت جواب صحیح مساله حرکت می‌نماید. هنگامی که یک الگوریتم ابتکاری بتواند از بروز کلیه خطاهای شناخته شده جلوگیری نماید، می‌تواند به عنوان الگوریتمی مناسب برای مساله مورد بررسی مطرح شود [۳]. روشهای ابتکاری بسیاری برای حل مساله زمان‌بندی ایستای کارها ارائه شده‌اند. بعضی از این روشها عبارتند از:

Opportunistic Load Balancing (OLB) [4], Minimum Execution Time (MET) [4], Minimum Completion Time (MCT) [4], Genetic Algorithms (GAs) [5-8], Simulated Annealing [9], Tabu Search [7]

یکی از بهترین روشهای ابتکاری، الگوریتم ژنتیک است [۴]. تحقیقات بسیاری در زمینه تطبیق و زمان‌بندی ایستای کارها توسط الگوریتم ژنتیک برای سیستم‌های چند پردازنده‌ای [۳،۸،۱۰،۱۱،۱۲] و سیستم‌های توزیع شده ناهمگن [۳،۵،۶،۱۳،۱۴] انجام شده است. در این مقاله، یک الگوریتم ژنتیک جدید با استفاده از روش شبیه‌سازی گداختگی و کدگشایی زودترین زمان اتمام کار[۲] (EFT) ارائه می‌شود که پارامترهای آن به تعداد کارها وابسته است.

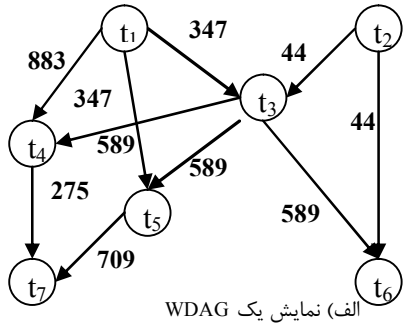
در بخش دوم، مساله زمان‌بندی کارها در یک سیستم محاسباتی توزیع شده ناهمگن بررسی شده و مدل ریاضی برای آن ارائه می‌شود. در بخش سوم، الگوریتم‌های ژنتیک معرفی می‌شود. در بخش چهارم الگوریتم ژنتیک وابسته به تعداد کارها با استفاده از شبیه‌سازی گداختگی (TDGASA) پیشنهاد می‌شود. نتایج شبیه‌سازی و مقایسه الگوریتم‌های پیشنهادی با الگوریتم مبنا در بخش پنجم ارائه می‌شود و در نهایت، نتیجه در بخش آخر آمده است.

۲- مدلسازی مساله زمان‌بندی

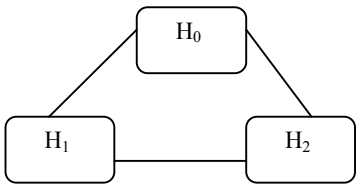
یک برنامه کاربردی که به مجموعه‌ای از کارها شکسته می‌شود، می‌تواند توسط یک گراف جهت دار غیر حلقه‌ای وزن‌دار[۳] WDAG=(T,<,E) مدل شود [۱۳] که در آن $T=\{t_i; i=1,...,n\}$ مجموعه‌ای از n کار، < نشان دهنده یک رابطه ترتیب جزئی[۴] بر روی T و E نیز مجموعه یالهای جهت‌دار و یا کمانها است. برای هر دو کار $t_i, t_j \in T$ وجود یک رابطه ترتیب جزئی $t_i < t_j$ بدان معنی است که کار t_j زمانی قابل اجرا است که کار t_i کامل شود و داده‌های لازم از ماشینی که کار t_i بر روی آن انجام شده به ماشینی که کار t_j بر روی آن زمان‌بندی شده است ارسال شود. به هر کمان از گره t_i به گره t_j عدد صحیحی منتسب می‌شود که نشان دهنده مقدار اطلاعاتی است که بایستی از کار t_i به کار t_j ارسال شود. این مقادیر توسط یک ماتریس n*n بنام D نشان داده می‌شوند که در آن $D_{i,j}$ نشان دهنده مقدار اطلاعات ارسالی از کار t_i به کار t_j است.

اگر سیستم توزیع شده دارای یک مجموعه $H=\{H_k; k=0,...,m-1\}$ از m ماشین محاسباتی متفاوت باشد که توسط یک شبکه با سرعت بالا به یکدیگر متصل شده‌اند، آنگاه ماتریس n*m زمان تکمیل تخمینی[۵] (ECT) به صورتی تعریف می‌شود که در آن $ECT_{i,j}$ نشان دهنده مدت زمان تخمینی اجرای کار t_i در ماشین m_j است.

در شکل (۱-الف) یک WDAG و در شکل (۱-ب) یک سیستم محاسباتی توزیع شده ناهمگن شامل سه ماشین ارائه شده است. جدول (۱) نشان دهنده ماتریس ECT برای گراف شکل (۱-الف) است.



الف) نمایش یک WDAG



ب) یک سیستم توزیع شده ناهمگن با سه ماشین

شکل (۱): الف) نمایش یک WDAG و ب) یک سیستم توزیع شده ناهمگن با سه ماشین

جدول(۱): مقادیر ECT کارها برای سیستم توزیع شده شکل(۱)

ماشینها / کارها	H_0	H_1	H_3
t_1	۸۷۲	۸۹۸	۷۰۸
t_2	۲۵۱	۶۲۴	۷۷۸
t_3	۵۴۲	۷۸۶	۲۳
t_4	۴۰	۷۳۷	۲۵۷
t_5	۷۴۲	۲۴۷	۵۳۵
t_6	۹۷۰	۷۴۹	۷۷۶
t_7	۴۵۷	۴۵۱	۱۰۵

یک ماتریس m*m بنام R، نرخ انتقال داده‌ها را بین ماشینهای مختلف در سیستم نشان می‌دهد. هر ماشین می‌تواند بصورت همزمان کارهای محاسباتی و ارتباطی را انجام دهد. اگر دو کار بر روی یک ماشین زمان‌بندی شوند، هزینه ارتباطی برای ارسال داده‌ها برابر صفر است، در غیر اینصورت چنانچه $D_{i,k}$ برابر مقدار اطلاعاتی باشد که بایستی از کار t_i به کار t_k ارسال شود و $R[H(i),H(k)]$ نشان دهنده سرعت انتقال داده بین دو ماشینی باشد که به ترتیب کارهای t_i و t_k بر روی آنها زمان‌بندی شده‌اند، آنگاه مدت زمان لازم برای انتقال این اطلاعات که هزینه ارتباطی (CommCost) نامیده می‌شود از رابطه (۱) بدست می‌آید.

$$CommCost(t_i,t_k)=\frac{D_{i,k}}{R[H(i),H(k)]} \qquad (۱)$$

با توجه به تعاریف فوق، مساله تطبیق و زمان‌بندی ایستای کارها در یک محیط محاسباتی ناهمگن توزیع شده به یک نگاشت $\Pi: T \rightarrow H$ تبدیل می‌شود. این نگاشت، مجموعه کارهای T را به مجموعه ماشینهای ناهمگن H منسوب می‌نماید به قسمی که زمانهای شروع و پایان اجرای هر کار و همچنین زمانهای آغاز و پایان انتقال داده‌ها بین ماشینها معین شود، محدودیتهای پیش نیازی بین کارها رعایت شود و طول زمان‌بندی[۶] (SL) یعنی بیشینه زمان اتمام اجرای کارها بر روی کل ماشینها، کمینه شود. جواب مساله از رابطه (۲) بدست می‌آید.

$$Answer=\min(SL=\max\{F_j \mid j=0,...,m-1\}) \qquad (۲)$$

که در آن F_j برابر زمان اتمام اجرای آخرین کار زمان‌بندی شده بر روی ماشین H_j است. این زمان شامل زمان محاسباتی، زمان ارتباطی و زمان انتظار به علت محدودیت‌های پیش‌نیازی است.

دو پارامتر دیگر به نامهای b_level و t_level برای هر گره موجود در گراف تعریف می‌شود. b_level برای یک گره یا کار برابر طولانی‌ترین مسیر از این گره به گره‌های برگ است. هنگامی که یک گره دارای فرزند نباشد، b_level برابر متوسط زمان اجرای آن کار در ماشینهای محاسباتی مختلف موجود در سیستم است. t_level برای یک کار برابر طولانی‌ترین مسیر بین این گره و یک گره ریشه در WDAG بدون در نظر گرفتن زمان اجرای آن کار است. در واقع t_level تعیین کننده زودترین زمان ممکن برای شروع اجرای یک کار است. بنابراین اگر کاری دارای والد نباشد، مقدار t_level آن صفر است. جدول (۲) نشان دهنده مقادیر متوسط زمان اجرای هر کار در ماشینهای مختلف (AvgECT)، b_level و t_level برای گراف شکل (۱) است.

جدول(۲): مقادیر AvgECT ، b_level و t_level برای کارهای شکل(۱)

پارامترها / کارها	AvgECT	b_level	t_level
t_1	۸۲۶،۰	۳۷۶۷،۰	۰،۰
t_2	۵۵۱،۰	۳۱۸۹،۰	۰،۰
t_3	۴۵۰،۳۳	۲۵۹۴،۰	۱۱۷۳،۰
t_4	۳۴۴،۶۷	۹۵۷،۳۳	۲۲۱۲،۳۳
t_5	۵۰۸،۰	۱۵۵۴،۶۷	۲۲۱۲،۳۳
t_6	۸۳۱،۶۷	۸۳۱،۶۷	۲۲۱۲،۳۳
t_7	۳۳۷،۶۷	۳۳۷،۶۷	۳۴۲۹،۳۳

۳- الگوریتم‌های ژنتیک

الگوریتم‌های ژنتیک، تکنیک‌های جستجوی تصادفی بر پایه انتخاب و ژنتیک طبیعی هستند. این الگوریتم‌ها برای حل یک مساله با یک مجموعه از راه حل‌های تصادفی به نام جمعیت اولیه آغاز می‌شوند. هر عضو این جمعیت، یک کروموزوم نامیده می‌شود. هر کروموزوم نشان دهنده یک راه حل مورد بررسی است و شامل رشته‌ای از ژن‌ها است. تعداد ژن‌های هر کروموزوم و مفهوم مقادیر آنها به نوع مساله وابسته

است. در الگوریتم‌های این مقاله، تعداد ژن‌های هر کروموزوم برابر تعداد گره‌های موجود در گراف WDAG و مقدار هر ژن، معرف اولویت نسبی کار متناظر با آن گره در زمان‌بندی است (هر کروموزوم نشان دهنده یک زمان‌بندی است)، بطوریکه مقدار اولویت نسبی بیشتر، بیانگر اجرای زودتر کار متناظر است.

مجموعه کروموزوم‌های مورد بررسی در هر تکرار الگوریتم ژنتیک، یک نسل نامیده می‌شوند. کروموزوم‌ها توسط یک یا چند تابع شایستگی مورد ارزیابی قرار گرفته و به منظور تولید نسل بعد، کروموزوم‌های جدید (فرزندان) با اعمال عملگرهای ژنتیک بر روی نسل جاری بدست می‌آیند. این عملگرهای ژنتیک عبارتند از الف) عملگر تبادل[۷] که دو کروموزوم نسل جاری را با یکدیگر ادغام می‌نماید و ب) عملگر جهش[۸] که موجب تغییر در مقادیر بعضی از ژن‌های هر کروموزوم می‌شود. سپس با استفاده از عملگر تکاملی انتخاب و با توجه به مقادیر تابع یا توابع شایستگی کروموزوم‌ها، تعدادی از والدین و فرزندان به نسل بعد انتقال می‌یابند.

مراحل سه گانه تولید، ارزیابی و انتخاب کروموزوم‌ها تا مشاهده شرایط توقف وابسته به مساله تحت بررسی، تکرار می‌شوند. با پایان یافتن الگوریتم، کروموزوم با بهترین مقدار تابع شایستگی به عنوان جواب معرفی می‌شود.

الگوریتم ژنتیک مبنا

الگوریتم ژنتیک معرفی شده در این بخش، به نام الگوریتم مبنا در شکل (۲) نشان داده شده است و دارای چهار گام است [۱۳].

در گام اول پارامترهای زیر از پایگاه داده خوانده می‌شوند: گراف جهت‌دار غیر حلقه‌ای وزن‌دار مربوط به کارها (WDAG)، ماتریس زمان تکمیل تخمینی کارها (ECT)، ماتریس نرخ انتقال داده‌ها (R). پارامترهای اندازه جمعیت[۹] هر نسل (N_p)، تعداد نسلها (N_g)، احتمال تبادل (X_r)، و احتمال جهش (M_r) نیز از کاربر دریافت می‌شوند.

در گام دوم مقادیر b_level و t_level برای هر گره موجود در WDAG محاسبه شده و در جمعیت اولیه برای اولین کروموزوم، به هر ژن مقدار b_level کار متناظرش نسبت داده می‌شود. برای کروموزوم‌های دیگر این نسل، مقدار هر ژن توسط جمع مقادیر این ژن‌ها در کروموزوم اول با یک عدد تصادفی در بازه (-t_level/2 , t_level/2) بدست می‌آید. اگر مقدار ژن i در کروموزوم j توسط نماد G_j^i نشان داده شود، آنگاه G_j^i از رابطه (۳) بدست می‌آید.

$$G_j^i = G_j^i + \text{Random}(-\text{t\_level}/2 , \text{t\_level}/2)) \qquad (۳)$$

زمان پاسخ یا زمان اتمام اجرای آخرین کار در سیستم با استفاده از روش کدگشایی EFT محاسبه شده و به عنوان تابع شایستگی کروموزوم انتخاب شده قرار می‌گیرد. بدین منظور ابتدا ژن‌های هر کروموزوم به ترتیب نزولی مقادیرشان در یک صف مرتب شده و چنانچه این ترتیب با هیچ ترتیب مرتب سازی توپولوژیکی گراف WDAG هماهنگی نداشته باشد – حداقل یکی از روابط پیش ترتیبی رعایت نشده باشد – این کروموزوم به عنوان یک کروموزوم غیر مجاز شناخته شده و مقدار تابع شایستگی منتسب به آن برابر بی‌نهایت می‌شود. در غیر اینصورت کارها به ترتیب از ابتدای صف خارج شده و بر روی ماشینی که زمان اتمام کار بر روی آن از بقیه ماشینها کمتر باشد، زمان‌بندی می‌شوند. سپس بهترین کروموزوم نسل اول در اولین عنصر آرایه Best_Schedule ذخیره می‌شود. طول این آرایه برابر تعداد نسل‌های دریافت شده در ابتدای الگوریتم است.

Step 1. Read the WDAG, ECT, and R from a database and get N_p , N_g , X_r and M_r from the user;
Step 2. Calculate the b-level and the t-level of each task in the WDAG;
Generate Initial Population ($P_{initial}$);
$P_{current} \leftarrow P_{initial}$;
$Schedules \leftarrow Decoding\_heuristic(P_{current})$;
$Best\_Schedule \leftarrow$ evaluate ($Schedules$);
Step 3. while stopping criterion not satisfied do begin
$P_{new} \leftarrow \{\ \}$; /* empty new population */
3-1. repeat for $(N_p/2)$ times
$dad \leftarrow select(P_{current}, Sum\_of\_fitness)$;
$mom \leftarrow select(P_{current}, Sum\_of\_fitness)$;
$P_{new} \leftarrow P_{new} \cup crsovr(dad, mom, child1, child2, X_r)$
endrepeat;
3-2. for each chromosome $\in P_{new}$ do begin
mutate (chromosome, M_r);
endfor;
3-3. $P_{new} \leftarrow P_{new} \cup \{Best\ four\ chromosomes\ of\ P_{current}\}$
$P_{current} \leftarrow P_{new}$;
$Schedules \leftarrow Decoding\_heuristic(P_{current})$;
$Best\_Schedule \leftarrow$ evaluate ($Schedules$);
endwhile;
Step 4. Report the best schedule.

شکل (۲): الگوریتم ژنتیک مبنا

در گام سوم، یک حلقه تا ارضا شرط توقف یعنی به تعداد N_g تکرار می‌شود که دارای سه زیرگام است. در زیرگام اول، یک حلقه برای فراخوانی تابع select به منظور انتخاب کروموزوم‌های پدر و مادر توسط چرخ گردان انجام می‌شود. در این چرخ گردان با احتمال متفاوت[۱۰]، احتمال انتخاب هر کروموزوم متناسب با مقدار تابع تجمعی تابع شایستگی آن است (کروموزوم‌ها از بدترین به بهترین مقدار شایستگی مرتب شده‌اند). سپس عملگر تبادل بر روی دو کروموزوم والد اعمال می‌شود. به منظور تصمیم گیری برای انتقال کروموزوم‌های

والد به نسل جدید و یا تولید دو کروموزوم فرزند و انتقال آنها به نسل جدید، یک عدد تصادفی ما بین صفر و یک تولید می‌شود. اگر این عدد تصادفی کوچکتر یا برابر نرخ احتمال تبادل (X_r) باشد، فرزندان تولید شده و به نسل جدید منتقل می‌شوند، در غیر اینصورت کروموزوم‌های والد به نسل جدید انتقال می‌یابند. به منظور تولید فرزندان، برای هر یک از ژن‌ها یک عدد تصادفی 0<k<1 تولید می‌شود. چنانچه k<0.5 باشد، ژن متناظر با کروموزوم پدر به فرزند اول و ژن متناظر با کروموزوم مادر به فرزند دوم منتقل می‌شود. در غیر اینصورت ژن متناظر با کروموزوم پدر به فرزند دوم و از کروموزوم مادر به فرزند اول منتقل می‌شود. زیرگام اول به تعداد ($N_p/2$) تکرار می‌شود، زیرا در هر تکرار دو کروموزوم به نسل جدید منتقل می‌شود.

در زیرگام دوم به منظور یافتن نقاط جدید در فضای جستجو به گونه‌ای که تنوع جمعیتی حفظ شود و به منظور خارج شدن از دام کمینه محلی از عملگر جهش استفاده می‌شود. این عملگر بر روی کروموزوم‌های حاصل از عمل تبادل با انتخاب یک ژن به صورت تصادفی با احتمال (M_r) و جمع محتوایش با مقداری تصادفی در بازه (-t_level/2 , t_level/2) عمل می‌نماید.

اگر بعد از انجام جهش، مقدار ژنی بیشتر از مقدار (b_level+t_level) آن شود، آنگاه مقدار (b_level+t_level) به آن نسبت داده می‌شود. همچنین اگر مقدار ژن کمتر از مقدار b_level آن شود، مقدار b_level به آن نسبت داده می‌شود.

در زیرگام سوم، با توجه به روش نخبه‌گزینی، چهار کروموزوم با بهترین تابع شایستگی از نسل جاری به نسل جدید منتقل می‌شوند. سپس نسل جدید جایگزین نسل جاری شده و با کدگشایی کروموزوم‌ها، تابع شایستگی هر یک بدست آمده و بهترین کروموزوم در آرایه Best_Schedule ذخیره می‌شود.

پس از تکرارهای حلقه گام سوم تا ارضا شرط توقف و خروج از آن، در گام چهارم آخرین عنصر ذخیره شده در آرایه Best_Schedule به عنوان بهترین جواب زمان‌بندی معرفی می‌شود.

۴- الگوریتم پیشنهادی TDGASA

الگوریتم ژنتیک مبنا ارائه شده در مرجع [۱۳]، برای تمام گراف‌های WDAG با هر تعداد کار، پارامترهای الگوریتم را ثابت فرض می‌نماید. این امر یکی از مشکلات اساسی الگوریتم فوق است زیرا چنانچه تعداد کارها کم باشد، نیازی به تحمل زمان محاسبه بالای الگوریتم نیست و چنانچه تعداد کارهای گراف WDAG زیاد باشد، احتمالا تعداد اعضای جمعیت و تعداد تکرار نسل‌های الگوریتم مبنا برای دستیابی به زمان پاسخ نزدیک به بهینه مناسب نیست. برای حل این مشکل، می‌توان از ایده جدیدی که به تعداد کارهای گراف WDAG وابسته است[۱۱]، استفاده نمود.

در این ایده پیشنهادی، دو پارامتر اصلی الگوریتم، یعنی تعداد اعضای هر نسل و همچنین تعداد نسلها برابر ضریبی از تعداد کارهای هر مساله در نظر گرفته می‌شود. این دو ضریب به ترتیب N_P_Factor و N_g_Factor نامیده می‌شوند.

بدیهی است چنانچه تعداد کارهای موجود در گراف WDAG کم باشد، به علت کوچک شدن دو پارامتر ذکر شده، زمان محاسبه الگوریتم کاهش می‌یابد. اما هنگامی که تعداد کارها زیاد باشد، به علت بزرگ شدن این دو پارامتر، زمان محاسبه نیز افزایش می‌یابد. برای رفع این مشکل، استفاده از شبیه‌سازی گداختگی[۱۲] با ارائه یک روش جدید در چگونگی بدست آوردن زمان مناسب برای اعمال ضرایب سرد شوندگی[۱۳]، پیشنهاد می‌شود. این الگوریتم جدید به نام TDGASA و یا الگوریتم ژنتیک وابسته به تعداد کارها با استفاده از شبیه‌سازی گداختگی نامیده می‌شود.

در الگوریتم TDGASA چنانچه بعد از چندین تکرار، شرایطی مبنی بر همگرایی نتایج حاصل از تکرارها، مشاهده شود، بعضی از پارامترهای الگوریتم به صورت هدفمند تغییر می‌یابند. این امر با کاهش تعداد اعضای جمعیت جاری و نرخ تبادل با ضرب مقادیر کوچکتر از یک در مقدار آنها و همچنین افزایش نرخ جهش با ضرب مقداری بزرگتر از یک در آن صورت می‌پذیرد. با کاهش تعداد اعضای جمعیت جاری و نرخ تبادل، زمان محاسبه الگوریتم کاهش می‌یابد و با افزایش نرخ جهش سعی در جلوگیری از به دام افتادن در کمینه محلی[۱۴] می‌شود. الگوریتم TDGASA همانند الگوریتم ژنتیک مبنا در شکل (۲) است اما گام اول به صورت شکل (۳-الف) تغییر یافته و زیرگام ۳-۴ به صورت شکل (۳-ب) اضافه می‌شود.

Step 1. Read the WDAG, ECT, and R from a database and get *Crossover_Factor*, *Mutation_ Factor*, N_p, N_g, N_p_Factor, N_g_Factor, *Sliding_Window*, X_r M_r, , *Population_ Factor and Comparison_ Factor*, from the user;
$N_p \leftarrow Number\_of\_tasks * N_p\_Factor$;
$N_g \leftarrow Number\_of\_tasks * N_g\_Factor$;

الف) تغییر در گام اول

3-4. Calculate value of CD
if (CD >= Comparison_Base)
/* So, Annealing happens */
$N_p \leftarrow N_p * Populatoin\_Factor$;
$X_r \leftarrow X_r * Crossover\_Factor$;
$M_r \leftarrow M_r * Mutation\_Factor$;
Reset Sliding_Counter to zero;
endif;

ب) اضافه نمودن زیرگام چهارم در گام سوم

شکل (۳): تغییرات الگوریتم TDGASA نسبت به الگوریتم مبنا

در گام اول علاوه بر خواندن WDAG و ماتریس‌های ECT و R از پایگاه داده، دیگر پارامترها نیز از کاربر گرفته می‌شوند.

ضرایب Population_Factor , Mutation_Factor , Crossover_Factor به عنوان ضرایب سرد شوندگی الگوریتم نامیده می‌شوند. چگونگی بدست آوردن زمان مناسب برای اعمال ضرایب سرد شوندگی در ادامه بیان می‌شود.

همانطور که گفته شد، بعد از انجام عملگرهای تبادل و جهش بر روی نسل جاری و ایجاد نسل جدید و همچنین انتقال چهار کروموزوم برتر نسل جاری به نسل جدید بدون اعمال هیچ عملگری، کروموزوم‌های موجود در نسل جدید کدگشایی شده و بهترین جواب بدست آمده در یک آرایه به نام Best_Schedule ذخیره می‌شود. از این رو مقدار شایستگی ذخیره شده در هر عنصر آرایه از عناصر قبلی بهتر است. از آنجا که هدف کمینه نمودن زمان پاسخ (مقدار شایستگی) است، آرایه مذکور، یک آرایه مرتب شده به ترتیب نزولی است.

در زیرگام چهارم، به منظور بررسی همگرایی عناصر موجود در آرایه Best_Schedule برای انجام عمل شبیه‌سازی گداختگی، از روش پیشنهادی زیر استفاده می‌شود: این روش از یک پنجره لغزان که طول آن به تعداد کارهای موجود در گراف WDAG وابسته است، استفاده می‌نماید. طول پنجره لغزان[۱۵] (SWL) از رابطه (۴) بدست می‌آید.

$$SWL = Number\_of\_tasks * Sliding\_Window \qquad (۴)$$

که در آن Number_of_tasks تعداد کارها و Sliding_Window ضریب پنجره لغزان است که مقدار مناسب برای آن در ادامه مشخص می‌شود. چنانچه نقطه شروع پنجره لغزان، مکان i از آرایه Best_Schedule باشد، مقدار درجه همگرایی[۱۶] (CD) از رابطه (۵) بدست می‌آید.

$$CD=\frac{\sum_{k=i}^{SWL+i-1} \text{Best\_Schedule}_k}{SWL * \text{Best\_Schedule}_i} \qquad (۵)$$

با توجه به اینکه مقادیر آرایه Best_Schedule نزولی هستند، همواره مقدار CD کوچکتر یا مساوی یک است. هر چه مقدار CD بدست آمده از رابطه (۵) به یک نزدیک‌تر باشد، مقادیر عناصر مورد بررسی به یکدیگر همگراتر بوده و اعمال ضرایب سرد شوندگی توجیه پذیرتر است. برای بررسی میزان همگرایی عناصر، مقدار CD با پارامتر دیگری به نام مبنای مقایسه[۱۷]، مقایسه می‌شود. چنانچه CD بزرگتر یا مساوی با این مقدار باشد، عمل سرد شوندگی رخ داده و شمارنده پنجره لغزان برابر صفر می‌شود، در غیر اینصورت تکرار مرحله بعد الگوریتم انجام می‌شود. به علت صفر شدن شمارنده پنجره لغزان بعد از انجام عمل سردشوندگی، حداقل فاصله بین دو عمل سرد شوندگی برابر SWL می‌شود.

تعیین مقدار پارامترهای الگوریتم TDGASA

به منظور تعیین مقدار بهینه پارامترهای الگوریتم TDGASA یک مجموعه شبیه‌سازی با استفاده از زبان برنامه‌نویسی #C، بر روی یک کامپیوتر Pentium IV با پردازنده اینتل ۲،۸ مگا هرتز و یک گیگا بایت حافظه انجام شد.

با استفاده از یک برنامه نوشته شده خودکار تولید تصادفی گراف، یک مجموعه گراف، شامل ۳۰ گراف WDAG با تعداد کار و تعداد ماشین متفاوت، به همراه ماتریس‌های وابستگی و زمان اجرای ECT به صورت تصادفی تولید شد. عناصر ماتریس R برای سادگی برابر یک فرض شده‌اند.

برای تعیین مقدار مناسب هر پارامتر، پارامترهای دیگر مساله ثابت نگه داشته شده و مقادیر متفاوتی به پارامتر مورد نظر تخصیص داده می‌شود. از آنجا که زمان محاسبه الگوریتم و زمان پاسخ بدست آمده برای هر مساله وابسته به شرایط همان مساله است، برای مقایسه مقادیر مختلف نسبت داده شده به هر پارامتر، یکی از این مقادیر را به عنوان مقدار پایه قرار داده و نتایج حاصل از مقادیر دیگر پارامتر با آن مقایسه می‌شود.

برای هر گراف با تقسیم زمان پاسخ بدست آمده از شبیه‌سازی‌ها بر زمان پاسخ متناظر حاصل از شبیه‌سازی با مقدار پایه، عددی با دقت چهار رقم اعشار حاصل می‌شود. هر عدد نشان دهنده کیفیت نسبی زمان پاسخ شبیه‌سازی‌ها با مقدار پارامتر مورد نظر به شبیه‌سازی انجام شده با مقدار پارامتر پایه است. با جمع ۳۰ عدد حاصل (هر عدد مربوط به یک گراف WDAG است) و تقسیم آن بر عدد ۳۰، عددی بدست می‌آید که هر چه از یک کوچکتر باشد، نشان دهنده برتری استفاده از مقدار پارامتر مورد نظر نسبت به مقدار پارامتر پایه است و هر چه عدد حاصل از یک بزرگتر باشد، نشان دهنده ضعف انتخاب مقدار پارامتر مورد نظر است. همین مراحل برای زمان محاسبه الگوریتم‌ها نیز تکرار می‌شود. در نهایت، با توجه توام به زمان پاسخ و زمان محاسبه نسبی حاصل از شبیه‌سازی‌ها، مقدار مناسب برای پارامتر مورد بحث تعیین می‌شود. با انجام شبیه‌سازی‌ها، مقدار پارامترها به صورت زیر انتخاب می‌شوند:

Crossover_Factor = ۰،۶; Mutation_Factor = ۱،۱; Population_Factor = ۰،۸; Comparison_Base = ۰،۹۵; Sliding_Window = ۰،۲۵.

۵- نتایج شبیه‌سازی

برای مقایسه دو الگوریتم زمان‌بندی ژنتیک مبنا و TDGASA، همانند بخش قبل شبیه‌سازی‌ها بر روی یک کامپیوتر

Pentium IV با پردازنده اینتل ۲،۸ مگا هرتز و یک گیگا بایت حافظه انجام شد. ۱۵ گراف به صورت تصادفی برای تعداد کارهای مابین صد تا دویست و تعداد چهار تا نه ماشین و با مشخصات جدول (۳)، توسط برنامه خودکار تولید گراف ایجاد شد. مقادیر ماتریس ECT به صورت تصادفی ایجاد شد و عناصر ماتریس R برابر یک فرض شدند. مقادیر پارامترهای X_r و M_r برای هر دو الگوریتم به ترتیب برابر ۰،۶ و ۰،۰۵ و مقادیر N_p و N_g برای الگوریتم مبنا به ترتیب برابر ۵۰۰ و ۱۰۰۰ انتخاب شدند. به منظور انجام شبیه‌سازی در شرایط تقریبا یکسان، در الگوریتم TDGASA مقادیر اولیه N_P_Factor و N_g_Factor به ترتیب برابر ۳ و ۶ انتخاب شدند تا در صورت ضرب در تعداد کارهای هر مساله (مابین صد تا دویست)، مقادیر N_p و N_g در محدوده الگوریتم مبنا حاصل شود.

دیگر پارامترهای الگوریتم TDGASA نیز برابر مقادیر بدست آمده در انتهای بخش چهارم است.

جدول (۳): مشخصات گراف‌های مورد استفاده برای مقایسه الگوریتم‌ها

شماره گراف	تعداد کار	تعداد ماشین	تعداد وابستگی
۱	۱۰۴	۴	۵۶۳
۲	۱۱۷	۹	۴۲
۳	۱۲۴	۷	۱۷۷۹
۴	۱۴۰	۹	۳۷۸۱
۵	۱۴۲	۷	۹۳۳۲
۶	۱۵۱	۵	۷۷۶۸
۷	۱۵۵	۵	۶۹۵۵
۸	۱۵۷	۵	۵۹۴۶
۹	۱۵۸	۸	۳۲۸۲
۱۰	۱۶۲	۶	۵۴۱۶
۱۱	۱۷۹	۶	۶۷۶
۱۲	۱۸۰	۶	۴۵۶۲
۱۳	۱۸۹	۶	۶۵۱۷
۱۴	۱۹۹	۵	۵۶۶۵
۱۵	۱۹۹	۴	۱۵۲۰۵

هر یک از دو الگوریتم سه بار برای هر گراف اجرا شد. جدول (۴) میانگین زمان پاسخ و میانگین زمان محاسبه الگوریتم‌های مختلف را بر حسب ثانیه و پس از انجام زمان‌بندی برای گراف‌های جدول (۳) نشان می‌دهد. با مبنا قرار دادن زمانهای الگوریتم TDGASA، نسبت میانگین زمان محاسبه و زمان پاسخ الگوریتم دیگر به این الگوریتم برای هر گراف بدست آمد. آنگاه با جمع مقادیر نسبت‌های بدست آمده (سطر جمع نسبت‌ها) و تقسیم حاصل جمع بر تعداد گراف‌ها، میانگین این نسبت‌ها (سطر میانگین نسبت‌ها) حاصل می‌شود.

همانطور که نتایج نشان می‌دهد میانگین زمان محاسبه برای الگوریتم TDGASA نسبت به الگوریتم ژنتیک مبنا کاهش چشمگیری در حدود ۸۳ درصد یافته است. این در حالی است که میانگین زمان پاسخ نیز اندکی کاهش یافته است.

جدول (۴): نتایج بدست آمده توسط الگوریتم‌های مختلف برای گراف‌های جدول (۳)

شماره گراف	الگوریتم مبنا		TDGASA	
	میانگین زمان پاسخ (s)	میانگین زمان محاسبه (s)	میانگین زمان پاسخ (s)	میانگین زمان محاسبه (s)
۱	۲۲۷۱۶	۲۶۱	۷۵	۲۲۷۱۶
۲	۱۸۱۰	۱۵۷۶	۶۳۶	۱۷۷۲
۳	۳۲۷۳۲	۴۰۲	۲۲۰	۳۳۲۶۳
۴	۵۰۳۳۴	۶۸۴	۳۲۷	۵۱۱۸۶
۵	۲۲۰۸۶۹	۹۳۵	۵۷۷	۲۲۰۸۶۹
۶	۲۵۶۱۲۹	۱۱۹۶	۵۸۳	۲۵۶۱۲۹
۷	۲۱۸۸۶۹	۹۴۰	۴۷۷	۲۱۰۱۴۸
۸	۱۹۷۹۲۹	۸۹۵	۴۶۰	۱۹۷۸۰۸
۹	۵۰۸۰۸	۷۲۱	۵۰۶	۵۱۰۴۰
۱۰	۱۰۸۳۹۳	۹۱۵	۴۹۳	۱۰۹۰۵۰
۱۱	۱۳۶۲۹	۹۰۸	۶۰۲	۱۳۲۰۸
۱۲	۸۴۴۴۴	۱۰۳۹	۶۸۳	۸۴۱۳۲
۱۳	۱۲۳۱۰۰	۱۲۳۴	۸۹۱	۱۲۲۵۰۸
۱۴	۱۴۵۳۱۸	۱۲۷۹	۱۰۱۴	۱۵۱۱۹۸
۱۵	۵۵۳۶۳۷	۱۸۱۲	۱۶۳۸	۵۵۳۶۳۷
جمع نسبت‌ها	۱۵،۰۲۱۹	۲۷،۵۲۸۲	۱۵	۱۵
میانگین نسبت‌ها	۱،۰۰۱۴	۱،۸۳۵۲۰	۱	۱

۶- نتیجه

مساله تطبیق و زمان‌بندی کارها در یک سیستم توزیع شده ناهمگن یک مساله از رده مسائل NP-Hard است. از این رو به جای استفاده از روشهای کلاسیک حل این مساله مانند برنامه نویسی پویا و برگشت به عقب، از روشهای ابتکاری حل مساله که جواب نزدیک به بهینه را در یک بازه زمانی قابل قبول بدست می‌آورند، استفاده می‌شود. در این میان الگوریتم ژنتیک به دلیل برخورداری از پتانسیل بالا در حل مسائل پیچیده مورد توجه خاصی قرار گرفته است.

در این مقاله یک الگوریتم ژنتیک جدید که تعداد اعضای جمعیت و تعداد تکرارهای آن وابسته به تعداد کارها است، ارائه شده و با استفاده از شبیه‌سازی گداختگی، زمان محاسبه این الگوریتم کاهش یافته است.با وجود آنکه معمولا یک نقطه مصالحه بین زمان محاسبه و زمان پاسخ برای الگوریتم‌های ژنتیک وجود دارد، اما با استفاده مناسب و به جا از شبیه‌سازی گداختگی، هر چند که زمان محاسبه کاهش چشمگیری داشته اما زمان پاسخ یا زمان اتمام آخرین کار نیز افزایش نمی‌یابد. در نهایت، این الگوریتم می‌تواند بدون اعمال تغییرات اساسی در بسیاری از زمینه‌های مرتبط دیگر چون مهندسی صنایع، کنترل فرآیندها، اقتصاد و تحقیق در عملیات به کار گرفته شود.

زیرنویس‌ها

[1] Task Dependent Genetic Algorithm using Simulated Annealing
[2] Earliest Finish Time
[3] Weighted Directed Acyclic Graph
[4] Partial Order
[5] Estimated Completion Time
[6] Scheduling Length
[7] Crossover
[8] Mutation
[9] Population size
[10] Biased roulette wheel
[11] Task Dependent
[12] Simulated Annealing
[13] Cooling factors
[14] Local minimum
[15] Sliding Window Length
[16] Convergence Degree
[17] Comparison Base

مراجع

[1] Tanenbaum, A. S., *Modern Operating Systems*, Prentice Hall, 1992.
[2] Watson, D.W., Antonio, J. K., Siegel, H. Gupta, J., R., and Atallah, M.J., *"Static matching of ordered program segments to dedicated machines in a heterogeneous computing environment"*, Proceedings of the Heterogeneous Computing Workshop, pp. 24–37, April 1996.
[3] Haupt, R.L., Haupt, S.E., *Parallel genetic algorithms*, John willy & Sons, 2004.
[4] Armstrong, R., Hensgen, D., and Kidd, T., *"The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions"*, 7th IEEE Heterogeneous Computing Workshop (HCW '98), pp. 79-87, 1998.
[5] Ali, S., Braun, T. D., Siegel, H. J., and Maciejewski, A. A., *Heterogeneous computing, in Encyclopedia of Distributed Computing*, Kluwer Academic, Norwell, MA, 2001.
[6] Braun, T. D., Siegel, H. J. and Beck, N., *"A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems"*, Journal of Parallel and Distributed Computing Vol. 61, pp. 810-837, 2001.
[7] Naharai, B., Youssef, A., and Choi, H. A., *"Matching and scheduling in a generalized optimal selection theory"*, Proceedings of the Heterogeneous Computing Workshop, pp. 3–8, April, 1994.
[8] Shenassa, M. H., Mahmoodi, M., *"A novel intelligent method for task scheduling in multiprocessor systems using genetic algorithm"*, journal of Franklin Institute, Elsevier, pp. 1-11, 2006.
[9] Shroff, P., Watson, D., Flann, N., and Freund, R., *"Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments"*, 5th IEEE Heterogeneous Computing Workshop (HCW '96), pp. 98-104, 1996.
[10] Auyeung, A., Gondra, I. and Dai, H.K. *"Multi-heuristic List Scheduling Genetic Algorithm for Task Scheduling"*, Proceedings of the Eighteenth Annual ACM Symposium on Applied Computing, ACM Press, pp. 721-724, 2003.
[11] Davidovic T., Crainic T.G. , *Benchmark-Problem Instances for Static Scheduling of Task Graphs with Communication Delays on Homogeneous Multiprocessor Systems*, C.R.T.'s publications, 2004.
[12] Lee, Y.H., Chen, C., *"A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems"*, the 9th workshop on compiler techniques for high-performance computing, 2003.
[13] Dhodhi, M. K., Ahmad, I., Yatama, A. and Ahmad, I., *"An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems"*, Journal of Parallel and Distributed Computing, Vol. 62, pp. 1338–1361, 2002.
[14] Radulescu, A., Gemund, A.van. *"Fast and effective task scheduling in heterogeneous systems"*, Proceeding of Heterogeneous Computing Workshop, 2000.