مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۸ ، زمستان ۱۳۹۶، ص ۶۹- ۵۳

ISSN: 2745-3673
http://www.Rcsj.ir

مجلـه‌ی علمی پژوهش در علوم رایانه (سال دوم)
شماره ۸ / زمستان ۱۳۹۶ / ص ۶۹-۵۳

الگوریتم های انتخاب منبع در راستای برنامه ریزی اقتصادی در سیستم های توزیع شده

مارال کمرپور [۱] ، مرتضی زلف پور [۲] ، شیما ابراهیمی نژاد [۳]

[۱] دانشجوی کارشناسی ارشد تجارت الکترونیک، واحد سپیدان، دانشگاه آزاد اسلامی، سپیدان، ایران.
[۲] استادیار گروه کامپیوتر و فناوری اطلاعات، واحد سپیدان، دانشگاه آزاد اسلامی، سپیدان، ایران.
[۳] دانشجوی کارشناسی ارشد تجارت الکترونیک، واحد سپیدان، دانشگاه آزاد اسلامی، سپیدان، ایران.

نام نویسنده مسئول:
مارال کمرپور

چکیده

در این مقاله، در راستای برنامه‌ریزی دسته‌ای شغلی مستقل در محاسبهٔ توزیعی مستقل با منابع غیراختصاصی، به ارائهٔ الگوریتم‌های انتخاب اسلات در مدل‌های اقتصادی می‌پردازیم. رویکردهای موجود به منابع مشترک تخصیص یافته و برنامه‌ریزی شغلی چندپردازنده در مدل‌های اقتصادی محاسبات توزیعی گرایش داشته و بر جستجوی اسلات‌های زمانی در برنامه‌های اشغال منابع مبتنی شده است. اسلات‌های زمانی پویشی باید با الزامات محدودهٔ لازم، ویژگی‌های منبع محاسباتی و هزینه مطابقت داشته باشد. به طور معمول چنین روش‌های برنامه‌ریزی، صرفاً گزینهٔ مناسب مجموعهٔ اسلات زمانی را مدنظر قرار می‌دهند. این مقاله، زمانبندی را نشان می‌دهد که حاکی از جستجوی چندین گزینه‌ای است. دو الگوریتم پیچیدگی خطی برای جستجوی گزینه های جایگزین با یکدیگر مقایسه شده است. دارا بودن چندین پیکره‌بندی منبع اختیاری برای هر شغل، فرصتی را برای انجام بهینه سازی اجرای تمام دسته‌های مشاغل و افزایش بهره‌وری کلی برنامه‌ریزی مهیا می‌کندو در همین راستا به ارائه یک نمونه از الگوریتم های بهینه سازی موازی که کمک شایانی به انتخاب بهترین گزینه به منظور کاهش زمان و هزینه در مدیریت منابع در مدل های اقتصادی در سیستم های توزیع شده دارد میپردازیم. مسئله تخصیص وظایف به پردازنده ها معروف به مسئله تخصیص یا مسئله نگاشت می باشد. دربخش معرفی الگوریتم موازی، تخصیص وظایف به پردازنده ها به صورت استاتیکی یا ایستا انجام شده و هدف اصلی تخصیص مقدار برابری بار به کلیه پردازنده ها و کاهش سربار برهم کنش میان آنها می باشد.

واژگان کلیدی: الگوریتم، مدل اقتصادی، مدیریت منابع

ISSN: 2745-3673
http://www.Rcsj.ir

مقدمه

دراین مقاله تلاش شده تا با پژوهشی بر دو موضوع مرتبط گامی در جهت شناسایی و ارائه بهترین روش ها و معرفی و مقایسه الگوریتم های بهینه سازی و اسلات زمانی برای مدیریت بهتر زمان و هزینه در سیستم های توزیع شده برداشته شود.

طرح اصلی زمانبندی

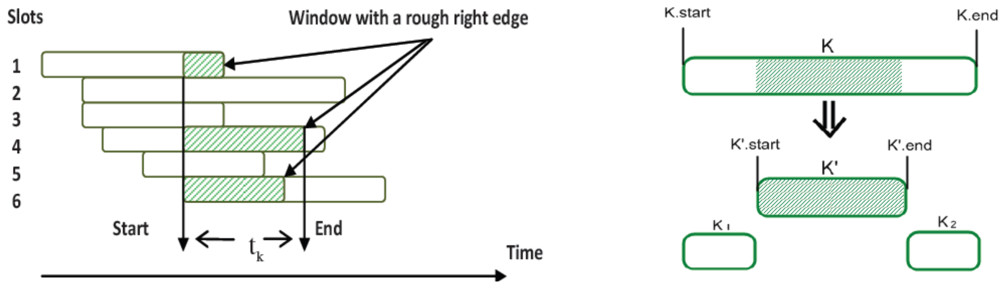
یافتن مجموعهٔ اسلات‌های زمانی، زمانبندی شغلی محسوب می‌شود. الزامات منبع تنظیم شده در درخواست منبع مشتمل بر زمان استفاده t و مشخصه‌های گره‌های محاسباتی (سرعت ساعت، حجم RAM، فضای حافظه و سیستم عملیاتی و غیره) است.

فرض کنید $J = \{j_{1,\dots,j}\}$ حاکی از دستهٔ مشتمل بر n شغل باشد. مجموعهٔ اسلات با شغل $j_i , i = 1, \dots , n$ تطبیق می‌یابد اگر این روند الزامات تعداد و نوع منابع، هزینهٔ c_i و زمان اجرای شغل t_i را مرتفع نماید. فرض می‌کنیم که برای هر شغل j_i در چرخهٔ زمانبندی فعلی، دست کم یک مجموعهٔ مناسب s_i وجود داشته باشد. در غیر این صورت، زمانبندی شغلی تا از سرگیری مجدد به تعویق می‌افتد. هر مجموعهٔ اسلات، برای اجرای شغل i ام در یک دستهٔ J ={ $j_{1,\dots,j}$} با یک جفت پارامتر هزینه $c_i(s_i)$ و زمان $t_i(s_i)$ مربوط به کاربرد منبع تعریف می‌شود و $c_i(s_i)$ حاکی از هزینهٔ اسلات‌ها در مجموعه و $t_i(s_i)$ نشان دهندهٔ زمان اجرای شغل i ام است.

در خلال هر چرخهٔ زمانبندی دسته‌ای، دو مسئله باید مرتفع شود.

۱. انتخاب مجموعه‌های آلترناتیو اسلات‌ها (آلترتانیوها) که الزامات (منبع، زمان و هزینه) را مرتفع می‌نمایند.
۲. انتخاب مجموعه‌ای که برحسب اجرای دسته‌ای شغل در چرخهٔ زمانبندی فعلی کارآ یا بهینه باشد.

برای تحقق بخشیدن طرح زمانبندی تشریح شده در بالا، ابتدا به طرح الگوریتم یافتن مجموعه‌ای از اسلات‌های آلترناتیو نیاز داریم. اسلات‌ها به واسطهٔ زمان شروع در دستور غیر کاهشی مرتب شده‌اند (شکل ۱ (الف)). در مورد گره‌های متجانس، مجموعه‌ای از اسلات‌ها برای یافتن هر شغل با پنجرهٔ مستطیلی ارائه شده است. در مورد CPU ها با عملکرد متفاوت، این روند پنجره‌ای با rough right edge خواهد بود و زمان استفاده برحسب زمان اجرای t_k بخش شغلی (کار) تعریف شده که از کمترین CPU بهره می‌گیرد – شکل ۱ (الف) را ببینید.



شکل ۱- انتخاب اسلات برای منابع غیرمتجانس: فهرست مرتب‌شدهٔ اسلات‌های موجود (الف)؛ کاهش اسلات (ب)

این طرح به صورت تکراری کار می‌کند و در خلال تکرار، به دنبال آلترناتیوی واحد برای هر شغل دسته‌ای خواهد بود. در مورد انتخاب اسلات موفق به ازای شغل i ام، فهرست اسلات‌های مشاهده شده برای شغل (i+1)ام تغییر یافته است. تمام بازه‌های زمانی که درگیر آلترناتیو شغلی i ام هستند از فهرست اسلات‌های خالی جدا شده‌اند (شکل ۱ (ب)). انتخاب اسلات‌ها به ازای هر شغل i+1)(ام، بر حسب لیست اصلاحی با روش تشریح شدهٔ فوق انجام شده است. برای نمونه، فرض کنید که اسلات k' در اسلات‌های پنجره‌ای مناسب وجود دارد. در نتیجه، زمان شروع آن با زمان شروع پنجره برابر خواهد بود: زمان شروع k' = پنجره. زمان شروع و زمان پایان آن برابر است با $k'.end = k'.start + t_k$، که در آن t_k ارزیابی زمان اجرای کار در گرهٔ CPU یی است که اسلات بدان اختصاص یافته است. اسلات K' باید از فهرست اصلی اسلات‌های سیستمی موجود کاسته شود. در ابتدا لازم است تا اسلات k – اسلات را بیابیم، بخشی از k'

که در آن وقفه k' از k حذف می‌شود. بنابراین، به طور کلی به حذف اسلات k' از فهرست اسلات مرتب شده نیاز داشته و دو اسلات جدید k_1 و k_2 را لحاظ می‌کنیم. زمان‌های شروع و خاتمهٔ آنها بدین صورت تعریف شده است:
$k_1.start\ time = k.start\ time$. اسلات‌های k_1 و k_2 باید به فهرست اسلات تعیین شده که برحسب ترتیب زمان شروع غیرکاهشی آمده فهرست شوند (شکل ۱ (الف) را ببینید). اسلات k_1 جایگاه مشابهی با اسلات k در این فهرست خواهد داشت، زیرا هر دو دارای زمان شروع مشابه هستند. چنانچه اسلات‌های k_1 و k_2 دارای بازهٔ زمانی صفر باشند، لزومی به افزودن آنها به این فهرست نیست. الگوریتم‌ها پس از پردازش آخرین مشاغل، تکرار بعدی را از شروع دسته آغاز کرده و سعی می‌کنند آلترناتیوهای دیگری را در فهرست اسلات‌های اصلاح شده بیابند. آلترناتیوها هیچ تقاطعی با زمان پردازش پیدا نکرده و بنابراین هر شغل می‌تواند به تعدادی از مجموعه اسلات‌های یافته شده بدون تجدید

سایر معادل‌های شغلی اختصاص یابد. جستجو برای آلترناتیوها هنگامی که الگوریتم نتواند هیچ مجموعه اسلات جایگزینی را در فهرست جاری اسلات‌ها بابت هریک از مشاغل دسته‌ای بیابد خاتمه خواهد یافت.

روش بهینه‌سازی فاز دوم این طرح زمانبندی، در [۲] آمده است. این روند از طریق روش‌های برنامه‌ریزی دینامیک با استفاده از معیارهای چندگانه منطبق با خط مشی اقتصادی VO اجرا شده است.

ما دو معیار را در چارچوب مدل خود در نظر می‌گیریم. هزینه اجرا و معیارهای زمانی برای هر دستهٔ شغلی J با استفاده از مجموعه اسلات مناسب $\bar{s} = (s_1, \ldots, s_n)$ وجود دارد. نخستین گروه معیاری مشتمل بر هزینهٔ کلی اجرای دستهٔ شغلی c$(\bar{s}) = \sum_{i=1}^{n} c_i(s_i)$ است. خط مشی اجرای VO تا حدودی منافع کاربران با استفاده از معیار زمان اجرا برای تمام مشاغل دستهٔ t$(\bar{s}) = \sum_{i=1}^{n} t_i(s_i)$ ارائه شده است. در راستای منع انحصارطلبی برخی منابع به کار رفته از سوی کاربران، حد b^* در بودجهٔ VO در نظر گرفته شده است که مقدار بیشینهٔ کل هزینهٔ کاربرد منابع در چرخهٔ زمانبندی جاری محسوب می‌شود. زمان کل اشغال اسلات‌ها t^* تمایل مالکان در قبال تعادل تقسیم شغلی (داخلی) محلی و (خارجی) جهانی را نشان می‌دهد.

فرض کنید $g_i(s_i)$ تابع خاص باشد که در آن کارآیی کاربرد مجموعه اسلات s_i برای هر شغل i ام تعیین می‌شود. به عبارت دیگر، $g_i(s_i) = c_i(s_i)$ یا $g_i(s_i) = t_i(s_i)$ باشد. فرض کنید $f_i(z_i)$ مقدار انتهایی معیار خاص با استفاده از مجموعهٔ اسلات s_i برای اجرای مشاغل $i, i+1, \ldots, n$ باشد که در آن z_i به عنوان زمان کل اشغال یا هزینه استفاده از اسلات‌ها بابت مشاغل $s_i, s_{i+1}, \ldots, s_n$ باشد. فرض کنید مقدار زمانی قابل قبول یا هزینهٔ اشغال اسلات به صورت $z_i(s_i)$ باشد. در نتیجه، داریم $z_i s_i \leq z_i \leq z^*$ که $Z*$ همان حد معین است. برای نمونه اگر $z_i s_i = t_i(t_i)$ باشد در نتیجه $t_i s_i \leq t_i \leq t^*$ که در آن T_i عبارت از کل زمان اشغال اسلات‌ها در هر شغل $i, i+1, \ldots, n$ است و t^* حد مقادیر t_i است که با در نظر گرفتن تعادل بین جریان شغلی جهانی (انتخاب کاربر) و جریان شغلی محلی (انتخاب مالک) انتخاب شده است. برای نمونه اگر $z_i(s_i) = c_i(s_i)$ باشد، در نتیجه داریم $c_i s_i \leq c_i \leq b^*$ که در آن c_i هزینه کلی کاربرد منابع برای کارهای $i, i+1, \ldots, n$ و b^* بودجهٔ VO محسوب می‌شود. در طرح خلاف جهت اجرا شده در [۲ $i = 1, z_1 = z^*$]، $1 < i \leq n$ دارای o $\leq z_i \leq z^*$ هستند. معادلهٔ کاربردی برای حصول شرط (با توجه به شرط ($z_i s_i$ مربوط به $f_i s_i$)برای شیوه اجرای خلاف جهت را می‌توان به صورت ذیل نوشت:

$$f_i(z_i) = \underset{s_i}{\operatorname{extr}}\{g_i s_i + f_{i+1} z_i s_i\}, i = 1, \ldots, n, f_{n+1}) \equiv 0. \qquad (1)$$

چنانچه بهینه‌سازی معیار واحد مربوط به اجرای دستهٔ شغلی را در نظر بگیریم، در نتیجه هر معیار $c(\bar{s})$ یا $t(\bar{s})$ باید با حد معین t^* یا b^* به ازای منافع طرف معین ـ کاربر، مالک و مدیر VO [۲] به حداقل برسد.

برای نمونه، حد تعیین شده برای زمان کل اشغال اسلات به واسطهٔ کارها را می‌توان بدین صورت بیان کرد:

$$t^{*} = \sum_{i=1}^{n} \sum_{s} [t_i(s_i)/l_i], \qquad (2)$$

که در آن l_i، تعداد مجموعه‌های اسلات قابل قبول برای شغل i ام است؛ $[\cdot]$ به معنای نزدیک‌ترین به $t_i(s_i)/l_i$ است که عدد صحیح محسوب نمی‌شود.

حد بودجۀ VO (b^{*}) را می‌توان به عنوان درآمد بیشینۀ مالکان منابع برحسب (۱) حد معین t^{*} به دست آورد که با استفاده از معادلۀ (۲) بدین صورت تعریف شده است:

$$b^{*} = \max_{s_i} \{ c_i s_i + f_{i+1}(t_i - t_i(s_i)) \} \qquad (3)$$

برای نمونه، در حال کلی مدل [۲]، لازم است از بردار معیار $< c\bar{s}, d\bar{s}, t\bar{s}, i\bar{s} >$ استفاده کنیم که در آن $d\,\bar{s} = b^{*} - c(\bar{s})$ و $i\,\bar{s} = t^{*} - t(\bar{s})$ است.

الگوریتم‌های جستجوی اسلات

فرض کنید یکی از درخواست‌های منابع با شغلی در دستۀ J همراه باشد. این درخواست‌های منبع بدین صورت تعیین می‌شوند: اسلات‌های زمانی همزمان N ، میزان عملکرد منبع در حداقل P و قیمت بیشینۀ منبع به ازای واحد زمانی که بیشتر از C نبوده و رابطه معکوس با بازۀ زمانی t دارد را ارائه می‌کند. در اینجا، الگوریتم جستجوی اسلات بابت شغل واحد و تغییر منبع به ازای واحد زمانی تشریح شده است. این همان الگوریتم مبتنی بر قیمت داخلی اسلات‌ها (ALP) با محدودیت هزینۀ اسلات‌های مجزا است. ALP رویکردی «متداول» برای انتخاب اسلات است [۷]. با وجود این، فرض می‌کنیم تحلیل این روند مرتبط با روش بهینه (انتخاب ۵) باشد. داده منبع مشتمل بر فهرست اسلات‌های موجود است و این اسلات‌ها برحسب زمان شروع و به صورت صعودی مرتب می‌شوند (شکل ۱ (الف) را ببینید). الگوریتم جستجو به فهرست مرتب‌شده برحسب کارکرد نیاز دارد و اجرای هر اسلات را منوط به اینکه الزامات آن برآورده شده باشد تضمین می‌کند.

۱. اسلات‌ها را برحسب زمان شروع به صورت صعودی مرتب کنید ـ شکل ۱ (الف) را ببینید.

۲. از فهرست نتیجه، اسلات مناسب بعدی s_k استخراج و مورد بررسی قرار گرفت.

اگر شرایط ذیل محقق شود، اسلات s_k مناسب است:

الف) میزان عملکرد منبع $p(s_k) \geq p$؛

ب) طول اسلات (بازۀ زمانی) کافی باشد (بنا به عملکرد واقعی منبع اسلات) $l(s_k) \geq tp(s_k)/p$؛ ج) بار منبع به ازای واحد زمانی $c(s_k) \geq c$.

اگر شرایط (الف)، (ب) و (ج) محقق شوند، اسلات s_k با موفقیت به فهرست پنجره افزوده شده است.

۱. ما زمان جبران d_k مربوط به اسلات k جاری را در رابطه با (k-1) ام به طول پنجره اضافه می‌کنیم.

۲. اسلات‌های که طول آنها با توجه به dk منقضی شده اند را از فهرست حذف می‌کنیم. انقضا بدین معناست که طول اسلات باقیمانده $l'(s_k)$ که مطابق آنچه در مرحله 2b نشان داده شده اندازه‌گیری می‌شود کافی نباشد، با فرض اینکه اسلات k ام برابر با شروع آخرین اسلات افزوده شده باشد: $l' s_k < t - t_{last} - ts_k p(s_k)/p$ که در آن $t(s_k)$ زمان شروع اسلات و t_{last} زمان شروع آخرین اسلات افزوده شده است.

۳. به مرحله ۲ که در آن پنجره دارای N اسلات است بروید.

۴. خاتمۀ الگوریتم

می‌توانیم صرفاً از طریق فهرست اسلات حرکت کنیم. اگر پیش از داشتن اسلات‌های N انباشته شده به آخر رسیدیم این به معنای ناکامی در یافتن پنجره شغلی است و زمانبندی آن به واسطۀ زمانبند متا تا زمان چرخۀ زمانبندی بعدی به تعویق افتاده است. در غیر این صورت، پنجره به عنوان مجموعه اسلات آلترناتیو شغلی انتخاب می‌شود. ALP به صورت چرخه‌ای برای هر شغل در هر

دستهٔ $j = \{j_{1,\dots,j}\}$ اجرا شده است. با جستجوی موفق برای پنجرهٔ شغل j_i ام، فهرست اسلات با کسر اسلات‌های شکل گرفته اصلاح می‌شود (شکل ۱ (ب) را ببینید). بنابراین، اسلات‌های شکل گرفته، در پردازش دستهٔ شغلی بعدی مدنظر قرار نمی‌گیرد.

در مدل اقتصادی [۲]، درخواست منبع کاربرد مشتمل بر پیش‌شرط قیمت منبع بیشینه است که به معنای قیمتی است که کاربر برای پرداخت آن به ازای استفاده از منبع موافق است. با وجود این، حیطهٔ جستجو را کاهش داده و الگوریتم‌های ساخت پنجره را به اسلات‌های پرهزینه محدود می‌سازد. تفاوت الگوریتم پیشنهادی بعدی آن است که در آن از طریق بودجهٔ بیشینه هر شغل، قیمت بیشینه را به ازای پیش‌شرط واحد زمانی C جایگزین می‌کنیم.

این روند، الگوریتم مبتنی بر قیمت بیشینه شغلی (AMP) است. بودجهٔ بیشینه برحسب S=CtN است که در آن t بازهٔ زمانی ذخیره و N تعداد اسلات لازم محسوب می‌شود. در نتیجه، از منظر مخالفت با ALP، هدف جستجو پنجرهٔ تشکیل شده با اسلات‌ها است که در آن هزینه کل فراتر از بودجهٔ بیشینه S نخواهد بود. به عبارت دیگر، AMP از داده منبع یکسان به عنوان ALP بهره می‌گیرد.

فرض کنید که متغیرهای دیگر به صورت ذیل است: ns_{-} تعداد اسلات‌های جاری در پنجره؛ $-m_{-}$ هزینهٔ کل اسلات‌های اولین N.

۱. نخستین پنجره شروع را از طریق اسلات‌های N با استفاده ALP به استثنای شرط 2c بیابید (به شرح ALP فوق مراجعه کنید)

۲. اسلات‌های پنجره را برحسب هزینه به صورت صعودی مرتب کنید.

هزینهٔ کلی اسلات‌های N اول یعنی m_{-} را محاسبه کنید. اگر $M_N \leq S$ باشد، به مرحله ۴ بروید، بنابراین پنجرهٔ نتیجه از طریق اسلات‌های N اول مربوط به پنجرهٔ فعلی شکل گرفته و سایرین به فهرست اسلات منبع باز می‌گردند. در غیر این صورت، به مرحله ۳ بروید.

۳. اسلات مناسب دیگری را مطابق با شرایط 2a و 2b مربوط به ALP اضافه کنید. زمان شروع پنجره جدید را تعیین کرده و انقضای آن را مانند مرحلهٔ ۴ در ALP کنترل کنید.

چنانچه $n_s < n$ باشد، مرحله جاری را تکرار کنید. اگر $n_s \geq n$ باشد، به مرحله ۲ بروید. اگر اسلات‌های این فهرست منقضی شده باشند و $n_s < n$ بود، در نتیجه الگوریتم ما ناموفق بوده و هیچ پنجرهٔ شغلی یافت نشده است.

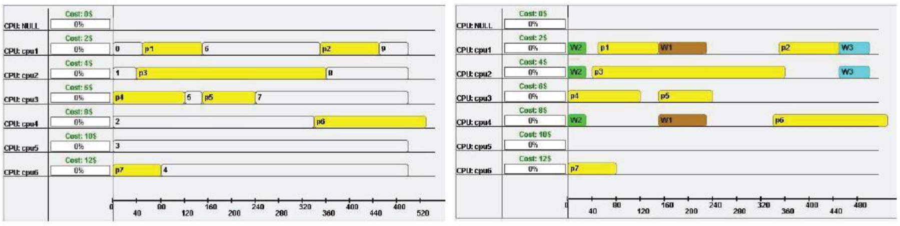
خاتمهٔ الگوریتم

می‌توانیم سه مشخصهٔ اصلی الگوریتم فوق را خاطرنشان شویم. نخست، هر دو الگوریتم، میزان عملکرد منبع را مدنظر قرار دادند. این روند امکان تشکیل پنجره‌های اسلات زمانی را با uneven right edge امکان‌پذیر می‌سازد (فرض می‌کنیم که تمامی اسلات‌های همزمان هر شغل باید به صورت همزمان آغاز شوند). دوم، هر دو الگوریتم حد بیشینهٔ قیمت را مدنظر قرار می‌دهند که از سوی کاربر تحمیل شده است. سوم، هر دو الگوریتم دارای اندازهٔ منابع سیستمی خطی هستند O(m) که در آن m تعداد اسلات‌های زمانی موجود است: ما صرفاً از طریق فهرست جلو می‌رویم و هرگز به معادل‌های پیشین رجوع نمی‌کنیم.

نمونهٔ جستجوی AMP

در این نمونه، به منظور سهولت و سادگی نمایش، مسئله را با مجموعهٔ یکنواختی از منابع در نظر می‌گیریم، بنابراین منبع دارای شکل مسطیلی و بدون rough right edge خواهد بود. فرض کنید که شرایط اولیه محیط محاسباتی توزیعی ذیل وجود داشته باشد. در این مورد، شش گره پردازنده – cpu6 (شکل ۲ الف) وجود دارد. هریک دارای هزینهٔ واحد مختص به خود است (هزینهٔ کاربرد آن به ازای واحد زمانی) که در این ستون برای سمت راست نام پردازنده فهرست شده است. علاوه بر این، هفت کار محلی p1-p7 وجود دارد که در حال حاضر برای اجرا در سیستم تحت بررسی دقیق زمانبندی شده است. اسلات‌های سیستمی موجود به صورت مستطیل‌های ۹-۰ ترسیم شده است ـ شکل ۲ (الف) را مشاهده کنید. اسلات‌ها به واسطهٔ زمان غیرکاهشی شروع و تعداد ترتیبی هر اسلات مرتب شده است که در بدنهٔ آن مشخص شده است. در این راستا، موقعیتی را در نظر می‌گیریم که در آن چرخهٔ زمانبندی مشتمل بر دستهٔ مشاغل سه گانه با الزامات منبع ذیل است.

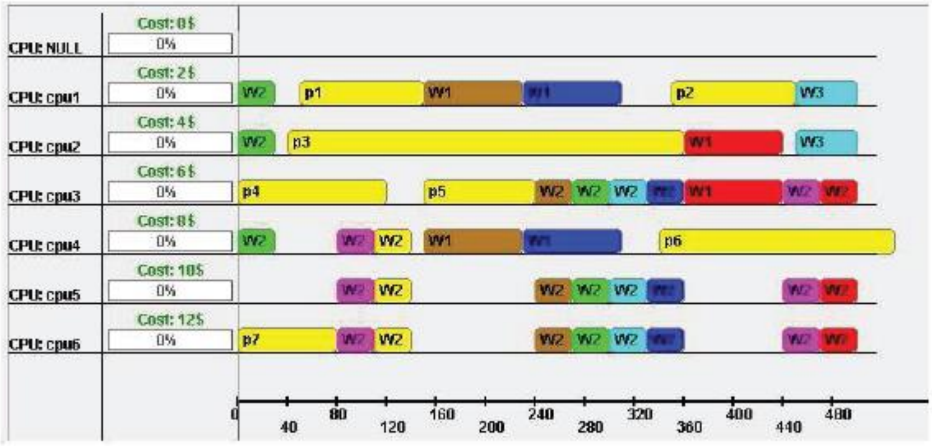
شغل ۱ الزامات : تعداد گره‌های پردازندهٔ مورد نیاز :۳؛ زمان اجرا: ۳۰؛ بیشینهٔ هزینه کل «پنجره» در هر زمان : ۱۰.
شغل ۲ الزامات : تعداد گره‌های پردازندهٔ مورد نیاز :۳؛ زمان اجرا: ۳۰؛ بیشینهٔ هزینه کل «پنجره» در هر زمان : ۳۰.
شغل ۳ الزامات : تعداد گره‌های پردازندهٔ مورد نیاز :۲؛ زمان اجرا: ۵۰؛ بیشینهٔ هزینه کل «پنجره» در هر زمان : ۶.



(الف) (ب)

شکل ۲- نمونهٔ جستجوی AMP : وضعیت اولیهٔ محیط محاسباتی (الف)؛ آلترناتیوهای یافته شده پس از تکرار اولیه (ب)

اول از همه، براساس جستجوی آلترناتیوهای AMP باید فهرست اسلات‌های موجود را تشکیل داده و نخستین آلترناتیو (نخستین پنجره مناسب) را برای نخستین شغل هر دسته بیابیم. آلترناتیو یافته شده برای شغل ۱ (شکل ۲ (ب) را ببینید) دارای دو مستطیل بر روی خطوط منبع CPU1 و CPU4 در بازهٔ زمانی [۲۳۰؛ ۱۵۰] به نام W1 است. هزینه کلی به ازای زمان این پنجره برابر با ۱۰ است. این روند نخستین پنجره احتمالی برآوردن درخواست منبع شغلی محسوب می‌شود. خاطرنشان می‌شود که سایر پنجره‌های احتمالی با زمان شروع قبل از آن با محدودیت هزینهٔ کلی تناسب ندارند. در نتیجه ما به کسر این پنجره از لیست اسلات‌های موجود و یافتن نخستین مجموعهٔ مناسب از اسلات‌ها برای دستهٔ شغلی دوم در لیست اصلاح شده نیاز دارد. علاوه بر این، عملکرد مشابهی برای شغل سوم انجام شده است (شکل ۲ (ب) را ببینید). پنجره‌های آلترانتیو برای هر شغل این دسته‌ها به ترتیب به صورت W1، W2 و W3 نامگذاری شده است. نخستین پنجرهٔ مناسب برای شغل دوم (با احتساب آلترانتیو W1 برای شغل اول) مشتمل بر سه اسلات در خطوط پردازندهٔ CPU1، CPU2 و CPU4 با هزینهٔ کلی ۱۴ بابت هر واحد زمانی است. نخستین آلترناتیو احتمالی بابت شغل سوم همان پنجرهٔ W3 در بازهٔ زمانی [۴۵۰؛ ۵۰۰] است. علاوه بر این، با احتساب آلترناتیوهای یافت شده پیشین، این الگوریتم جستجوی مجموعه اسلات‌های آلترناتیو بعدی را برحسب اولویت شغلی انجام می‌دهد. این الگوریتم به صورت تکراری عمل می‌کند و سعی در یافتن پنجره‌های آلترنایتو هر دسته شغلی در هر تکرار دارد. شکل ۳، نمودار نهایی تمام آلترناتیوهای یافت شده در خلال این جستجو را نشان می‌دهد.



شکل ۳- چارت نهایی تمام آلترناتیوهای یافت شده

خاطرنشان می‌شود که در رویکرد ALP، حد هزینهٔ اسلات‌های جداگانه باید برابر با ۱۰ بابت شغل ۲ باشد (همانگونه که این روند دارای محدودیت هزینه کلی برابر با ۳۰ برای پنجره اختصاص یافته به گره‌های سه پردازشگر است). بنابراین، cpu6 پردازنده با مقدار هزینه کاربردی ۱۲ در خلال جستجوی آلترناتیو با الگوریتم ALP لحاظ نشده است. با وجود این، بدیهی است که در رویکرد AMP ارائه شده، هشت آلترناتیو یافت شده است. آنها از اسلات‌های اختصاص داده شده به خط cpu6 استفاده کرده و بنابراین با حد هزینه کلی پنجره تناسب دارند.

ISSN: 2745-3673
http://www.Rcsj.ir

مطالعات شبیه‌سازی

این آزمایش شامل مقایسهٔ نتایج زمانبندی دستهٔ شغلی با استفاده از مجموعه‌های مختلف اسلات‌های مناسب یافت شده در رویکردهای AMP و ALP فوق‌الذکر است. جستجوی آلترناتیوها در مجموعهٔ مشابهی از اسلات‌های سیستم خالی انجام شده است. در خلال، چرخهٔ زمانبندی شبیه‌سازی شدهٔ واحد، تهیه فهرست مرتب شدهٔ اسلات‌های خالی و دستهٔ شغلی انجام شده است. برای انجام مجموعه‌ای از آزمایشات، آن را با تولید فهرست مرتب شدهٔ اسلات‌های موجود (شکل ۱ (الف) را ببینید) و با مجموعهٔ از پیش‌اختصاص داده شده ویژگی‌های جایگزین تولید شدهٔ کل مدل سیستم توزیعی و حصول اسلات‌های موجود از آن سازگار یافتیم. گروههای SlotGenerator و JobGenerator برای تشکیل فهرست اسلات مرتب شده و دستهٔ شغلی در خلال مجموعه‌های آزمایش مورد استفاده قرار گرفته است. در اینجا شرحی از پارامترها و مقادیر مورد استفاده در خلال شبیه‌سازی آمده است.

SlotGenerator :

- تعداد اسلات‌های سیستمی موجود در فهرست مرتب شده برحسب [۱۵۰، ۱۲۰] متفاوت است.
- طول اسلات‌های واحد برحسب [۳۰۰، ۵۰] است.
- احتمال این که اسلات‌های نزدیک در این لیست دارای زمان شروع یکسان باشند برابر با 0.4 است.
-زمان بین اسلات‌های مجاور در این لیست به صورت [۱۰، ۰] است.
- قیمت اسلات به طور تصادفی از [0.75p,1.25p] است که در آن p=(1.7) به ازای (عملکرد گره است).

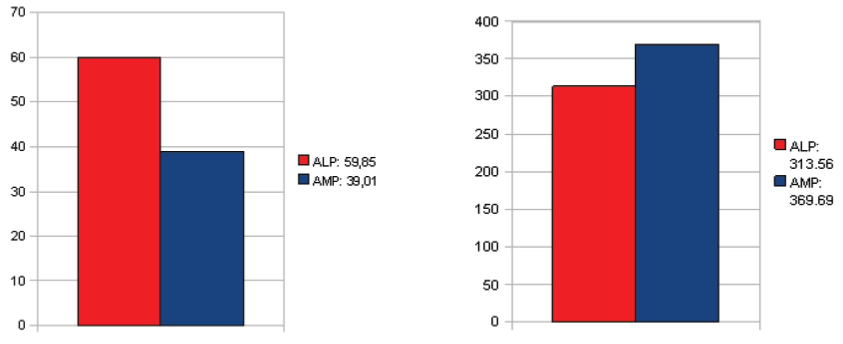
JobGenerator:

- تعداد مشاغل دستهٔ [۷،۳]؛
- تعداد گره‌های محاسباتی یافته شده در [۱، ۶]؛
- طول (پیچیدگی ارائه شده) شغل [۵۰، ۱۵۰]؛
- کمینه عملکرد گره‌های مورد نیاز [۲،۱].

تمام دسته‌های شغلی و گزینه‌های فهرست اسلات متغیرهای تصادفی هستند که دارای توزیع یکپارچه در فواصل زمانی معین هستند.

فرض کنید که کار تخصیص اسلات در خلال بیشینه‌سازی زمان اجرای کل دستهٔ شغلی برحسب فرمول (۱) باشد: $\underset{s_i}{\mathrm{mint}}(\bar{s})$ با حد b^* در (۳) باشد. فرض می‌کنیم که در (۱) : با فرض $c_i = 0$ داریم $j_i c_i = \infty$.

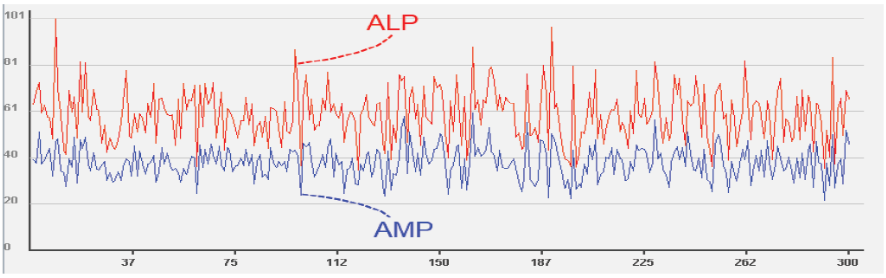
تعداد ۲۵۰۰ چرخهٔ زمانبندی شبیه‌سازی شده انجام شده بود. صرفاَ آزمایشاتی که هنگام انجام آن تمام شغل‌های دسته‌ای دارای دست کم یک آلترناتیو مناسب اجرای باشد مدنظر قرار گرفتند. هنگام مقایسه با معیار بهینه‌سازی هدف، الگوریتم AMP با میزان ۳۵ درصد، فراتر از ALP بود. کل زمان اجرای میانگین شغل دسته‌ای برای آلترناتیوهای یافت شده با ALP برابر با ۵۹،۸۵ بود و در مورد آلترناتیوهای یافت شده در AMP برابر با ۳۹،۰۱ بود (شکل ۴ (الف)). شایان ذکر است که هزینه میانگین اجرای هر شغل دسته‌ای برای روش ALP عبارت از ۳۱۳،۵۶ بود و این در حالی است که از هزینهٔ میانگین اجرای شغل الگوریتم AMP برابر با ۳۶۹،۶۹ استفاده شده که ۱۵ درصد بیشتر است ـ شکل ۴ (ب) را ببینید. مقایسهٔ نتایج زمانبندی مربوط به ۳۰۰ آزمایش اول را می‌توانید در شکل ۵ مشاهده کنید. این روند حاکی از افزایش چشمگیر روش AMP در هر آزمایش واحد است. تعداد کل آلترناتیوهای یافته شده توسط ALP برابر با ۲۵۸۰۷۹ یا میانگین ۷،۳۹ برای هر شغل بود. در همین زمان، رویکرد اصلاح شده (AMP) تعداد ۱۱۶۰۰۲۹ آلترناتیو یا میانگین ۳۴،۲۸ برای هر شغل واحد کسب کرد.



(الف) (ب)

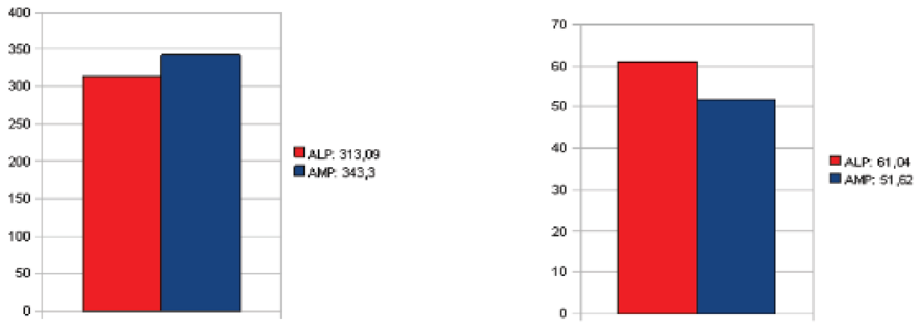
کمینه‌سازی زمان اجرای شغل دسته‌ای: میانگین کل زمان اجرای شغل (الف)؛ میانگین کل هزینهٔ اجرای شغل (ب).

بنابر نتایج این آزمایش، در می‌یابیم که استفاده از AMP، زمان اجرای کلی را از طریق افزایش هزینه اجرا ، به حداقل می‌رساند. تعداد نسبتاً زیادی از آلترناتیوها، تنوع انتخاب ترکیب اسلات کارآمد [۲] را با استفاده از الگوریتم AMP یافته است.



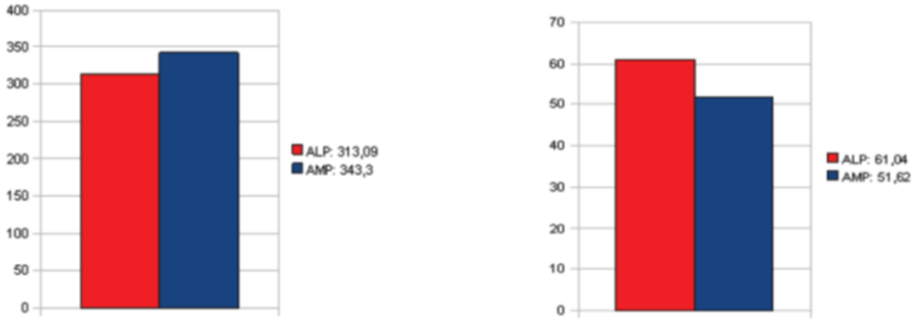
شکل ۵- مقایسه میانگین زمان اجرای ALP و AMP در کمینه‌سازی زمان اجرای دستهٔ شغلی

اکنون فرض کنیم که کار تخصیص اسلات در خلال کمینه‌سازی کل هزینه اجرای دستهٔ شغلی از طریق فرمول (۱): $\min c(\bar{s})$ با ثابت t^* در (۲) است. فرض می‌کنیم که در (۱): اگر $t_i = 0$ باشد $f_i t_i = 0$ است. نتایج مربوط به ۸۵۷۱ آزمایش واحد که در هر کدام مشاغل دسته‌ای به طور موفقیت‌آمیزی به مجموعهٔ مناسبی از منابع اختصاص یافته بودند و این منابع با استفاده از دو شیوهٔ جستجوی اسلات جمع‌آوری شده بود. میانگین هزینه اجرای کل دستهٔ شغلی برای الگوریتم ALP برابر با ۳۱۳,۰۹ و برای آلترناتیوهای یافت شده با AMP برابر با ۳۴۳,۳ بود. این روند، حاکی از نوعی مزیت در معیار هدف صرفاً ۹ درصدی برای رویکرد ALP نسبت به AMP بود (شکل ۶ (الف)). میانگین زمان اجرای کل شغل دسته‌ای برای آلترناتیوهای یافت شده در ALP برابر با ۶۱,۰۴ بود. با استفاده از الگوریتم AMP، میانگین زمان اجرای شغل برابر با ۵۱,۶۲ بود که در واقع ۱۵ درصد کمتر از روش استفاده از ALP بود (شکل ۶ (ب)).



تعداد میانگین اسلات‌های پردازش شده در آزمایش واحد برابر با ۱۳۵,۱۱ بود. این تعداد با تعداد میانگین اسلات‌های مربوط به تمام ۲۵۰۰۰ آزمایش منطبق بود و این امر حاکی از عدم تاثیر قاطع تعداد اسلات‌های موجود به تعداد مشاغل زمانبندی شده موفق بود. تعداد میانگین مشاغل دسته‌ای در یک چرخهٔ زمانبندی واحد برابر با ۴,۱۸ بود. این مقدار کوچکتر از میانگین کلی ۲۵۰۰۰ آزمایش است. با تعداد زیادی از مشاغل ALP دسته‌ای، اغلب قادر به یافتن مجموعه‌های آلترناتیو اسلات‌ها برای مشاغل معین نخواهیم بود و آزمایش لحاظ نشد.

میانگین تعداد آلترناتیوهای یافت شده با ALP برابر با ۲۵۳۸۵۵ یا میانگین ۷,۲۸ به ازای هر شغل بود. الگوریتم AMP قادر به یافتن تعداد ۱۱۵۱۱۶ آلترناتیو یا میانگین ۳۴,۲۳ به ازای هر شغل بود. خاطرنشان می‌شود که در مجموعه آزمایشات پیشین، این تعداد به ترتیب ۷,۳۹ و ۳۴,۲۸ بود.



الف

ب

شکل ۶- کمینه‌سازی هزینه اجرای کل دستهٔ شغلی: میانگین کلی هزینه اجرای شغل (الف)؛ میانگین کلی زمان اجرای شغل (ب).

مباحث

با توجه به نتایج آزمیشات، می‌توان گفت که استفاده از رویکرد AMP در مرحلهٔ جستجوی آلترناتیوها، در مقایسه با کاربرد ALP «معمولی» مزیت بیشتری دارد. این مزایا اغلب در تعداد زیادی از آلترناتیوها یافت شده و در نتیجه در انعطاف‌پذیری انتخاب برنامهٔ موثر اجرای دسته‌ای وجود داشته و نیز AMP زمان کل اجرای شغلی کمتری نسبت به ALP را فراهم می‌آورد. AMP امکان جستجوی آلترناتیوها را میان گره‌های پردازشگر نسبتاً پرهزینه با میزان عملکرد بالاتر فراهم می‌سازد. مجموعه‌های آلترناتیو اسلات‌ها در ALP از تجانس بیشتری برخوردار بوده و تفاوت چندانی با سایر مقادیر زمان و هزینه کل اجرا ندارد. بنابراین، توزیع دسته‌ای شغل‌ها از طریق بهینه‌سازی مبتنی بر معیارهای متفاوت [۲] تفاوت چندانی با سایرین ندارد. فاکتورهای ذیل به تشریح نتایج می‌پردازند. نخست، مشخصه‌های محاسبهٔ هزینه کل کاربرد اسلات $c_t = ctn/p$ که در آن c هزینه کاربرد اسلات به ازای واحد زمانی و p میزان عملکرد نسبی گره پردازشگر که اسلات بدان اختصاص یافته و t بازه زمانی است و به واسطه این شغل با فرض اینکه این شغل در گره‌های اتالونی با pضروری است اجرایی خواهد شد. در مدل پیشنهادی، هزینهٔ c اسلات بالاتر از عملکرد P گرهی است که اسلات بدان اختصاص یافته است. از این رو، زمان اجرای شغل t/P نسبتاً کم است. بنابراین، هزینه بالای اسلات به ازای واحد زمانی از طریق عملکرد بالای گرهٔ CPU جبران می‌شود و بنابراین زمان کمتری برای انجام شغل و واحدهای زمانی کمتری برای پرداختن به آن مورد نیاز است.

بنابراین، در برخی موارد، کل هزینهٔ اجرای یکسان باقی می‌ماند حتی اگر اسلات‌ها پرهزینه باشند. مقدار C/P معیار نسبت هزینه / کیفیت است. با فرض اینکه در درخواست منبع، هزینهٔ بیشینه C یک اسلات جداگانه و میزان عملکرد کمینهٔ P یک گره باشد، کاربر مقدار کمینهٔ قابل قبول قیمت / کیفیت را تعیین می‌کند. تفاوت بین رویکردهای ALP و AMP در ان است که جستجوهای ALP در مورد آلترناتیوهای با ضریب قیمت / کیفیت مناسب میان اسلات‌های با هزینه استفاده کمتر از C است. AMP جستجو میان تمام اسلات‌های موجود را انجام می‌دهد (به طور معمول، هر دو الگوریتم کماکان دارای محدودیت در عملکرد کمینه گره قابل پذیرش هستند). این روند به تشریح علت این امر که چرا آلترناتیوهای یافته شده از طریق AMP دارای زمان اجرایی کل کمتری هستند می‌پردازد.

دوم، این امر خاطرنشان می‌سازد که در خلال جستجو ALP، اسلات‌های موجود صرفنظر از کل «پنجره» مورد توجه قرار گرفته است. پنجرهٔ ALP مشتمل بر اسلات‌هایی است که هریک دارای ارزش هزینه کمتر از C هستند. در زمان مشابه AMP، انعطاف‌پذیری بیشتری وجود دارد.اگر در برخی مراحل، اسلات با هزینه δ کمتر از C به پنجرهٔ مطلوب افزوده شده باشد، در نتیجه الگوریتم AMP برای افزودن اسلات‌ها به هزینه بنا بر δ بیشتر از C در مراحل جدید لحاظ خواهند شد. به طور عادی، در این مورد، محدودیت کلی هزینه در

نظر گرفته شده است. این همان علت بیشتر بودن میانگین هزینه اجرای شغل هنگام استفاده از الگوریتم AMP است زیرا این روند به دنبال استفاده از کل بودجه برای یافتن نخستین آلترناتیو مناسب است. نکته دیگر عملکرد الگوریتم در مجموعهٔ مشابهی از اسلات‌ها است. می‌توان گفت که هر پنجره‌ای که بتواند با ALP یافت شود می‌تواند از طریق AMP نیز پیدا شود. با وجود این، می‌توان پنجره‌ها را با الگوریتم AMP یافت که به طور معمول در ALP یافت نمی‌شوند. یافتن پنجره‌ای که مشتمل بر دست کم یک اسلات با هزینه بیشتر از C باشد کافی است. این روند حاکی از میانگین رویکرد AMP به واسطهٔ تعداد الگوریتم‌های یافت شده است. کمبود طرح AMP عبارت از کل هزینه اجرای دسته‌ای میانگین است که همیشه بالاتر از هزینه اجرای کاربرد زمانبندی شده دسته مشابه از طریق الگوریتم ALP است. این روند پیامد اختصاصی تعیین مقدار حد بودجه و مرحله زمانبندی دسته شغلی است [۲]. با وجود این، می‌توان هزینه اجرای کل دسته را برای حد بودجه کاربر در هر آلترناتیو یافته شده در هر جستجو کاهش داد و این همان آزمایش محدود به $s = ctn$ است. این فرمول را می‌توان به صورت $s = pctn$ اصلاح کرد که در آن p عدد مثبت و کمتر از یک است (برای نمونه 0.8). تنوع p امکان توزیع انعطاف‌پذیر در چرخه‌های زمانبندی مختلف را بنا به ساعت زمانی روز، سطح بار منبع و غیره فراهم می‌آورد [۲].

معرفی الگوریتم موازی و توضیح مساله:

مسئله موازی را می توان به m وظیفه ارتباطی نشان داده شده با یک گراف بدون جهت $g_t = (v_t, e_t)$ تقسیم نمود که v_t مجموعه رئوس e_t:$\{t_1, t_2, \dots, t_m\}$ مجموعه لبه های برچسب گذاری شده با هزینه های ارتباطی بین رئوس را نشان می دهد. شبکه متصل n پردازنده ای، $\{p_1, p_2, \dots, p_n)$ با یک ماتریکس $l\ n^*n$ نشان داده شده است که l_{ij} برابر با ۱ می باشد در صورتی که پردازنده های iو j به هم متصل باشند و در غیر این صورت صفر می باشد.

وظیفه t_i از مجموعه v_t را می توان برروی هر یک از n پردازنده سیستم اجرا نمود. هر وظیفه دارای یک هزینه اجرایی وابسته به خود برروی یک پردازنده می باشد. هزینه های اجرای وظایف با ماتریکس X مشخص شده اند؛ x_{ip} هزینه اجرای وظیفه iبرروی پردازنده p می باشد. زمانی که دو وظیفه t_i و t_j اجرا شده برروی دو پردازنده متفاوت نیاز به مبادله داده ها داشته باشند، آنگاه هزینه ارتباطی تحمیل خواهد شد.

ارتباط میان وظایف با ماتریکس C نشان داده شده است که c_{ij} هزینه ارتباطی بین وظیفه I و j را نشان می دهد به شرطی که برروی دو پردازنده متفاوت قرار گرفته باشند. بار روی پردازنده ترکیبی از کلیه هزینه های اجرایی و ارتباطی وابسته به وظایف تخصیص داده شده به آن می باشد. زمان کلی تکمیل برنامه به زمان مورد نیاز توسط پردازنده ای با بیشترین یا سنگین بار اشاره می کند.
مسئله تخصیص وظیفه، در واقع یافتن نگاشت مجموعه m وظیفه به n پردازنده می باشد به گونه ای که زمان کل تکمیل کاربه حداقل می رسد. نگاشت یا تخصیص وظایف به پردازنده ها با ماتریکس A نشان داده شده است، که در اینجا A_{ip} ۱ است در صورتی که iبه پردازنده pاختصاص داده شده باشد و درغیر این صورت صفر می باشد. بار روی پردازنده p از رابطه زیر بدست می آید:

$$\sum_{i=1}^{m} x_{ip} \bullet a_{ip} + \sum_{q=1}^{n} \sum_{i=1}^{m} \sum_{j=1}^{m} (c_{ij} a_{ip} a_{jq} l_{pq})$$

$(p \neq q)$

جزء اول معادله هزینه کل اجرای وظایف تخصیص داده شده به پردازنده p و جزء دوم سربار ارتباطی برروی p را نشان می دهد.
به منظور یافتن پردازنده ای با سنگین ترین بار، باررروی هر یک از n پردازنده باید محاسبه شود. تخصیص بهینه حال، مسئله ای است که موجب مینیموم بار برروی سنگین ترین پردازنده در میان کلیه تخصیص ها می شود. n^m تخصیص ممکن وجود دارد و یافتن تخصیص بهینه، معروف به مسئله $np - hard$ می باشد.

بررسی اجمالی تکنیک A^*

a^* الگوریتم جستجوی بهترین- نخست است که از آن برای حل مسائل بهینه سازی در هوش مصنوعی و سایر بخشها استفاده شده است. این الگوریتم مسئله را به صورت یک درخت جستجو ترسیم می کند. سپس، گره های درخت را جستجو می کند که از ریشه شروع شده و گره آغازین نام دارند (معمولاً راه حل صفر). هزینه وابسته به هر گره با تابع هزینه f محاسبه می شود.گره ها بر طبق این هزینه، برای جستجو مرتب می شوند، یعنی، گرهی با مینیموم هزینه اول جستجو می شود. الگوریتم لیست به هم پیوسته ای از گره ها (بر طبق مقادیر f گره ها) را نگهداری کرده و همیشه گرهی با بهترین هزینه برای بسط را انتخاب می کند.

بسط دهی گره در واقع به تولید کلیه جانشینان یا بچه ها اشاره می کند. از آنجایی که این الگوریتم همیشه بهترین گره هزینه را انتخاب می کند، در نتیجه یک راه حل بهینه را تضمین می نماید.

شیوه ترتیبی

در این مطالعه از تکنیک a^* برای مسئله تخصیص استفاده کرده و آن را الگوریتم تخصیص بهینه با جستجوی ترتیبی (OASS) می نامیم. الگوریتم OASS به صورت زیر توصیف شده است:

```
Build initial node s and insert it into the list OPEN
Set f(s)=0
Repeat
    Select the node n with smallest f value.
    if n <> Solution
        Generate successors of n
        for each successor node n' do
            if (n' is not at the last level in the search tree
                $fn' = gn' + h(n')$
            else $fn' = gn'$
            Insert n' into OPEN
        end for
    end if
    if (n = solution )
        Report the solution and stop
Until n is Solution or (OPEN is empty)
```

Shen و Tsai [13] یک الگوریتم جستجوی فضای حالت برای نیازمندیهای بهینه فرمول نویسی کردند. در این فرمول، هر گره در درخت جستجو، معرف یک تخصیص جزئی بوده و گره هدف، تخصیص کامل را نشان می دهد. سپس از الگوریتم A^* برای پیمایش فضای جستجو استفاده شده است.

مطالعه بعدی Ramakrishnan [13] نشان داد که مرتبه در نظر گرفتن وظایف برای تخصیص تاثیر زیادی بر عملکرد وکارایی الگوریتم (برای تابع هزینه بکاررفته) اعمال می نماید. مطالعه آنها نشان داد که با مرتب سازی دقیق وظایف می توان کارایی را ارتقاء داد.

آنها تعدادی فرایند اکتشافی پیشنهاد کردند که فرایند تعیین توالی مینی ماکس بهترین عملکرد را نشان داده است.

شکل ۱. نمونه ای از گراف وظیفه و پردازنده و شبکه، هزینه های اجرای وظایف برروی پردازنده های مختلف را نشان می دهد.

شکل ۲. درخت جستجو برای مسئله نمونه را نشان می دهد (گره های تولید شده= ۳۹، گره های بسط داده شده= ۱۳). به فرض وجود مجموعه ۵ وظیفه، $\{t_0, t_1, t_2, t_3, t_4\}$ و مجموعه سه پردازنده $\{p_0, p_1, p_2\}$ به صورت نشان داده شده در شکل ۱، آنگاه

ISSN: 2745-3673
http://www.Rcsj.ir

درختهای جستجوی حاصله در شکل ۲ نشان داده شده اند. گره در درخت جستجو از تخصیص جزئی وظایف به پردازنده ها و مقدار f (هزینه تخصیص جزئی) تشکیل می شود. تخصیص m وظیفه به n پردازنده با رشته m رقمی $'a_0 a_1 \ldots a_{m-1}'$ نشان داده شده است که $a_i (0 \leq i \leq m-1)$ پردازنده (صفر تا n-1) را نشان می دهد که iامین وظیفه به آن اختصاص داده شده است. تخصیص جزئی بدان معناست که برخی وظایف تخصیص داده نشده اند؛ مقدار a_i برابربا $'X'$ نشان می دهد که iامین وظیفه تخصیص داده نشده است. هر سطح از درخت نظیر یک وظیفه می باشد، بنابراین، مقدار $'X'$ در رشته تخصیص را با اعداد پردازنده عوض می کند. بسط دهی گره در واقع به اضافه شدن تخصیص وظیفه جدید به تخصیص جزئی اشاره می کند. بنابراین عمق (D) درخت جستجو برابر با تعداد وظایف m بوده و هر گره درخت دارای ماکزیمم n جانشین (بدون پردازنده) می باشد.

گره ریشه از مجموعه کل وظایف تخصیص داده نشده $'XXXXX'$ تشکیل می شود. به طور مثال، در شکل ۲، تخصیص t_0 به $'0XXXX'$، t_0 به $P_1('1XXXX')$، و به $p_2('2XXXX')$ با تعیین هزینه های تخصیص در سطح اول درخت در نظر گرفته شده است.

تخصیص t_0 به $P_0('0XXXX')$، موجب می گردد هزینه کل $f(n)$ برابر با ۳۰ باشد. در این مورد $g(n)$ برابر با ۱۵ می باشد که هزینه اجرای t_0 برروی p_0 می باشد. در این مورد $h(n)$ نیز برابر با ۱۵ می باشد که مجموع مینیموم اجرا یا هزینه های ارتباطی t_1 و t_4 (ارتباط وظایف با t_0) را نشان می دهد. هزینه های تخصیص t_0 به $p_1\ (26)$ و t_0 به $p_2\ (24)$ به صورت مشابه محاسبه شده است. این سه گره در لیست OPEN درج شده اند. از آنجایی که ۲۴ مینیموم هزینه می باشد، در نتیجه گره $'2XXXX'$ برای بسط دهی انتخاب شده است. جستجو تا زمانی ادامه می یابد که گرهی با تخصیص کامل $('20112')$ برای بسط دهی انتخاب شده باشد. در این نقطه، از آنجایی که این گرهی با تخصیص کامل و مینیموم هزینه است، در نتیجه گره هدف محسوب می گردد. شایان توجه است که کلیه رشته های تخصیص منحصر به فرد می باشند. کلاً ۳۹ گره تولید و ۱۳ گره بسط داده شده است. در مقایسه، جستجوی جامع به منظور یافتن راه حل بهینه $n^m = 243$ تولید می کند. توالی مینی ماکس تولید شده به صورت $\{t_0, t_1, t_2, t_4, t_3\}$ می باشد. بنابراین، t_4 قبل از t_3 در نظر گرفته شد.

شیوه موازی سازی

به منظور تمایز بین پردازنده هایی که الگوریتم تخصیص وظیفه موازی از پردازنده ها در حوزه مسئله برروی آنها اجرا می شود، مورد اول را با PE نشان می دهیم (عنصر پردازش که در این مورد پردازنده Intel Paragon می باشد). ما الگوریتم موازی را الگوریتم تخصیص بهینه با جستجوی موازی (OAPS) می نامیم. ابتدا راجع به استراتژی های موازنه بار دینامیکی (پویا) و تقسیم بندی اولیه توضیح می دهیم.

تقسیم بندی اولیه

ابتدا فضای جستجو به صورت ایستا و بر اساس تعداد عناصر پردازش p(PEs) در سیستم و ماکزیمم تعداد جانشینان S گره در درخت جستجو تقسیم می شود. سه وضعیت در اینجا وجود دارد:

مورد ۱) p < s: هر PE فقط گره اول را بسط می دهد که این مسئله موجب شکل گیری و تولید S گره جدید می شود. هر PE یک

گره دریافت کرده و گره های اضافی را به شیوه Round Robin (RR) دریافت می کند. مورد ۲) P=S : فقط گره اول بسط داده شده و هر PE یک گره دریافت می کند. مورد ۳) P<S: هر PE گره های بسط داده شده از گره اولیه را حفظ می کند تا زمانی که تعداد گره ها در لیست بیشتر یا برابر با P شود. لیست به صورت مقادیر هزینه افزایشی گره ها مرتب می شود. گره اول در لیست به PE_1، گره دوم به PE_2، گره سوم به PE2، گره چهارم به PE_{p-1} رفته و این روند به همین ترتیب ادامه می یابد. با استفاده از RR گره های زیادی توزیع خواهند شد. (اگرچه تضمینی نیست که بعد از مقداری بسط دهی، گره بهترین هزینه منجر به گره هزینه خوب می گردد اما الگوریتم

سعی می کند گره های خوب را تا حد امکان به صورت یکنواخت در میان PEها توزیع نماید). اگر راه حل در طول این فرایند یافت شود، آنگاه الگوریتم خاتمه می یابد. شایان توجه است که هیچ PE اصلی اول گره هارا تولید نکرده و سپس آنها را در میان سایر PE ها توزیع نماید.

موازنه و تعادل بار پویا

اگرهیچ گونه ارتباطی بین PE ها بعد از تخصیص ایستا اول وجود نداشته باشد، آنگاه برخی از آنها برروی بخش خوبی از فضای جستجو کار می کنند، در حالیکه دیگران گره های غیر ضروری را بسط می دهند (گره هایی که الگوریتم سریالی بسط نخواهد یافت). این مسئله موجب سرعت بخشی و تسریع ضعیف می شود. به منظور اجتناب از این مسئله، PE ها نیاز به برقراری ارتباط دارند تا بدین طریق بتوانند از بهترین بخش فضای جستجو سهم بوده و از کار غیر ضروری اجتناب نمایند. در این فرمول، PE صریحاً با استفاده از استراتژی ارتباطی RR در همسایگی اش و به طور ضمنی با انتشار راه حل برای کلیه PE ها، به این مهم دست می یابد. در مراحل ۱۶-۱۳ الگوریتم، PE به طور متناوب (زمانی که OPEN تا حد آستانه u افزایش می یابد) همسایه را به شیوه RR انتخاب کرده و سپس بهترین گره را برای آن همسایه ارسال می کند. بدین طریق از بهترین بخش فضای جستجو در همسایگی استفاده می کند. به غیر از موازنه بار، PE راه حلش (زمانی که یک راه حل می یابد) را نیز برای کلیه PE ها منتشر می کند. بدین طریق مانع از کار غیر ضروری برای PE می شود که برروی بخش بد فضای جستجو مشغول کار می باشد. زیرا به محض اینکه گره ، راه حل هزینه بهتری نسبت به بهترین گره فعلی دریافت می کند، بسط دهی گره های غیر ضروری را متوقف می کند. راه حل فقط در صورتی منتشر می شود که هزینه اش بهتر از راه حل قبلی دریافت شده از PE دیگر باشد. الگوریتم OAPS به صورت زیر توصیف شده است:

الگوریتم OAPS:

```
(1) Init- Partition()
(2) SetUP-Neighborhood()
(3)Repeat
(4) Expand the best cost node from OPEN
(5) if (a Solution found)
(6) if (it’s better than previously received solutions )
(7) Broad Cast the solution to all PEs
(8)  else
(9)    inform neighbors that I am done
(10)   end if
(11)  Record the Solution and stop
(12) end if
(13)   If (OPEN’s length increases by a threshold u)
(14)     Select a neighbor PE j using RR
(15)     Send the current best node from OPEN to j
(16)    end if
(17)    if (Received a node from neighbor)
(18)     Insert it to OPEN
(19)     if (Received a Solotion from a PE)
(20)       Insert it to OPEN
(21)       if (Sender is a neighbor)
(22)      Remove this from neighborhood list
(23)  end if
(24) Until (OPEN is empty)OR(OPEN is full)
```

با تقسیم بندی اولیه، هر PE ابتدا همسایه اش را مشخص می کند تا بدین طریق بفهمد کدام PE ها در همسایگی اش قرار دارند.

برخی گره های اولیه برای هر PE تولید سپس از گره های اولیه آغاز می شوند، هر PE تکرارهایی از A^* ترتیبی را اجرا خواهد نمود. سپس PE ها با یکدیگر برای مبادله بهترین گره ها و انتشار راه حل ها، وارد تعامل می شوند. زمانی که PE راه حلی می یابد، آن را در فایل

مشترک بین همه PEها رکوردو ثبت می نماید. PE که راه حل را می یابد، گره ها را بیشتر بسط نداده و منتظر دریافت راه حل از PE های دیگر می ماند. بالاخره، بهترین راه حل، راه حلی با مینیموم هزینه در میان کلیه PEها می باشد. به منظور توضیح در مورد عملیات الگوریتم OAPS، از مثال بکاررفته در بخش قبل برای الگوریتم تخصیص ترتیبی استفاده می کنیم. این عملیات در شکل ۳ نشان داده شده است. در اینجا فرض می کنیم الگوریتم موازی برروی سه PE متصل به هم به صورت یک زنجیره خطی اجرا می شود، به عبارتی PE_0 و PE_2 دارای یک همسایه PE_1 ، و PE_1 دارای دو همسایه می باشد، زیرا در قسمت وسط قرار دارد. ابتدا، سه گره به صورت مورد ترتیبی تولید می شوند. سپس از طریق تقسیم بندی اولیه، این گره ها به 3PE تخصیص داده می شوند. سپس هر PE مراحلی را طی می کند. در هر مرحله، دو فاز وجود دارد: فاز بسط دهی و فاز ارتباطی. در فاز بسط دهی، PE گره هایش را به صورت ترتیبی بسط می دهد (گره های جدیداً ایجاد شده با مرزهای ضخیم نشان داده شده اند). روند بسط دهی تا زمان رسیدن به آستانه (u) ادامه می یابد- در این مثال، این آستانه ۳ در نظر گرفته شده است. در فاز ارتباطی، PE همسایه را انتخاب و سپس گره بهترین هزینه را برایش ارسال می کند. انتخاب همسایه ها به شیوه RR صورت می گیرد. در مثال، مبادله گره های بهترین هزینه در میان همسایه ها با پیکان های نقطه چین نشان داده شده است. در مرحله پنجم، PE1 راه حلش را یافته، آن را برای سایر PE ها منتشر کرده و سپس متوقف می شود. در مرحله آخر، PE0 نیز راه حلش را بر PE2 (در اینجا برای سهولت کار نشان داده نشده است) منتشر کرده و بالاخره راه حلش را رکورد و متوقف می شود.

نتیجه‌گیری و پژوهش‌های آتی

در این مقاله، به مسئله زمانبندی دسته‌های شغلی مستقل در محیط نامتجانس با منابع تفکیک‌ناپذیر پرداختیم. زمانبندی دسته‌ای شغل مشتمل بر دو فاز است. نخست، مجموعه‌های مستقل اسلات‌های مناسب که باید برای هر شغل یافته شود. فاز دوم انتخاب ترکیب موثر اسلات‌های آلترناتیو است.برای نیل به اهداف رویکردهای ALP و AMP ، جستجوی اسلات و تخصیص آنها مورد مقایسه قرار گرفته است. براساس نتایج تجربی، می‌توان گفت که AMP امکان یافتن آلترناتیوهای سریعتری را در زمان کمتر برای انجام مشاغل فراهم می‌آورد. با وجود این، هزینه کلی اجرای شغلی با استفاده از AMP نسبتاً بالاتر است. هنگام مقایسه با معیار بهینه‌سازی هدف در خلال کمینه‌سازی زمان کل اجرای دسته‌ای، AMP به طور چشمگیری فراتر از ALP «معمول» عمل می‌کند. در همین زمان نیز به معرفی روش های طراحی الگوریتم های موازی تخصیص بهینه در بهبود سیستم های توضیع شده و همچنین ارائه نمونه و روش کارکرد ان پرداختیم. ما مسئله را تحت حداقل فرضیه نظیر گراف وظیفه اختیاری من جمله هزینه های اختیاری برروی گره ها و لبه های گراف، و پردازنده های متصل به هم از طریق یک شبکه اتصال در نظر گرفتیم.

در پژوهش آتی، به الگوریتم‌های مشترک تخصیص یافته در راستای ادغام آنها با راهبردهای مشترک زمانبندی مقیاس‌پذیر با بهره گیری از الگوریتم های موازی خواهیم پرداخت.برای درک بهتر مفاهیم و فهم رفتار الگوریتم های ارائه شده و برخی اصلاحات ظریف، اجرای مطالعات آتی ضروری می باشد. در حال حاضر ما مشغول کشف این احتمالات هستیم.

ISSN: 2745-3673
http://www.Rcsj.ir

منابع و مراجع

[1] S.K. Garg, R. Buyya, and H.J. Siegel, Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management, Proceedings of the 32nd Australasian Computer Science Conference (ACSC 2009), Wellington, New Zealand (2009) 151-159.

[2] V.V. Toporkov, A. Toporkova, A. Tselishchev, D. Yemelyanov, and A. Bobchenkov, Economic Models of Scheduling in Distributed Systems, In: T. Walkowiak, J. Mazurkiewicz, J. Sugier, and W. Zamojski (eds.), Monographs of System Dependability. Dependability of Networks. – Wroclaw: Oficyna Wydawnicza Politechnki Wroclawskiej 2 (2010) 143-154.

[3] J. P. Degabriele and D. Pym, Economic Aspects of a Utility Computing Service, Trusted Systems Laboratory HP Laboratories Bristol HPL-2007-101, Technical Report (2007) 1-23.

[4] A. Ailamaki, D. Dash, and V. Kantere, Economic Aspects of Cloud Computing, Flash Informatique, Special HPC (2009) 45-47.

[5] J. Bredin, D. Kotz, and D. Rus, Economic Markets as a Means of Open Mobile-Agent Systems, Proceedings of the Workshop “Mobile Agents in the Context of Competition and Cooperation (MAC3)”, 1999, 43-49.

[6] R. Buyya, D. Abramson, and J. Giddy, Economic Models for Resource Management and Scheduling in Grid Computing, J. of Concurrency and Computation: Practice and Experience 5 (14) (2002) 1507 – 1542.

[7] C. Ernemann, V. Hamscher, and R. Yahyapour, Economic Scheduling in Grid Computing, Proceedings of the 8th Job Scheduling Strategies for Parallel Processing, D.G. Feitelson, L. Rudolph, and U. Schwiegelshohn (eds.), Springer, Heidelberg, LNCS 2537 (2002) 128-152.

[8] K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz, Multicriteria Aspects of Grid Resource Management, In: Grid resource management. State of the art and future trends, J. Nabrzyski, J.M. Schopf, and J.Weglarz (eds.): Kluwer Academic Publishers, 2003, 271 – 293.

[9] V. Toporkov, Application-level and Job-flow Scheduling: an Approach for Achieving Quality of Service in Distributed Computing, Proceedings of the 10th International Conference on Parallel Computing Technologies, Springer, Heidelberg, LNCS 5698 (2009) 350 – 359.

[10] V.V. Toporkov, Job and Application-Level Scheduling in Distributed Computing, Ubiquitous Computing and Communication Journal .Special Issue on ICIT 2009 Conference (selected papers) - Applied Computing 3 (4) (2009) 559 – 570.

[11] A.W. Mu'alem and D.G. Feitelson, Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling, IEEE Transactions on Parallel and Distributed Systems 6 (12) (2001) 529-543.

[12] D. Jackson, Q. Snell, and M. Clement, Core Algorithms of the Maui Scheduler, Springer, Heidelberg, LNCS 2221 (2001) 87-102.

[13] V.V. Toporkov and A. Tselishchev, Safety Scheduling Strategies in Distributed Computing, International Journal of Critical Computer-Based Systems, 1/2/3 (1) (2010) 41-58.

[14] V.V. Toporkov, A. Toporkova, A. Tselishchev, and D. Yemelyanov, Scalable Co-Scheduling Strategies in Distributed Computing,

[15] S. H. Bokhari, “On the Mapping Problem,” IEEE Trans. On Computers vol. C-30, March 1981, pp. 207-214.

[16] T. Bultan and C. Aykanat, “A new heuristic based on mean field annealing,” Joumal of Parallel and Distributed Computing, vol. 16, no. 4, pp 292-305, Dec 1992.

[17] V. Chaudhary and J. K. Aggarwal, “A generalized scheme for mapping parallel algorithms,” IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 3, Mar 1993.

[18] Garey and D. S. Johnson, Computers and ctability: A Guide to the Theory of NP-completeness, (Freeman, San Francisco, CA, 1979).

[19] D. E. Goldberg, “Genetic algorithms in search, optimization, and Machine leaming,” (Addison, Wesely, Reading, MA 1989)

[20] S. Hurley, “Taskgraph Mapping Using a Genetic Algorithm: A comparision of Fitness Functions,” Parallel Computing, vol. 19, pp. 1313-1317, NOV 1993.

ISSN: 2745-3673
http://www.Rcsj.ir

[21] V. Mary Lo, “Heuristic Algorithms for Task Assignment in Distributed Systems,” IEEE Trans on Computers. vol37, no 11, Nov1988.

[22] P.-Yio R. Ma, E. Y. S Lee, “A Task Allocation Model for Distributed Computing Systems,”IEEE Tram on Computers, vol. c-31, no. 1, Jan. 1982.

[23] N. J. Nilson, Problem Solving Methods in Artificial Intellegence. New York McGraw-Hill, 197 1. [IO] S. Ramakrishnan, H. Chao, and L.A. Dunning, “A Close Look at Task Assignment in Distributed Systems,” IEEE [ll] C.-Ch. Shen and W.-H. Tsai, “A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using a Minimax Criterion,” IEEE Trans on Computers, vol. c-34, no. 3, pp. 197-203, March 1985.

[24] H. S. Stone, “Multiprocessor Scheduling with the aid of Network Flow Algorithms,” IEEE Trans. Software Engineering, SE-3, vol. 1. pp. 85-93, Jan 1977. 1131 J. B. Sinclair, “Efficient Computation of Optimal Assignments for Distributed Tasks,” Journal of Parallel and